Guile-GnuTLS

Guile binding for GnuTLS
for version 4.0.0.2-6da1, 27 August 2023

Ludovic Courtès

This manual is last updated 27 August 2023 for version 4.0.0.2-6da1 of Guile-GnuTLS.

Table of Contents

1 Preface

This manual describes `Guile-GnuTLS`, the GNU Guile (`https://www.gnu.org/software/guile/`) Scheme programming interface to GnuTLS (`https://gnutls.org`). The reader is assumed to have basic knowledge of the TLS protocol and GnuTLS library (see Section "Introduction to GnuTLS" in *GnuTLS Manual*).

At this stage, not all the C functions are available from Scheme, but a large subset thereof is available.

2 Guile Preparations

The Guile bindings for GnuTLS are available for the Guile 3.0 and 2.2 series, as well as the legacy 2.0 series.

By default they are installed into `/usr/local/share/guile/site/`). Normally Guile will not find the module there without help. You may experience something like this:

```
$ guile
...
scheme@(guile-user)> (use-modules (gnutls))
ERROR: no code for module (gnutls)
```

There are two ways to solve this. The first is to make sure that when building Guile-GnuTLS, the bindings will be installed in the same place where Guile looks. You may do this by using the `--with-guile-site-dir` parameter as follows:

```
$ ./configure --with-guile-site-dir=no
```

This will instruct Guile-GnuTLS to attempt to install the bindings where Guile will look for them. It will use `guile-config info pkgdatadir` to learn the path to use.

If Guile was installed into `/usr`, you may also install Guile-GnuTLS using the same prefix:

```
$ ./configure --prefix=/usr
```

If you want to specify the path to install the Guile bindings you can also specify the path directly:

```
$ ./configure --with-guile-site-dir=/opt/guile/share/guile/site
```

The second solution requires some more work but may be easier to use if you do not have system administrator rights to your machine. You need to instruct Guile so that it finds the Guile-GnuTLS bindings. Either use the `GUILE_LOAD_PATH` environment variable as follows:

```
$ GUILE_LOAD_PATH="/usr/local/share/guile/site:$GUILE_LOAD_PATH" guile
scheme@(guile-user)> (use-modules (gnutls))
scheme@(guile-user)>
```

Alternatively, you can modify Guile's `%load-path` variable (see Section "Build Config" in *The GNU Guile Reference Manual*).

At this point, you might get an error regarding `guile-gnutls-v-2` similar to:

```
gnutls.scm:361:1: In procedure dynamic-link in expression (load-extension "guile-gnutl
gnutls.scm:361:1: file: "guile-gnutls-v-2", message: "guile-gnutls-v-2.so: cannot open
```

In this case, you will need to modify the run-time linker path, for example as follows:

```
$ LD_LIBRARY_PATH=/usr/local/lib GUILE_LOAD_PATH=/usr/local/share/guile/site guile
scheme@(guile-user)> (use-modules (gnutls))
scheme@(guile-user)>
```

To check that you got the intended GnuTLS library version, you may print the version number of the loaded library as follows:

```
$ guile
scheme@(guile-user)> (use-modules (gnutls))
scheme@(guile-user)> (gnutls-version)
"4.0.0.2-6da1"
scheme@(guile-user)>
```

3 Guile API Conventions

This chapter details the conventions used by Guile API, as well as specificities of the mapping of the C API to Scheme.

3.1 Living on the cutting edge

Some GnuTLS features have been introduced recently. To keep compatibility with older GnuTLS versions, they may not all be available. If a GnuTLS feature is not available, then its corresponding variable will not be exported by the `(gnutls)` module.

3.2 Enumerates and Constants

Lots of enumerates and constants are used in the GnuTLS C API. For each C enumerate type, a disjoint Scheme type is used—thus, enumerate values and constants are not represented by Scheme symbols nor by integers. This makes it impossible to use an enumerate value of the wrong type on the Scheme side: such errors are automatically detected by type-checking.

The enumerate values are bound to variables exported by the `(gnutls)` module. These variables are named according to the following convention:

- All variable names are lower-case; the underscore `_` character used in the C API is replaced by hyphen `-`.
- All variable names are prepended by the name of the enumerate type and the slash `/` character.
- In some cases, the variable name is made more explicit than the one of the C API, e.g., by avoid abbreviations.

Consider for instance this C-side enumerate:

```
typedef enum
{
  GNUTLS_CRD_CERTIFICATE = 1,
  GNUTLS_CRD_ANON,
  GNUTLS_CRD_SRP,
  GNUTLS_CRD_PSK
} gnutls_credentials_type_t;
```

The corresponding Scheme values are bound to the following variables exported by the `(gnutls)` module:

```
credentials/certificate
credentials/anonymous
credentials/srp
credentials/psk
```

Hopefully, most variable names can be deduced from this convention.

Scheme-side “enumerate” values can be compared using `eq?` (see Section “Equality” in *The GNU Guile Reference Manual*). Consider the following example:

```
(let ((session (make-session connection-end/client)))

  ;;
```

```
;; ...
;;

;; Check the ciphering algorithm currently used by SESSION.
(if (eq? cipher/arcfour (session-cipher session))
    (format #t "We're using the ARCFOUR algorithm")))
```

In addition, all enumerate values can be converted to a human-readable string, in a type-specific way. For instance, `(cipher->string cipher/arcfour)` yields `"ARCFOUR 128"`, while `(key-usage->string key-usage/digital-signature)` yields `"digital-signature"`. Note that these strings may not be sufficient for use in a user interface since they are fairly concise and not internationalized.

3.3 Procedure Names

Unlike C functions in GnuTLS, the corresponding Scheme procedures are named in a way that is close to natural English. Abbreviations are also avoided. For instance, the Scheme procedure corresponding to `gnutls_certificate_set_dh_params` is named `set-certificate-credentials-dh-parameters!`. The `gnutls_` prefix is always omitted from variable names since a similar effect can be achieved using Guile's nifty binding renaming facilities, should it be needed (see Section "Using Guile Modules" in *The GNU Guile Reference Manual*).

Often Scheme procedure names differ from C function names in a way that makes it clearer what objects they operate on. For example, the Scheme procedure named `set-session-transport-port!` corresponds to `gnutls_transport_set_ptr`, making it clear that this procedure applies to session.

3.4 Representation of Binary Data

Many procedures operate on binary data. For instance, `pkcs3-import-dh-parameters` expects binary data as input.

Binary data is represented on the Scheme side using bytevectors (see Section "Bytevectors" in *The GNU Guile Reference Manual*). Homogeneous vectors such as SRFI-4 `u8vectors` can also be used[1].

As an example, generating and then exporting Diffie-Hellman parameters in the PEM format can be done as follows:

```
(let* ((dh  (make-dh-parameters 1024))
       (pem (pkcs3-export-dh-parameters dh
                                        x509-certificate-format/pem)))
  (call-with-output-file "some-file.pem"
    (lambda (port)
      (uniform-vector-write pem port))))
```

3.5 Input and Output

The underlying transport of a TLS session can be any Scheme input/output port (see Section "Ports and File Descriptors" in *The GNU Guile Reference Manual*). This has to be specified using `set-session-transport-port!`.

[1] Historically, SRFI-4 `u8vectors` are the closest thing to bytevectors that Guile 1.8 and earlier supported.

However, for better performance, a raw file descriptor can be specified, using `set-session-transport-fd!`. For instance, if the transport layer is a socket port over an OS-provided socket, you can use the `port->fdes` or `fileno` procedure to obtain the underlying file descriptor and pass it to `set-session-transport-fd!` (see Section "Ports and File Descriptors" in *The GNU Guile Reference Manual*). This would work as follows:

```
(let ((socket (socket PF_INET SOCK_STREAM 0))
      (session (make-session connection-end/client)))

  ;;
  ;; Establish a TCP connection...
  ;;

  ;; Use the file descriptor that underlies SOCKET.
  (set-session-transport-fd! session (fileno socket)))
```

Once a TLS session is established, data can be communicated through it (i.e., *via* the TLS record layer) using the port returned by `session-record-port`:

```
(let ((session (make-session connection-end/client)))

  ;;
  ;; Initialize the various parameters of SESSION, set up
  ;; a network connection, etc.
  ;;

  (let ((i/o (session-record-port session)))
    (display "Hello peer!" i/o)
    (let ((greetings (read i/o)))

      ;; ...

      (bye session close-request/rdwr))))
```

Note that each write to the session record port leads to the transmission of an encrypted TLS "Application Data" packet. In the above example, we create an Application Data packet for the 11 bytes for the string that we write. This is not efficient both in terms of CPU usage and bandwidth (each packet adds at least 5 bytes of overhead and can lead to one `write` system call), so we recommend that applications do their own buffering.

