Java TOTP Server Library

Java TOTP Server Library

Time-Based One-Time Password (TOTP)
Java Server Library 1.2.0, 24 July 2018

Enrico M. Crisostomo

This manual is for the Java TOTP server library (version 1.2.0, 24 July 2018).

Short Contents

Table of Contents

1 Introduction

The Time-Based One-Time Password algorithm (TOTP) (RFC 6238) is a an extension of the HMAC-based One-Time Password algorithm (HOTP) (RFC 4226) which is the basis of many two-phase authentication solutions such as Google's, Microsoft's and others.

The *Google Authenticator* client application is probably the most widely used software-based TOTP token generator, and is available as a free application on all the mainstream mobile operating systems. It is probably the de-facto standard TOTP password generator and it is used to manage not only Google's, but other accounts' passwords as well. Google Authenticator is a flexible, easy to use, easy to configure, free application that can be leveraged by any system, provided it has a back-end that implements the functionality to validate a TOTP password. This Java library provides the building blocks of such a back-end:

- Credential generation.
- Credential verification.
- QR code generation[1] for the easy configuration[2] of an account in the Google Authenticator application.

1.1 Design Philosophy

This library has been designed with *simplicity* in mind, trying to follow the KISS principle. This basically means that the library *does one thing*, and it strives hard *to do it well*: TOTP authentication. No more, no less.

There exist more complex and comprehensive solutions, and if you need something more than being able to perform TOTP authentication, chances are you should assess whether to use this library or some other product.

On the other hand, if what you are looking *is* performing TOTP authentication, then this library will provide an easy to use API and a *very* compact library: currently, the size of the library JAR archive is smaller than 20 kB. Most importantly, this library has very few[3] dependencies:

- The *Apache Commons Codec* library.
- The *Apache HTTP components* library.

The library contains Java SE code and the back-end components are implemented as POJOs (using current Java jargon). Therefore, it can be in-

[1] The QR code can be obtained using a generated URL, which relies on the Google Chart HTTP API to create a compliant QR code encoding the necessary information.

[2] Google Authenticator can use the built-in camera of a cellular phone to take a picture of a QR code and configure a new account using the information contained therein.

[3] This library's dependencies are compact as well and do not pull-in a chain of dependencies themselves.

cluded and used in any Java application and does not have dependencies with any framework or technology[4] other than a Java SE 7 runtime environment.

[4] Earlier versions of this library had a dependency with JAX-RS v. 2 that was removed in v. 0.4.5.

2 Overview

The TOTP algorithm (specified by RFC 6238) is an extension of the HOTP algorithm that uses a time-based moving factor instead of an event counter. One of the purposes of that work is, in the words of the authors:

> The proposed algorithm can be used across a wide range of network applications, from remote Virtual Private Network (VPN) access and Wi-Fi network logon to transaction-oriented Web applications. The authors believe that a common and shared algorithm will facilitate adoption of two-factor authentication on the Internet by enabling interoperability across commercial and open-source implementations.
>
> —*RFC 6238*

In fact, as expected by the authors, both two-factor authentication and TOTP are being widely adopted. Google enabled TOTP-based two-factor authentication for its users and developed a software TOTP token generator, *Google Authenticator*, running on all the mainstream smartphone operating systems, which is now the de-facto standard token generator (at least for non-corporate use).

TOTP-based two-factor authentication schemes usually require the user to send its TOTP password alongside its credentials. The TOTP password is time-based, it is calculated with a cryptographic algorithm and its validity is very short: RFC 6238 recommends to used a time-step size of 30 seconds. The very short expiration time of each password is a key security factor against attacks.

The security analysis (RFC 4226) of this algorithm concludes that the best attack against the HOTP function is the *brute force* attack. The security of the algorithm can be approximate (RFC 4226) as

$$\text{Security} = \frac{s \cdot v}{10^d}$$

where s is the *look-ahead synchronization window size*[1], v is the number of *verification attempts* and d is the number of *digits* in the TOTP password. That is: if a brute force attack with v verification attempts is used, then its probability of success is

$$p = \frac{s \cdot v}{10^d}$$

RFC 4226 recommends that $d \geq 6$, and this library validates 6-digit TOTP passwords, which is the password length currently used by Google Authenticator. Passwords of up to 9 decimal digits could be used, which is the maximum that can be represented by the 31-bit string returned by the *dynamic truncation* function.