A lower-level I/O API is provided by `record-send` and `record-receive!` which take a bytevector (or a SRFI-4 vector) to represent the data sent or received. While it might improve performance, it is much less convenient than the session record port and should rarely be needed.

3.6 Exception Handling

GnuTLS errors are implemented as Scheme exceptions (see Section "Exceptions" in *The GNU Guile Reference Manual*). Each time a GnuTLS function returns an error, an exception with key `gnutls-error` is raised. The additional arguments that are thrown include an error code and the name of the GnuTLS procedure that raised the exception. The error

code is pretty much like an enumerate value: it is one of the `error/` variables exported by the `(gnutls)` module (see Section 3.2 [Enumerates and Constants], page 3). Exceptions can be turned into error messages using the `error->string` procedure.

The following examples illustrates how GnuTLS exceptions can be handled:

```
(let ((session (make-session connection-end/server)))

  ;;
  ;; ...
  ;;

  (catch 'gnutls-error
    (lambda ()
      (handshake session))
    (lambda (key err function . currently-unused)
      (format (current-error-port)
              "a GnuTLS error was raised by `~a': ~a~%"
              function (error->string err)))))
```

Again, error values can be compared using `eq?`:

```
  ;; `gnutls-error' handler.
  (lambda (key err function . currently-unused)
    (if (eq? err error/fatal-alert-received)
        (format (current-error-port)
                "a fatal alert was caught!~%")
        (format (current-error-port)
                "something bad happened: ~a~%"
                (error->string err))))
```

Note that the `catch` handler is currently passed only 3 arguments but future versions might provide it with additional arguments. Thus, it must be prepared to handle more than 3 arguments, as in this example.

4 Guile Examples

This chapter provides examples that illustrate common use cases.

4.1 Anonymous Authentication Guile Example

Anonymous authentication is very easy to use. No certificates are needed by the communicating parties. Yet, it allows them to benefit from end-to-end encryption and integrity checks.

The client-side code would look like this (assuming *some-socket* is bound to an open socket port):

```
;; Client-side.

(let ((client (make-session connection-end/client)))
  ;; Use the default settings.
  (set-session-default-priority! client)

  ;; Don't use certificate-based authentication.
  (set-session-certificate-type-priority! client '())

  ;; Request the "anonymous Diffie-Hellman" key exchange method.
  (set-session-kx-priority! client (list kx/anon-dh))

  ;; Specify the underlying socket.
  (set-session-transport-fd! client (fileno some-socket))

  ;; Create anonymous credentials.
  (set-session-credentials! client
                            (make-anonymous-client-credentials))

  ;; Perform the TLS handshake with the server.
  (handshake client)

  ;; Send data over the TLS record layer.
  (write "hello, world!" (session-record-port client))

  ;; Terminate the TLS session.
  (bye client close-request/rdwr))
```

The corresponding server would look like this (again, assuming *some-socket* is bound to a socket port):

```
;; Server-side.

(let ((server (make-session connection-end/server)))
  (set-session-default-priority! server)
  (set-session-certificate-type-priority! server '())
  (set-session-kx-priority! server (list kx/anon-dh))
```

```
;; Specify the underlying transport socket.
(set-session-transport-fd! server (fileno some-socket))

;; Create anonymous credentials.
(let ((cred (make-anonymous-server-credentials))
      (dh-params (make-dh-parameters 1024)))
  ;; Note: DH parameter generation can take some time.
  (set-anonymous-server-dh-parameters! cred dh-params)
  (set-session-credentials! server cred))

;; Perform the TLS handshake with the client.
(handshake server)

;; Receive data over the TLS record layer.
(let ((message (read (session-record-port server))))
  (format #t "received the following message: ~a~%"
          message)

  (bye server close-request/rdwr)))
```

This is it!

4.2 Using GnuTLS as a cryptography library

The low-level functions in GnuTLS can be accessed for various tasks.

4.2.1 Hash Message Authentication Code

The library provides support for *Hash Message Authentication Code* (*hmac*). This API provides a way to hash a message in a way that is only reproducible with the knowledge of a secret.

This first example demonstrates how to use `hmac-fast` to hash a bytevector in memory.

```
;;; GnuTLS --- Guile bindings for GnuTLS.
;;; Copyright (C) 2023 Free Software Foundation, Inc.
;;;
;;; GnuTLS is free software; you can redistribute it and/or
;;; modify it under the terms of the GNU Lesser General Public
;;; License as published by the Free Software Foundation; either
;;; version 2.1 of the License, or (at your option) any later version.
;;;
;;; GnuTLS is distributed in the hope that it will be useful,
;;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
;;; Lesser General Public License for more details.
;;;
;;; You should have received a copy of the GNU Lesser General Public
;;; License along with GnuTLS; if not, write to the Free Software
```

```
;;; Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA

(use-modules (ice-9 rdelim) (rnrs bytevectors) (gnutls))

(format #t "What is the secret?\n")

(let ((secret (read-line)))
  (format #t "What message do you want to hash?\n")
  (let ((message (read-line)))
    (format #t "The digest is: ~s\n"
            (hmac-direct mac/sha256
                         (string->utf8 secret)
                         (string->utf8 message)))))
```

The next example shows how to hash a whole file that might not fit in memory.

```
;;; GnuTLS --- Guile bindings for GnuTLS.
;;; Copyright (C) 2023 Free Software Foundation, Inc.
;;;
;;; GnuTLS is free software; you can redistribute it and/or
;;; modify it under the terms of the GNU Lesser General Public
;;; License as published by the Free Software Foundation; either
;;; version 2.1 of the License, or (at your option) any later version.
;;;
;;; GnuTLS is distributed in the hope that it will be useful,
;;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
;;; Lesser General Public License for more details.
;;;
;;; You should have received a copy of the GNU Lesser General Public
;;; License along with GnuTLS; if not, write to the Free Software
;;; Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA

(use-modules (ice-9 rdelim)
             (ice-9 binary-ports)
             (rnrs bytevectors)
             (gnutls))

(format #t "What is the secret?\n")

(let ((secret (read-line)))
  (format #t "Which file do you want to hash?\n")
  (let ((file-name (read-line)))
    ;; Create a new state that will be reused when new bytes are
    ;; available.
    (let ((state (make-hmac mac/sha256 (string->utf8 secret))))
      (call-with-input-file file-name
        (lambda (port)
```

```
          (let hash-all ()
            ;; Read raw bytes from the file.
            (let ((next (get-bytevector-some port)))
              (if (eof-object? next)
                  ;; No more data in the file
                  (format #t "The digest is: ~s\n"
                          (hmac-output state))
                  (begin
                    ;; Hash the bytes we got, and continue.
                    (hmac! state next)
                    (hash-all))))))
        #:binary #t))))
```

The final example shows how you can re-use a state to continue hashing different inputs. This requires the `hmac-copy` function, which is not always available (see Section 3.1 [Living on the cutting edge], page 3).

```
;;; GnuTLS --- Guile bindings for GnuTLS.
;;; Copyright (C) 2023 Free Software Foundation, Inc.
;;;
;;; GnuTLS is free software; you can redistribute it and/or
;;; modify it under the terms of the GNU Lesser General Public
;;; License as published by the Free Software Foundation; either
;;; version 2.1 of the License, or (at your option) any later version.
;;;
;;; GnuTLS is distributed in the hope that it will be useful,
;;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
;;; Lesser General Public License for more details.
;;;
;;; You should have received a copy of the GNU Lesser General Public
;;; License along with GnuTLS; if not, write to the Free Software
;;; Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA

(use-modules (rnrs bytevectors) (gnutls))

(when (defined? 'hmac-copy)

  (let ((hash-with-prefix
         (lambda (secret prefix)
           ;; Return a hasher of a string as a 1-argument function,
           ;; by first adding a prefix to it.
           (let ((tag (make-prompt-tag)))
             (call-with-prompt
              tag
              (lambda ()
                (let ((state (make-hmac mac/sha256 secret)))
                  (hmac! state prefix)
```

```
                (let ((line (abort-to-prompt tag)))
                  ;; The flow may reenter multiple times here, so
                  ;; we have to copy the hmac state.
                  (let ((copy (hmac-copy state)))
                    (hmac! copy line)
                    (hmac-output copy)))))
            (lambda (k) k))))))

;; So if "Prefix " is the prefix, it will be hashed only once.
(let ((expected-output-1
       (hmac-direct mac/sha256
                    (string->utf8 "secret!")
                    (string->utf8 "Prefix and then some")))
      (expected-output-2
       (hmac-direct mac/sha256
                    (string->utf8 "secret!")
                    (string->utf8 "Prefix and other data"))))
  ;; hasher is a 1-argument function that computes the hash of
  ;; "Prefix " + its argument (as bytevectors), but re-uses the
  ;; state it has after hashing "Prefix ".
  (let ((hasher (hash-with-prefix (string->utf8 "secret!")
                                  (string->utf8 "Prefix "))))
    (let ((output-1 (hasher (string->utf8 "and then some")))
          (output-2 (hasher (string->utf8 "and other data"))))
      (unless (and (equal? output-1 expected-output-1)
                   (equal? output-2 expected-output-2))
        (error "This cannot happen."))))))
```

4.2.2 Hash Digest Algorithms

The API for hash algorithm is similar to that for hmac, except that there is no secret data to reproduce the hash. So for instance, the second hmac example becomes (see Section 4.2.1 [Hash Message Authentication Code], page 8):

```
;;; GnuTLS --- Guile bindings for GnuTLS.
;;; Copyright (C) 2023 Free Software Foundation, Inc.
;;;
;;; GnuTLS is free software; you can redistribute it and/or
;;; modify it under the terms of the GNU Lesser General Public
;;; License as published by the Free Software Foundation; either
;;; version 2.1 of the License, or (at your option) any later version.
;;;
;;; GnuTLS is distributed in the hope that it will be useful,
;;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
;;; Lesser General Public License for more details.
;;;
;;; You should have received a copy of the GNU Lesser General Public
```

```
;;; License along with GnuTLS; if not, write to the Free Software
;;; Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA

(use-modules (ice-9 rdelim)
             (ice-9 binary-ports)
             (rnrs bytevectors)
             (gnutls))
(format #t "Which file do you want to hash?\n")

(let ((file-name (read-line)))
  ;; Create a new state that will be reused when new bytes are
  ;; available.
  (let ((state (make-hash digest/sha256)))
    (call-with-input-file file-name
      (lambda (port)
        (let hash-all ()
          ;; Read raw bytes from the file.
          (let ((next (get-bytevector-some port)))
            (if (eof-object? next)
                ;; No more data in the file
                (format #t "The digest is: ~s\n"
                        (hash-output state))
                (begin
                  ;; Hash the bytes we got, and continue.
                  (hash! state next)
                  (hash-all))))))
      #:binary #t)))
```

4.2.3 Authenticated Encryption

The goal of authenticated encryption is to make sure that the data has been encrypted by a party that knows a shared secret. The encryption and decryption procedures are very similar. Both parties must know a shared secret key and a nonce. The nonce is a value that must only be used once to encrypt data. The nonce may be as long as you want, but the secret key must be the exact size expected by the cipher algorithm.

The API can also use extra authentication data, that can change on a packet-by-packet basis, whatever your definition for a packet is.

To use the API, the cipher algorithm must be compatible with AEAD. When it is, it defines a default tag size. You can override the tag size, or use the 0 value to use the default tag size.

This example encrypts a file:

```
;;; GnuTLS --- Guile bindings for GnuTLS.
;;; Copyright (C) 2023 Free Software Foundation, Inc.
;;;
;;; GnuTLS is free software; you can redistribute it and/or
;;; modify it under the terms of the GNU Lesser General Public
;;; License as published by the Free Software Foundation; either
```

```
;;; version 2.1 of the License, or (at your option) any later version.
;;;
;;; GnuTLS is distributed in the hope that it will be useful,
;;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
;;; Lesser General Public License for more details.
;;;
;;; You should have received a copy of the GNU Lesser General Public
;;; License along with GnuTLS; if not, write to the Free Software
;;; Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA

(use-modules (ice-9 rdelim)
             (ice-9 binary-ports)
             (rnrs bytevectors)
             (gnutls))

(format #t "What is the secret?\n")

(let ((secret (read-line)))
  (set! secret
        (string->utf8 secret))
  (unless (equal? (bytevector-length secret)
                  (cipher-key-size cipher/aes-256-gcm))
    (error "incorrect key length"))
  (format #t "Which file do you want to encrypt?\n")
  (let ((file-name (read-line)))
    ;; Create a new state that will be reused when new bytes are
    ;; available.
    (let ((cipher (make-aead-cipher cipher/aes-256-gcm secret)))
      (call-with-output-file (string-append file-name ".encrypted~")
        (lambda (out)
          (call-with-input-file file-name
            (lambda (in)
              ;; Read raw bytes from the file.
              (let do-encrypt ()
                (let ((next (get-bytevector-some in)))
                  (unless (eof-object? next)
                    (let ((encrypted
                           (aead-cipher-encrypt
                            cipher
                            ;; Do not re-use the same nonce twice. The nonce
                            ;; size is constrained; for aes256/GCM, this is 12
                            ;; bytes.
                            (string->utf8 "12 randbytes")
                            (string->utf8 "Additional secret data")
                            0
                            next)))
```

```
                        (put-bytevector out encrypted)
                        (do-encrypt))))))
              #:binary #t))
          #:binary #t))
      (rename-file (string-append file-name ".encrypted~")
                   (string-append file-name ".encrypted"))))
```

And this example decrypts the same file. Please note that the decrypted file is written to disk, which is acceptable for this simple example:

```
;;; GnuTLS --- Guile bindings for GnuTLS.
;;; Copyright (C) 2023 Free Software Foundation, Inc.
;;;
;;; GnuTLS is free software; you can redistribute it and/or
;;; modify it under the terms of the GNU Lesser General Public
;;; License as published by the Free Software Foundation; either
;;; version 2.1 of the License, or (at your option) any later version.
;;;
;;; GnuTLS is distributed in the hope that it will be useful,
;;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
;;; Lesser General Public License for more details.
;;;
;;; You should have received a copy of the GNU Lesser General Public
;;; License along with GnuTLS; if not, write to the Free Software
;;; Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA

(use-modules (ice-9 rdelim)
             (ice-9 binary-ports)
             (rnrs bytevectors)
             (gnutls))

(format #t "What is the secret?\n")

(let ((secret (read-line)))
  (set! secret
        (string->utf8 secret))
  (unless (equal? (bytevector-length secret)
                  (cipher-key-size cipher/aes-256-gcm))
    (error "incorrect key length"))
  (format #t "Which file do you want to decrypt?\n")
  (let ((file-name (read-line)))
    ;; Create a new state that will be reused when new bytes are
    ;; available.
    (let ((cipher (make-aead-cipher cipher/aes-256-gcm secret)))
      (call-with-output-file (string-append file-name ".decrypted~")
        (lambda (out)
          (call-with-input-file file-name
```

```
            (lambda (in)
              ;; Read raw bytes from the file.
              (let do-decrypt ()
                (let ((next (get-bytevector-some in)))
                  (unless (eof-object? next)
                    (let ((decrypted
                           (aead-cipher-decrypt
                            cipher
                            ;; The same value as used at encryption time:
                            (string->utf8 "12 randbytes")
                            (string->utf8 "Additional secret data")
                            0
                            next)))
                      (put-bytevector out decrypted)
                      (do-decrypt))))))
            #:binary #t))
        #:binary #t))
    (rename-file (string-append file-name ".decrypted~")
                 (string-append file-name ".decrypted"))))
```