[1] Referred to simply as *window size* in the rest of this manual.

Evaluating the probability p of making a successful attack with $s = 10$ and $d = 6$ yields

$$p = \frac{v}{10^5}$$

that is, v successes every 10.000 attempts. By limiting the number of unsuccessful authentication attempts a user can perform, a superior bound on p can be established that satisfies the security requirements.

3 API Description

The TOTP algorithm is used to determine whether the password provided by the *prover*[1] is proved as authentic by the *verifier*, given the prover's *shared secret* and current Unix time.

The two basic operations that this library implements are:

- Credential *creation*.
- Credential *authentication*.

Credential creation is the process in which the server generates the *shared secret* and shares it with the client. The shared secret is generated using a cryptographically strong pseudorandom generator and has to be communicated to the client, so that he can store it and configure its token. A common way to communicate this information is sending it to the client encoded into a QR code. The user only needs to scan the QR code with the Google Authenticator application and then discard it immediately. For the sake of security, TOTP token generators do not generally let users retrieve the shared secrets after an account has been configured. This way, only individuals in physical possession of a token can use the shared secret to generate TOTP passwords.

Credential authentication is the process in which the server applies the TOTP algorithm to the shared secret and the token-generated password of the prover to determine its authenticity.

All the API members are published by the following interfaces and classes:

`IGoogleAuthenticator`
: This is the main API interface and it publishes the library entry points.

`GoogleAuthenticatorKey`
: This class is the credentials container, mainly used when returning newly-created credentials.

`GoogleAuthenticatorQRGenerator`
: This helper class provides QR code generation using the Google Chart HTTP API.

`ICredentialRepository`
: This interface can optionally be implemented by callers to provide callback functions to the library used to interact with a 'credential repository', where newly-generated credentials are *stored* and keys are *retrieved* when a password must be validated.

[1] The user providing the password generated by his token.

3.1 Random Number Generator Algorithm and Provider

The creation of a shared secret is an important process because the ability of taking over or guessing somebody else's secret could result in account hijacking. For this reason, and since it is not always practicable or desirable to devise such a creation strategy, this library offers the possibility of creating shared secrets using data coming from a cryptographically strong pseudorandom generator (PRNG). The default provider (`SUN`) and the default algorithm (`SHA1PRNG`) used by this library are available on Oracle Java runtime environments and they were not tested on other implementations where they are likely to be missing.

Users can configure the library to use an alternate provider and algorithm by providing their names as values of the following system properties:

`com.warrenstrange.googleauth.rng.algorithm`
: If a property with this name is set, the specified random number generator algorithm is used.

`com.warrenstrange.googleauth.rng.algorithmProvider`
: If a property with this name is set, the specified random number generator algorithm provider is used. The provider can be overridden if and only if the algorithm is overridden as well.

While explicitly specifying a provider and an algorithm may impair interoperability, on the other hand it guarantees the expected level of randomness and a predictable behaviour. The performance problems often reported when using a blocking entropy gathering device (see [NativePRNG], page 6) are an example of why the authors chose the current defaults.

3.1.1 `NativePRNG`

On some environments the `NativePRNG` algorithm may be used by default. This algorithm uses the `/dev/random` device to get random data and this device is *blocking* on certain platforms (such as Linux). If the blocking behaviour is undesirable and `/dev/urandom` can be used as a valid alternative entropy gathering device, then the `NativePRNG` algorithm can be configured to get random data from `/dev/urandom` by setting the system property `java.security.egd` to `/dev/urandom`, for example passing the following argument to the Java virtual machine:

```
-Djava.security.egd=file:/dev/urandom
```

3.2 `ICredentialRepository`-Related Methods

The `IGoogleAuthenticator` interface originally provided the following methods:

`createCredentials()`
: Create new credentials.

`authorize(String secret, int verificationCode)`
: Validate the specified `verificationCode` with the specified `secret` using the TOTP algorithm.