4.2.4 Low-lovel encryption API

In some cases, you may want to use a lower-level encryption API. In this example, the data to encrypt spans an integer number of blocks. You need to specify the initialization vector, to seed the encryption, and a private key.

```
;;; GnuTLS --- Guile bindings for GnuTLS.
;;; Copyright (C) 2023 Free Software Foundation, Inc.
;;;
;;; GnuTLS is free software; you can redistribute it and/or
;;; modify it under the terms of the GNU Lesser General Public
;;; License as published by the Free Software Foundation; either
;;; version 2.1 of the License, or (at your option) any later version.
;;;
;;; GnuTLS is distributed in the hope that it will be useful,
;;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
;;; Lesser General Public License for more details.
;;;
;;; You should have received a copy of the GNU Lesser General Public
;;; License along with GnuTLS; if not, write to the Free Software
;;; Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA

(use-modules (ice-9 rdelim)
             (ice-9 binary-ports)
             (rnrs bytevectors)
             (gnutls))
```

```
;; To define a symmetric encryption cipher context, you need an algorithm, a
;; key, and an initialization vector.

(define algorithm cipher/aes-128-cbc)

(define cipher
  (let ((initialisation-vector
         (string->utf8 "Initialisation.."))
        (key
         (string->utf8 "The 16-byte key.")))
    (unless (eqv? (bytevector-length initialisation-vector)
                  (cipher-iv-size algorithm))
      (error "Incorrect initialization vector size."))
    (unless (eqv? (bytevector-length key)
                  (cipher-key-size algorithm))
      (error "Incorrect key size."))
    (make-cipher algorithm key initialisation-vector)))

;; The context may be used to encrypt and decrypt data, if the data spans an
;; integer number of blocks.

(define block-size
  (cipher-block-size (cipher-algorithm cipher)))

(define data
  (string->utf8 "The data to encrypt must be a bytevector \
whose length must be a multiple of the block size. If you \
want to use the low-level cipher API, you must manage the \
data padding yourself, and know the message length."))

(define encrypted
  (cipher-encrypt cipher data))

(define decrypted
  (cipher-decrypt cipher encrypted))

(unless (equal? data decrypted)
  (error "data decryption failed."))
```

4.2.5 Public key cryptography

The following example shows how to use elliptic curve cryptography with a private key.

```
;;; GnuTLS --- Guile bindings for GnuTLS.
;;; Copyright (C) 2023 Free Software Foundation, Inc.
;;;
;;; GnuTLS is free software; you can redistribute it and/or
;;; modify it under the terms of the GNU Lesser General Public
```

```
;;; License as published by the Free Software Foundation; either
;;; version 2.1 of the License, or (at your option) any later version.
;;;
;;; GnuTLS is distributed in the hope that it will be useful,
;;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
;;; Lesser General Public License for more details.
;;;
;;; You should have received a copy of the GNU Lesser General Public
;;; License along with GnuTLS; if not, write to the Free Software
;;; Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA

(use-modules (ice-9 rdelim)
             (ice-9 binary-ports)
             (ice-9 match)
             (rnrs bytevectors)
             (gnutls))

(define (read-curve)
  (format #t "curve:\n")
  (string->ecc-curve (read-line)))

(define (read-parameter name)
  (format #t "~a:\n" name)
  (base64-decode (read-line)))

(define (read-parameters)
  (let* ((curve (read-curve))
         (x (read-parameter 'x))
         (y (read-parameter 'y))
         (k (read-parameter 'k)))
    (values curve x y k)))

(define private-key
  (receive (curve x y k) (read-parameters)
    (let ((key (import-raw-ecc-private-key curve x y k)))
      key)))

(define message (string->utf8 "Hello, world!"))

(define signature
  (private-key-sign-data private-key
                         sign-algorithm/ecdsa-secp521r1-sha512
                         message
                         '()))

(define public-key
```

```
  (let ((key (private-key->public-key private-key
                                      (list key-usage/digital-signature))))
    key))

(public-key-verify-data public-key sign-algorithm/ecdsa-secp521r1-sha512
                        message signature)

(format #t "I could sign a message with that key.\n")
```

The next example shows how to generate a private key.

```
;;; GnuTLS --- Guile bindings for GnuTLS.
;;; Copyright (C) 2023 Free Software Foundation, Inc.
;;;
;;; GnuTLS is free software; you can redistribute it and/or
;;; modify it under the terms of the GNU Lesser General Public
;;; License as published by the Free Software Foundation; either
;;; version 2.1 of the License, or (at your option) any later version.
;;;
;;; GnuTLS is distributed in the hope that it will be useful,
;;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
;;; Lesser General Public License for more details.
;;;
;;; You should have received a copy of the GNU Lesser General Public
;;; License along with GnuTLS; if not, write to the Free Software
;;; Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA

(use-modules (ice-9 receive)
             (rnrs bytevectors)
             (gnutls))

(receive (curve x y k)
    (let ((key (generate-private-key
                pk-algorithm/ecc ecc-curve/secp521r1)))
      (private-key-export-raw-ecc key))
  (write
   `((curve . ,(ecc-curve->string curve))
     (x . ,(base64-encode x))
     (y . ,(base64-encode y))
     (k . ,(base64-encode k))))
  (newline))
```

In addition, abstract public or private keys can be obtained from X509 certificate and private key, with `x509-certificate->public-key` and `x509-private-key->private-key`.

4.2.6 Generating random numbers

Gnutls lets you generate different kinds of pseudo-random numbers, depending on the implications if it is guessed. Here is how you generate a random number with the lowest security requirements:

```
;;; GnuTLS --- Guile bindings for GnuTLS.
;;; Copyright (C) 2023 Free Software Foundation, Inc.
;;;
;;; GnuTLS is free software; you can redistribute it and/or
;;; modify it under the terms of the GNU Lesser General Public
;;; License as published by the Free Software Foundation; either
;;; version 2.1 of the License, or (at your option) any later version.
;;;
;;; GnuTLS is distributed in the hope that it will be useful,
;;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
;;; Lesser General Public License for more details.
;;;
;;; You should have received a copy of the GNU Lesser General Public
;;; License along with GnuTLS; if not, write to the Free Software
;;; Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA

(use-modules (rnrs bytevectors)
             (gnutls))

(define random-data
  (gnutls-random
   ;; Choose a security level: /nonce, /random or /key.
   random-level/nonce
   ;; Choose the number of bytes:
   4))

(let ((dice-roll
       (remainder
        (car (bytevector->uint-list random-data (endianness little) 4))
        6)))
  (format #t "You roll a ~a.\n" (+ dice-roll 1)))
```

4.2.7 Encoding binary data

When working with gnutls, you may come across a lot of binary data, in the form of guile bytevectors. Base16 and base64 are popular encoding schemes for binary data.

This example shows how to encode and decode binary data to and from base16.

```
;;; GnuTLS --- Guile bindings for GnuTLS.
;;; Copyright (C) 2023 Free Software Foundation, Inc.
;;;
;;; GnuTLS is free software; you can redistribute it and/or
;;; modify it under the terms of the GNU Lesser General Public
```

```
;;; License as published by the Free Software Foundation; either
;;; version 2.1 of the License, or (at your option) any later version.
;;;
;;; GnuTLS is distributed in the hope that it will be useful,
;;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
;;; Lesser General Public License for more details.
;;;
;;; You should have received a copy of the GNU Lesser General Public
;;; License along with GnuTLS; if not, write to the Free Software
;;; Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA

(use-modules (rnrs bytevectors)
             (gnutls))

(define data
  (string->utf8 "Hello, world!"))

(define encoded
  (hex-encode data))

(define decoded
  (hex-decode encoded))

(format #t "The base16 encoding is: ~s\n"
        encoded)

(format #t "Decoding it back gives: ~s\n"
        (utf8->string decoded))
```

This example shows how to encode and decode binary data to and from base64.

```
;;; GnuTLS --- Guile bindings for GnuTLS.
;;; Copyright (C) 2023 Free Software Foundation, Inc.
;;;
;;; GnuTLS is free software; you can redistribute it and/or
;;; modify it under the terms of the GNU Lesser General Public
;;; License as published by the Free Software Foundation; either
;;; version 2.1 of the License, or (at your option) any later version.
;;;
;;; GnuTLS is distributed in the hope that it will be useful,
;;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
;;; Lesser General Public License for more details.
;;;
;;; You should have received a copy of the GNU Lesser General Public
;;; License along with GnuTLS; if not, write to the Free Software
;;; Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
```

```
(use-modules (rnrs bytevectors)
             (gnutls))

(define data
  (string->utf8 "Hello, world!"))

(define encoded
  (base64-encode data))

(define decoded
  (base64-decode encoded))

(format #t "The base64 encoding is: ~s\n"
        encoded)

(format #t "Decoding it back gives: ~s\n"
        (utf8->string decoded))
```

Unfortunately, gnutls does not provide an API to encode data to the popular base64-url encoding. However, it is possible to convert from base64 to base64-url and back.

```
;;; GnuTLS --- Guile bindings for GnuTLS.
;;; Copyright (C) 2023 Free Software Foundation, Inc.
;;;
;;; GnuTLS is free software; you can redistribute it and/or
;;; modify it under the terms of the GNU Lesser General Public
;;; License as published by the Free Software Foundation; either
;;; version 2.1 of the License, or (at your option) any later version.
;;;
;;; GnuTLS is distributed in the hope that it will be useful,
;;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
;;; Lesser General Public License for more details.
;;;
;;; You should have received a copy of the GNU Lesser General Public
;;; License along with GnuTLS; if not, write to the Free Software
;;; Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA

(use-modules (rnrs bytevectors)
             (gnutls))

(define (base64->base64-url str)
  ;; Replace '+' with '-', '/' with '_', and remove the '=' padding
  ;; characters.
  (string-filter
   (lambda (c)
     (not (eqv? c #\=)))
```

```
   (string-map
    (lambda (c)
      (case c
        ((#\+) #\-)
        ((#\/) #\_)
        (else c)))
    str)))

(define (base64-url->base64 str)
  ;; Replace '-' with '+', '_' with '/', and add padding characters.
  (string-append
   (string-map
    (lambda (c)
      (case c
        ((#\-) #\+)
        ((#\_) #\/)
        (else c)))
    str)
   (case (remainder (string-length str) 4)
     ((2) "==")
     ((3) "=")
     (else ""))))

(define data
  (string->utf8 "~~ Hello, world! ~~"))

(define encoded
  (base64->base64-url (base64-encode data)))

(define decoded
  (base64-decode (base64-url->base64 encoded)))

(format #t "The base64-url encoding is: ~s\n"
        encoded)

(format #t "Decoding it back gives: ~s\n"
        (utf8->string decoded))
```

5 Guile Reference

This chapter lists the Scheme procedures exported by the `(gnutls)` module (see Section "Using the Guile Module System" in *The GNU Guile Reference Manual*).

`gnutls-random` *level length* [Scheme Procedure]
Return a random vector of *length* bytes.

`x509-private-key->private-key` *privkey flags* [Scheme Procedure]
Convert the X509 private key, *privkey*, to an abstract private key.

`x509-certificate->public-key` *crt* [Scheme Procedure]
Convert the X509 certificate, *crt*, to an abstract public key.

`public-key-verify-hash` *key algo hash-data signature* [Scheme Procedure]
Verify the hash data *signature*.

`public-key-verify-data` *key algo data signature* [Scheme Procedure]
Verify the data *signature*.

`public-key-encrypt-data` *key data* [Scheme Procedure]
Encrypt the *data*.

`private-key-decrypt-data` *key data* [Scheme Procedure]
Decrypt the *data*.

`private-key-sign-hash` *key algo hash-data flags* [Scheme Procedure]
Sign the *hash_data* and return the signature. *flags* is a list of privkey flags. Available flags are: `privkey/sign-flag-tls1-rsa privkey/sign-flag-rsa-pss flag-reproducible`.

`private-key-sign-data` *key algo data flags* [Scheme Procedure]
Sign the *data* and return the signature. *flags* is a list of privkey flags.Available flags are: `privkey/sign-flag-tls1-rsa privkey/sign-flag-rsa-pss privkey/flag-reproducible`.

`generate-private-key` *algo bits-or-curve* [Scheme Procedure]
Return a new private key.

`public-key-preferred-hash-algorithm` *key* [Scheme Procedure]
Return the preferred hash algorithm for *key*, and a boolean indicating whether this algorithm is mandatory.

`private-key-pk-algorithm` *key* [Scheme Procedure]
Return the private key algorithm used by *key* and the number of bits.

`public-key-pk-algorithm` *key* [Scheme Procedure]
Return the public key algorithm used by *key* and the number of bits.

`public-key-export` *key format* [Scheme Procedure]
Export a public key to PEM or DER.

`private-key->public-key` *key usage* [Scheme Procedure]
Return the public part of *key*. *usage* is a list of key usage flags, such as `key-usage/digital-signature`.

`private-key-export-raw-rsa` *key* [Scheme Procedure]
Export a RSA private key, and return 8 parameters: M, E, D, P, Q, U, E1, E2.

`private-key-export-raw-ecc` *key* [Scheme Procedure]
Export a ECC private key, and return 4 parameters: the curve, X, Y and K.

`private-key-export-raw-dsa` *key* [Scheme Procedure]
Export a DSA private key, and return 5 parameters: P, Q, G, Y and X.

`public-key-export-raw-rsa` *key* [Scheme Procedure]
Export a RSA public key, and return 2 parameters: M and E.

`public-key-export-raw-ecc` *key* [Scheme Procedure]
Export a ECC public key, and return 3 parameters: the curve, X and Y.

`public-key-export-raw-dsa` *key* [Scheme Procedure]
Export a DSA public key, and return 4 parameters: P, Q, G and Y.

`import-raw-rsa-private-key` *m e d p q u e1 e2* [Scheme Procedure]
Create a new RSA private key. *d* starting at 3.7.0, and *u*, *e1* and *e2* are optional, pass `#f` to not set them.

`import-raw-ecc-private-key` *curve x y k* [Scheme Procedure]
Create a new ECC private key.

`import-raw-dsa-private-key` *p q g y x* [Scheme Procedure]
Create a new DSA private key. Starting at 3.7.0, the *y* parameter is optional, pass `#f` if unknown.

`import-raw-rsa-public-key` *m e* [Scheme Procedure]
Create a new RSA public key.

`import-raw-ecc-public-key` *curve x y* [Scheme Procedure]
Create a new ECC public key.

`import-raw-dsa-public-key` *p q g y* [Scheme Procedure]
Create a new DSA public key.

`hex-decode` *data* [Scheme Procedure]
Try and decode *data* from base16, return it as a bytevector.

`base64-decode` *data* [Scheme Procedure]
Try and decode *data*, return it as a bytevector.

`base64-encode` *data* [Scheme Procedure]
Return as an ASCII string the base64 encoding of *data*.

`hex-encode` *data* [Scheme Procedure]
Return as an ASCII string the base16 encoding of *data*.

`ecc-curve-size` *curve* [Scheme Procedure]
Return the size of *curve*, in bytes (0 on failure).

`sign-algorithm-is-secure?` *sign for-certs* [Scheme Procedure]
Check whether the *sign* algorithm is considered safe. *for-certs?* is `#t` if the security is for signing a certificate, or `#f` for other data.

`sign-algorithm-supports?` *sign pk* [Scheme Procedure]
Check whether the *sign* algorithm can be used with the *pk* public-key algorithm.

`ecc-curve->pk-algorithm` *curve* [Scheme Procedure]
Return the public key algorithm that can be used with *curve*.

`sign-algorithm->pk-algorithm` *sign* [Scheme Procedure]
Return a public key algorithm that can sign data with the *sign* algorithm.

`oid->ecc-curve` *oid* [Scheme Procedure]
Return the ECC curve identified by *oid*.

`oid->sign-algorithm` *oid* [Scheme Procedure]
Return the sign algorithm identified by *oid*.

`oid->pk-algorithm` *oid* [Scheme Procedure]
Return the public key algorithm identified by *oid*.

`sign-algorithm->digest-algorithm` *sign* [Scheme Procedure]
Return the digest algorithm used for the *sign* algorithm.

`pk-algorithm->sign-algorithm` *pk digest* [Scheme Procedure]
Return the signature algorithm compatible with the *pk* public-key algorithm and the *digest* algorithm.

`ecc-curve-list` [Scheme Procedure]
Return the list of ECC curves. **This function is not thread-safe.**

`sign-algorithm-list` [Scheme Procedure]
Return the list of public key algorithms. **This function is not thread-safe.**

`pk-algorithm-list` [Scheme Procedure]
Return the list of public key algorithms. **This function is not thread-safe.**

`ecc-curve->oid` *curve* [Scheme Procedure]
Return the OID allocated to *curve*.