When support for the `ICredentialRepository` callbacks was designed, a new set of methods were added to allow the user to specify a *`userName`* to be used to save or retrieve his key. Such methods are the following:

`createCredentials(String userName)`
: Create new credentials.

`authorizeUser(String userName, int verificationCode)`
: Validate the specified `verificationCode` with the secret of the user with the specified `userName` using the TOTP algorithm.

Their functionality is identical to that of the aforementioned methods, but in this case no *`secret`* is returned or passed as a parameter, but is saved or retrieved from the configured `ICredentialRepository` instance instead.

In the following sections the first generation methods will be used for tutorial purposes, but any of those API calls could be substituted by the corresponding `ICredentialRepository`-enabled call.

3.3 Credential Creation

A new set of credentials can be created using the `createCredentials` method:

```
GoogleAuthenticatorKey createCredentials();
```

This method computes a new random shared secret wrapped into a `GoogleAuthenticatorKey` instance.

3.3.1 Credentials

The `GoogleAuthenticatorKey` class represents newly generated credentials, as returned by `createCredentials`. Instances of this class contains the following data members:

- The shared secret `key`.
- The *verification code*[2] `verificationCode`. The verification code is an optional feature which is not used by some clients such as Google Authenticator.
- A list of *scratch codes* `scratchCodes`. Scratch codes are randomly generated data[3] that are optionally used as 'recovery passwords' in case the token generator is not available. If this feature is implemented, scratch codes should be usable only *once*. This library provides scratch codes

[2] The verification code is the TOTP password calculated at $t = u$, where u is the Unix Epoch.

[3] Generated together with the shared secret using the same, cryptographically strong pseudorandom generator.

as an ancillary feature, to offer the client randomly generated data with the same guarantees the shared secret offers. How scratch codes are used, however, is a responsibility of the prover and this library offers no facility to store them or validate them.

3.4 Generating a QR Code

The Google Authenticator application can be quickly configured using a QR code: the application requests the user to take a photograph of the code and the application uses the data encoded therein to configure a new account.

This approach has several advantages: human errors are reduced to a minimum, or eliminated altogether, the setup process is easy and quick, but most importantly, the shared secret is *never* shown in plain text. Unless a malicious user succeeds in stealing a usable picture of the QR code to configure another Google Authenticator instance with the stolen credentials, the shared secret cannot be read, not even by the legitimate owner himself. Therefore, QR code greatly reduce the risk of credentials being intercepted during the delicate phase of the initial interchange.

This library supports this use case providing a ready-to-use Google Chart HTTP API call to display a QR code encoding all the data of the newly generated credentials:

- The *issuer*.
- The *account name*.
- The *label*.
- The *shared secret*.

The QR code encoding the aforementioned data can be created invoking the `GoogleAuthenticatorQRGenerator.getOtpAuthURL()` method:

```
public static String getOtpAuthURL(
  String issuer,
  String accountName,
  GoogleAuthenticatorKey credentials);
```

as in the following example, where `Test Org.` is the *issuer* and `test@test.org` is the account name[4]:

```
final GoogleAuthenticatorKey key =
  googleAuthenticator.createCredentials();
final String otpAuthURL =
  GoogleAuthenticatorQRGenerator.getOtpAuthURL(
    "Test Org.",
    "test@test.org",
    key);
```

The `GoogleAuthenticatorQRGenerator.getOtpAuthURL` method will return a Google Chart API URL which can then be used to generate the QR

[4] Account names often are email addresses, but they need *not* be.

code image. An example URL that generates a valid QR code encoding Google Authenticator's configuration settings is the following[5]:

```
https://chart.googleapis.com/chart? \
  chs=200x200&                      \
  chld=M%7C0&                       \
  cht=qr&                           \
  chl=otpauth%3A%2F%2Ftotp%2FTest%2520Issuer%3Atest%40issuer.org%3F \
      secret%3D7GYQCQ2KA34VADUR%26issuer%3DTest%2BIssuer
```

The first query parameters are 'technical' and are passed to the Google Chart API to configure the chart to generate. The `chl` parameter carries the data to be encoded, which must be UTF-8 URL-encoded[6].

3.4.1 The `otpauth` Scheme

Google Authenticator expects an URI conforming to the otpauth scheme to configure a new account.