`sign-algorithm->oid` *algo* [Scheme Procedure]
Return the OID allocated to *algo*.

`pk-algorithm->oid` *algorithm* [Scheme Procedure]
Return the OID associated to *algorithm*.

`string->ecc-curve` *id* [Scheme Procedure]
Return ECC curve identified by *id*.

`string->sign-algorithm` *id* [Scheme Procedure]
Return the signature algorithm identified by *id*.

`string->pk-algorithm` *id* [Scheme Procedure]
Return the public key algorithm identified by *id*.

`cipher-algorithm` *handle* [Scheme Procedure]
Return the underlying cipher algorithm.

`cipher-tag` *handle tagsize* [Scheme Procedure]
Read a tag.

`cipher-add-auth!` *handle data* [Scheme Procedure]
Add authentication *data*.

`cipher-set-iv!` *handle data* [Scheme Procedure]
Set the IV *data*.

`cipher-decrypt` *handle data* [Scheme Procedure]
Decrypt the *data*.

`cipher-encrypt` *handle data* [Scheme Procedure]
Encrypt the *data*.

`make-cipher` *algorithm key iv* [Scheme Procedure]
Return a new cipher context, using the cipher *algorithm*.

`aead-cipher-algorithm` *handle* [Scheme Procedure]
Return the underlying AEAD cipher algorithm.

`aead-cipher-decrypt` *handle nonce auth tagsize data* [Scheme Procedure]
Decrypt the *data*, checking that the authentication data *auth* is correct. Pass 0 as *tagsize* to use the default tag size for the underlying algorithm.

`aead-cipher-encrypt` *handle nonce auth tagsize data* [Scheme Procedure]
Encrypt the *data*, with additional *auth* data. Use 0 for *tagsize* to use the default tag size for the algorithm.

`make-aead-cipher` *algorithm key* [Scheme Procedure]
Return a new AEAD cipher context, using the AEAD *algorithm*, and with *key* (a bytevector) as the secret.

`cipher-iv-size` *algorithm* [Scheme Procedure]
Return the length of the initialization vector for *algorithm*.

`cipher-block-size` *algorithm* [Scheme Procedure]
Return the required block size for *algorithm*.

`cipher-key-size` *algorithm* [Scheme Procedure]
Return the required key size for *algorithm*.

`cipher-tag-size` *algorithm* [Scheme Procedure]
Return the default tag size for *algorithm*, or 0 if this is not an AEAD algorithm.

`hash-output` *hash* [Scheme Procedure]
Return the digest of the current *hash* state.

`hash-length` *algorithm* [Scheme Procedure]
Return the length of the *algorithm* digest output, or 0 if unavailable.

`hash-algorithm` *hash* [Scheme Procedure]
Return the algorithm that *hash* has been built for.

`hash!` *hash text* [Scheme Procedure]
Hash the *text* bytes in the *hash* state.

`make-hash` *algorithm* [Scheme Procedure]
Start a hash operation according to *algorithm*.

`hash-direct` *algorithm text* [Scheme Procedure]
Hash *text* according to *algorithm*. Return the digest as a bytevector.

`mac-nonce-size` *algorithm* [Scheme Procedure]
Return the length of the nonce for *algorithm*, or 0 if unavailable.

`set-hmac-nonce!` *hmac nonce* [Scheme Procedure]
Set *nonce* in the *hmac* state.

`hmac-output` *hmac* [Scheme Procedure]
Return the digest of the current *hmac* state.

`hmac-length` *algorithm* [Scheme Procedure]
Return the length of the *algorithm* HMAC output, or 0 if unavailable.

`hmac-algorithm` *hmac* [Scheme Procedure]
Return the algorithm that *hmac* has been built for.

`hmac!` *hmac text* [Scheme Procedure]
Hash the *text* bytes in the *hmac* state.

`make-hmac` *algorithm key* [Scheme Procedure]
Return a new hmac object that can be fed further input to hash. Use the given MAC or HMAC *algorithm*, and use *key* (a bytevector) as the secret.

`hmac-direct` *algorithm key text* [Scheme Procedure]
Hash *text* with *algorithm*, and the secret *key*. It will not work if *algorithm* requires a nonce, such as UMAC or GMAC. Both *key* and *text* must be bytevectors.

`set-log-level!` *level* [Scheme Procedure]
Enable GnuTLS logging up to *level* (an integer).

`set-log-procedure!` *proc* [Scheme Procedure]
Use *proc* (a two-argument procedure) as the global GnuTLS log procedure.

`%set-certificate-credentials-openpgp-keys!` *cred pub sec* [Scheme Procedure]
Use certificate *pub* and secret key *sec* in certificate credentials *cred*.

`%openpgp-keyring-contains-key-id?` *keyring id* [Scheme Procedure]
Return `#f` if key ID *id* is in *keyring*, `#f` otherwise.

`import-openpgp-keyring` *data format* [Scheme Procedure]
Import *data* (a u8vector) according to *format* and return the imported keyring.

`%openpgp-certificate-usage` *key* [Scheme Procedure]
Return a list of values denoting the key usage of *key*.

`%openpgp-certificate-version` *key* [Scheme Procedure]
Return the version of the OpenPGP message format (RFC2440) honored by *key*.

`%openpgp-certificate-algorithm` *key* [Scheme Procedure]
Return two values: the certificate algorithm used by *key* and the number of bits used.

`%openpgp-certificate-names` *key* [Scheme Procedure]
Return the list of names for *key*.

`%openpgp-certificate-name` *key index* [Scheme Procedure]
Return the *index*th name of *key*.

`%openpgp-certificate-fingerprint` *key* [Scheme Procedure]
Return a new u8vector denoting the fingerprint of *key*.

`%openpgp-certificate-fingerprint!` *key fpr* [Scheme Procedure]
Store in *fpr* (a u8vector) the fingerprint of *key*. Return the number of bytes stored in *fpr*.

`%openpgp-certificate-id!` *key id* [Scheme Procedure]
Store the ID (an 8 byte sequence) of certificate *key* in *id* (a u8vector).

`%openpgp-certificate-id` *key* [Scheme Procedure]
Return the ID (an 8-element u8vector) of certificate *key*.

`%import-openpgp-private-key` *data format* [*pass*] [Scheme Procedure]
Return a new OpenPGP private key object resulting from the import of *data* (a uniform array) according to *format*. Optionally, a passphrase may be provided.

`%import-openpgp-certificate` *data format* [Scheme Procedure]
Return a new OpenPGP certificate object resulting from the import of *data* (a uniform array) according to *format*.

`set-x509-certificate-serial!` *cert serial* [Scheme Procedure]
Set the serial number of *cert* to the bytevector *serial*.

`x509-certificate-serial` *cert* [Scheme Procedure]
Return the serial number of *cert*.

`set-x509-certificate-ca-status!` *cert status* [Scheme Procedure]
Set the CA status flag of *cert* to *status*, either `#t` or `#f`.

`x509-certificate-ca-status` *cert* [Scheme Procedure]
Return the CA status of *cert*.

`set-x509-certificate-expiration-time!` *cert time* [Scheme Procedure]
Set the expiration time of *cert* to *time*.

`x509-certificate-expiration-time` *cert* [Scheme Procedure]
Return the expiration time of *cert*.

`set-x509-certificate-activation-time!` *cert time* [Scheme Procedure]
Set the activation time of *cert* to *time*.

`x509-certificate-activation-time` *cert* [Scheme Procedure]
Return the activation time of *cert*.

`set-x509-certificate-key!` *cert key* [Scheme Procedure]
Set the public parameters of *cert* using the private key *key*.

`set-x509-certificate-dn-by-oid!` *cert oid name* [Scheme Procedure]
Set the part of the name of the certificate request subject for *cert* corresponding to *oid* to the string *name*.

`sign-x509-certificate!` *cert issuer key* [Scheme Procedure]
Sign *cert* using *cert*, also a certificate, and *key*, the issuer's private key.

`x509-certificate-fingerprint` *cert algo* [Scheme Procedure]
Return the fingerprint (a u8vector) of the certificate *cert*, computed using the digest algorithm *algo*.

`x509-certificate-subject-alternative-name` *cert index* [Scheme Procedure]
Return two values: the alternative name type for *cert* (i.e., one of the `x509-subject-alternative-name/` values) and the actual subject alternative name (a string) at *index*. Both values are `#f` if no alternative name is available at *index*.

`set-x509-certificate-subject-key-id!` *cert id* [Scheme Procedure]
Set the subject key ID for *cert* to the bytevector *id*.

`x509-certificate-subject-key-id` *cert* [Scheme Procedure]
Return the subject key ID (a u8vector) for *cert*.

`x509-certificate-authority-key-id` *cert* [Scheme Procedure]
Return the key ID (a u8vector) of the X.509 certificate authority of *cert*.

`x509-certificate-key-id` *cert* [Scheme Procedure]
Return a statistically unique ID (a u8vector) for *cert* that depends on its public key parameters. This is normally a 20-byte SHA-1 hash.

`set-x509-certificate-version!` *cert version* [Scheme Procedure]
Set the version of *cert* to *version*.

`x509-certificate-version` *cert* [Scheme Procedure]
Return the version of *cert*.

`set-x509-certificate-key-usage!` *cert flags* [Scheme Procedure]
Set the key_usage of *cert* to *flags*, a list of usage flags.

`x509-certificate-key-usage` *cert* [Scheme Procedure]
Return the key usage of *cert* (i.e., a list of `key-usage/` values), or the empty list if *cert* does not contain such information.

`x509-certificate-public-key-algorithm` *cert* [Scheme Procedure]
Return two values: the public key algorithm (i.e., one of the `pk-algorithm/` values) of *cert* and the number of bits used.

`x509-certificate-signature-algorithm` *cert* [Scheme Procedure]
Return the signature algorithm used by *cert* (i.e., one of the `sign-algorithm/` values).

`x509-certificate-matches-hostname?` *cert hostname* [Scheme Procedure]
Return true if *cert* matches *hostname*, a string denoting a DNS host name. This is the basic implementation of RFC 2818 (`https://tools.ietf.org/html/rfc2818`) (aka. HTTPS).

`x509-certificate-issuer-dn-oid` *cert index* [Scheme Procedure]
Return the OID (a string) at *index* from *cert*'s issuer DN. Return `#f` if no OID is available at *index*.

`x509-certificate-dn-oid` *cert index* [Scheme Procedure]
Return OID (a string) at *index* from *cert*. Return `#f` if no OID is available at *index*.

`x509-certificate-issuer-dn` *cert* [Scheme Procedure]
Return the distinguished name (DN) of X.509 certificate *cert*.

`x509-certificate-dn` *cert* [Scheme Procedure]
Return the distinguished name (DN) of X.509 certificate *cert*. The form of the DN is as described in RFC 2253 (`https://tools.ietf.org/html/rfc2253`).

`pkcs8-import-x509-private-key` *data format* [*pass* [*encrypted*]] [Scheme Procedure]
Return a new X.509 private key object resulting from the import of *data* (a uniform array) according to *format*. Optionally, if *pass* is not `#f`, it should be a string denoting a passphrase. *encrypted* tells whether the private key is encrypted (`#t` by default).

`export-x509-private-key` *key format* [Scheme Procedure]
Return a bytevector resulting from the export of *key* (an X.509 private key) according to *format*.

`import-x509-private-key` *data format* [Scheme Procedure]
Return a new X.509 private key object resulting from the import of *data* (a uniform array) according to *format*.

`generate-x509-private-key` *algorithm bits flags* [Scheme Procedure]
Return a new X.509 private key object of size *bits* generated using *algorithm*, a pk-algorithm enum value, and*flags*, a list of privkey enum values.

`export-x509-certificate` *cert format* [Scheme Procedure]
Return a bytevector resulting from the export of *cert* (an X.509 certificate) according to *format*.

`import-x509-certificate` *data format* [Scheme Procedure]
Return a new X.509 certificate object resulting from the import of *data* (a uniform array) according to *format*.

`make-x509-certificate` [Scheme Procedure]
Return a new, empty X.509 certificate object.

`server-session-psk-username` *session* [Scheme Procedure]
Return the username associated with PSK server session *session*.

`set-psk-client-credentials!` *cred username key key-format* [Scheme Procedure]
Set the client credentials for *cred*, a PSK client credentials object.

`make-psk-client-credentials` [Scheme Procedure]
Return a new PSK client credentials object.

`set-psk-server-credentials-file!` *cred file* [Scheme Procedure]
Use *file* as the password file for PSK server credentials *cred*.

`make-psk-server-credentials` [Scheme Procedure]
Return new PSK server credentials.

`peer-certificate-status` *session* [Scheme Procedure]
Verify the peer certificate for *session* and return a list of `certificate-status` values (such as `certificate-status/revoked`), or the empty list if the certificate is valid.

`set-certificate-credentials-verify-flags!` *cred [flags...]* [Scheme Procedure]
Set the certificate verification flags to *flags*, a series of `certificate-verify` values.

`set-certificate-credentials-verify-limits!` *cred max-bits max-depth* [Scheme Procedure]
Set the verification limits of `peer-certificate-status` for certificate credentials *cred* to *max_bits* bits for an acceptable certificate and *max_depth* as the maximum depth of a certificate chain.

`set-certificate-credentials-x509-keys!` *cred certs privkey* [Scheme Procedure]
Have certificate credentials *cred* use the X.509 certificates listed in *certs* and X.509 private key *privkey*.

`set-certificate-credentials-x509-key-data!` *cred cert key format* [Scheme Procedure]
Use X.509 certificate *cert* and private key *key*, both uniform arrays containing the X.509 certificate and key in format *format*, for certificate credentials *cred*.

`set-certificate-credentials-x509-crl-data!` *cred data format* [Scheme Procedure]
Use *data* (a uniform array) as the X.509 CRL (certificate revocation list) database for *cred*. On success, return the number of CRLs processed.

`set-certificate-credentials-x509-trust-data!` *cred data format* [Scheme Procedure]
Use *data* (a uniform array) as the X.509 trust database for *cred*. On success, return the number of certificates processed.

`set-certificate-credentials-x509-crl-file!` *cred file format* [Scheme Procedure]
Use *file* as the X.509 CRL (certificate revocation list) file for certificate credentials *cred*. On success, return the number of CRLs processed.

`set-certificate-credentials-x509-trust-file!` *cred file format* [Scheme Procedure]
Use *file* as the X.509 trust file for certificate credentials *cred*. On success, return the number of certificates processed.

`set-certificate-credentials-x509-key-files!` *cred cert-file key-file format* [Scheme Procedure]
Use *file* as the password file for PSK server credentials *cred*.

`set-certificate-credentials-dh-parameters!` *cred dh-params* [Scheme Procedure]
Use Diffie-Hellman parameters *dh_params* for certificate credentials *cred*.

`make-certificate-credentials` [Scheme Procedure]
Return new certificate credentials (i.e., for use with either X.509 or OpenPGP certificates.

`set-anonymous-server-dh-parameters!` *cred dh-params* [Scheme Procedure]
Set the Diffie-Hellman parameters of anonymous server credentials *cred*.

`make-anonymous-client-credentials` [Scheme Procedure]
Return anonymous client credentials.

`make-anonymous-server-credentials` [Scheme Procedure]
Return anonymous server credentials.

`set-session-dh-prime-bits!` *session bits* [Scheme Procedure]
Use *bits* DH prime bits for *session*.

`pkcs3-export-dh-parameters` *dh-params format* [Scheme Procedure]
Export Diffie-Hellman parameters *dh_params* in PKCS3 format according for *format* (an `x509-certificate-format` value). Return a `u8vector` containing the result.

`pkcs3-import-dh-parameters` *array format* [Scheme Procedure]
Import Diffie-Hellman parameters in PKCS3 format (further specified by *format*, an `x509-certificate-format` value) from *array* (a homogeneous array) and return a new `dh-params` object.

`make-dh-parameters` *bits* [Scheme Procedure]
Return new Diffie-Hellman parameters.

`set-session-transport-port!` *session port* [Scheme Procedure]
Use *port* as the input/output port for *session*.

`set-session-transport-fd!` *session fd* [Scheme Procedure]
Use file descriptor *fd* as the underlying transport for *session*.

`set-session-record-port-close!` *port close* [Scheme Procedure]
Set *close*, a one-argument procedure, as the procedure called when *port* is closed. *close* will be passed *port*. It may be called when `close-port` is called on *port*, or when *port* is garbage-collected. It is a useful way to free resources associated with *port* such as the session's transport file descriptor or port.

`session-record-port` *session* [*close*] [Scheme Procedure]
Return a read-write port that may be used to communicate over *session*. All invocations of `session-port` on a given session return the same object (in the sense of `eq?`).

If *close* is provided, it must be a one-argument procedure, and it will be called when the returned port is closed. This is equivalent to setting it by calling `set-session-record-port-close!`.

`record-get-direction` *session* [Scheme Procedure]
Determine whether GnuTLS was interrupted when sending or receiving from *session*. This information can be used when deciding if to wait to be able to read or write from a socket before retrying. Returns 0 if interrupted when reading and 1 if interrupted when writing.

`record-receive!` *session array* [Scheme Procedure]
Receive data from *session* into *array*, a uniform homogeneous array. Return the number of bytes actually received.

`record-send` *session array* [Scheme Procedure]
Send the record constituted by *array* through *session*.

`set-session-server-name!` *session type name* [Scheme Procedure]
For a client, this procedure provides a way to inform the server that it is known under *name*, *via* the `SERVER NAME` TLS extension. *type* must be a `server-name-type` value, *server-name-type/dns* for DNS names.

`set-session-credentials!` *session cred* [Scheme Procedure]
Use *cred* as *session*'s credentials.

`cipher-suite->string` *kx cipher mac* [Scheme Procedure]
Return the name of the given cipher suite.

`set-session-priorities!` *session priorities* [Scheme Procedure]
Have *session* use the given *priorities* for the ciphers, key exchange methods, MACs and compression methods. *priorities* must be a string (see Section "Priority Strings" in *GnuTLS, Transport Layer Security Library for the GNU system*). When *priorities* cannot be parsed, an `error/invalid-request` error is raised, with an extra argument indication the position of the error.

`set-session-default-priority!` *session* [Scheme Procedure]
Have *session* use the default priorities.

`set-server-session-certificate-request!` *session request* [Scheme Procedure]
Tell how *session*, a server-side session, should deal with certificate requests. *request* should be either `certificate-request/request` or `certificate-request/require`.

`session-our-certificate-chain` *session* [Scheme Procedure]
Return our certificate chain for *session* (as sent to the peer) in raw format (a u8vector). In the case of OpenPGP there is exactly one certificate. Return the empty list if no certificate was used.

`session-peer-certificate-chain` *session* [Scheme Procedure]
Return the a list of certificates in raw format (u8vectors) where the first one is the peer's certificate. In the case of OpenPGP, there is always exactly one certificate. In the case of X.509, subsequent certificates indicate form a certificate chain. Return the empty list if no certificate was sent.

`session-client-authentication-type` *session* [Scheme Procedure]
Return the client authentication type (a `credential-type` value) used in *session*.

`session-server-authentication-type` *session* [Scheme Procedure]
Return the server authentication type (a `credential-type` value) used in *session*.

`session-authentication-type` *session* [Scheme Procedure]
Return the authentication type (a `credential-type` value) used by *session*.

`session-protocol` *session* [Scheme Procedure]
Return the protocol used by *session*.

`session-certificate-type` *session* [Scheme Procedure]
Return *session*'s certificate type.

`session-compression-method` *session* [Scheme Procedure]
Return *session*'s compression method.

`session-mac` *session* [Scheme Procedure]
Return *session*'s MAC.

`session-kx` *session* [Scheme Procedure]
Return *session*'s kx.

`session-cipher` *session* [Scheme Procedure]
Return *session*'s cipher.

`alert-send` *session level alert* [Scheme Procedure]
Send *alert* via *session*.

`alert-get` *session* [Scheme Procedure]
Get an aleter from *session*.

`reauthenticate` *session* [Scheme Procedure]
Perform a re-authentication step for *session*.

`rehandshake` *session* [Scheme Procedure]
Perform a re-handshaking for *session*.

`handshake` *session* [Scheme Procedure]
Perform a handshake for *session*.

`bye` *session how* [Scheme Procedure]
Close *session* according to *how*.

`make-session` *end* [*flags...*] [Scheme Procedure]
Return a new session for connection end *end*, either `connection-end/server` or `connection-end/client`. The optional *flags* arguments are `connection-flag` values such as `connection-flag/auto-reauth`.

`gnutls-version` [Scheme Procedure]
Return a string denoting the version number of the underlying GnuTLS library, e.g., `"1.7.2"`.

`openpgp-keyring?` *obj* [Scheme Procedure]
Return true if *obj* is of type `openpgp-keyring`.

`openpgp-private-key?` *obj* [Scheme Procedure]
Return true if *obj* is of type `openpgp-private-key`.

`openpgp-certificate?` *obj* [Scheme Procedure]
Return true if *obj* is of type `openpgp-certificate`.

`private-key?` *obj* [Scheme Procedure]
Return true if *obj* is of type `private-key`.

`public-key?` *obj* [Scheme Procedure]
Return true if *obj* is of type `public-key`.

`cipher-hd?` *obj* [Scheme Procedure]
Return true if *obj* is of type `cipher-hd`.

`aead-cipher?` *obj* [Scheme Procedure]
Return true if *obj* is of type `aead-cipher`.

`hash?` *obj* [Scheme Procedure]
Return true if *obj* is of type `hash`.

`hmac?` *obj* [Scheme Procedure]
Return true if *obj* is of type `hmac`.

`x509-private-key?` *obj* [Scheme Procedure]
Return true if *obj* is of type `x509-private-key`.

`x509-certificate?` *obj* [Scheme Procedure]
Return true if *obj* is of type `x509-certificate`.

`psk-client-credentials?` *obj* [Scheme Procedure]
Return true if *obj* is of type `psk-client-credentials`.

`psk-server-credentials?` *obj* [Scheme Procedure]
Return true if *obj* is of type `psk-server-credentials`.

`srp-client-credentials?` *obj* [Scheme Procedure]
Return true if *obj* is of type `srp-client-credentials`.

`srp-server-credentials?` *obj* [Scheme Procedure]
Return true if *obj* is of type `srp-server-credentials`.

`certificate-credentials?` *obj* [Scheme Procedure]
Return true if *obj* is of type `certificate-credentials`.

`dh-parameters?` *obj* [Scheme Procedure]
Return true if *obj* is of type `dh-parameters`.

`anonymous-server-credentials?` *obj* [Scheme Procedure]
Return true if *obj* is of type `anonymous-server-credentials`.

`anonymous-client-credentials?` *obj* [Scheme Procedure]
Return true if *obj* is of type `anonymous-client-credentials`.

`session?` *obj* [Scheme Procedure]
Return true if *obj* is of type `session`.

`openpgp-certificate-format->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `openpgp-certificate-format` value.

`random-level->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `random-level` value.

`ecc-curve->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `ecc-curve` value.

`oid->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `oid` value.

`privkey->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `privkey` value.

`error->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `error` value.

`certificate-verify->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `certificate-verify` value.

`key-usage->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `key-usage` value.

`psk-key-format->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `psk-key-format` value.

`server-name-type->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `server-name-type` value.

`sign-algorithm->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `sign-algorithm` value.

`pk-algorithm->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `pk-algorithm` value.

`x509-subject-alternative-name->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `x509-subject-alternative-name` value.

`x509-certificate-format->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `x509-certificate-format` value.

`certificate-type->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `certificate-type` value.

`protocol->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `protocol` value.

`close-request->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `close-request` value.

`certificate-request->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `certificate-request` value.

`certificate-status->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `certificate-status` value.

`handshake-description->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `handshake-description` value.

`alert-description->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `alert-description` value.

`alert-level->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `alert-level` value.

`connection-flag->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `connection-flag` value.

`connection-end->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `connection-end` value.

`compression-method->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `compression-method` value.

`digest->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `digest` value.

`mac->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `mac` value.

`credentials->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `credentials` value.

`params->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `params` value.

`kx->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `kx` value.

`cipher->string` *enumval* [Scheme Procedure]
Return a string describing *enumval*, a `cipher` value.

Appendix A Copying Information

GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at

your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `https://www.gnu.org/licenses/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    "CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    "Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Procedure Index

I

K

M

O

P

R

S

X

Concept Index