`otpauth://`*`type/label?parameters`*

This URI is generated by the library using the information passed to the API methods. The most important pieces of information are:

- The *`type`*, which is *always* `totp`.
- The *`label`*, conforming to the following ABNF grammar:

  ```
  label = accountname / issuer (":" / "%3A") *"%20" accountname
  ```

- The *`parameters`*. The library automatically adds the two required parameters `issuer` and `secret`. The secret is the shared secret, encoded in Base32 (RFC 5348).

3.4.1.1 The Issuer

The issuer identifies the service provider associated with the account being created, and it must be URL-encoded. The issuer information is *optional* and is present in both the label and the query parameters of an `optauth` URI. Although optional, it is *strongly recommended* to include it.

3.4.2 The `QR` Code

The Google Chart API currently generates the QR code as an image in PNG format (see Figure 3.1).

---

[5] Lines are splitted with `\` and intented for better readability and separation of the query string parameters..

[6] The encoding of all the URL fragments is performed by the library.

Figure 3.1: A Google Chart-generated QR code.

Since one of the reasons why QR codes are used in the first place is making it more difficult to a man in the middle to steal a newly-generated identity, and since it encodes the shared secret, implementors should make sure the QR code is treated as securely as any other kind of credential.

3.5 Generating a Password

A TOTP password can be created using one of the `getTotpPassword*` methods, specifying:

- The secret or the username.
- The time for which the password should be generated.

3.6 Validating a Password

A TOTP password is validated using one of the different `authorize*` methods provided by the library. To validate a password, the TOTP algorithm requires:

- The password to validate.
- The shared secret[7].
- The time-based moving factor.

The client and the server should agree on how to calculate the time-based moving factor. RFC 6238 *recommends* using a default time-step size of 30 seconds and this library adheres to that recommendation. The library API allows callers to specify the number of time-step windows that should be checked during the validation process, but does not currently allows the time-step size to be overridden.

When the number x of time-step windows to use is specified, the implementation will check all the integral time-step in an interval I (roughly) centered in the current instant of time: $I = [\lfloor -(x-1)/2 \rfloor, \lfloor x/2 \rfloor]$.

[7] The library API treats one-time passwords as integers.

4 API

The API is currently published by the following classes and interfaces:

`GoogleAuthenticator`
: The `IGoogleAuthenticator` implementation.

`GoogleAuthenticatorConfig`
: The configuration parameters used during the TOTP password validation, including time step size, window size, number of digits and key representations.

`GoogleAuthenticatorConfig.GoogleAuthenticatorConfigBuilder`
: The factory for `GoogleAuthenticatorConfig` instances.

`GoogleAuthenticatorException`
: The root exception used by the library.

`GoogleAuthenticatorKey`
: This class is the credentials container, mainly used when returning newly-created credentials.

`GoogleAuthenticatorQRGenerator`
: Helper class providing QR code generation using the Google Chart HTTP API.

`ICredentialRepository`
: This interface can optionally be implemented by callers to provide callback functions to the library used to interact with a 'credential repository', where newly-generated credentials are *stored* and keys are *retrieved* when a password must be validated.

`IGoogleAuthenticator`
: This is the main API interface and it publishes the library entry points.

`KeyRepresentation`
: An enumeration of all the available secret key representations, currently:

  - `BASE32`
  - `BASE64`

`ReseedingSecureRandom`
: A wrapper class around `SecureRandom` that takes care of reseeding the wrapper instance every certain number of operations. Currently, this parameter is not overridable.

4.1 GoogleAuthenticator

The `GoogleAuthenticator` class is the provided implementation of the `IGoogleAuthenticator` interface. This class adds no public methods to those defined in its primary interface.

This implementation conforms to the recommendations and the default parameter values specified in RFC 6238. Specifically, this class uses:

- The HMAC-SHA-1 algorithm is used, as specified by RFC 4226. RFC 6238 specifies that implementors *may* also use HMAC-SHA-256 and HMAC-SHA-512 but, although the code already supports them, currently there is no way to override the usage of HMAC-SHA-1.
- A time-step size of 30 seconds is used, as recommended in by RFC 6238.

The parameters used during the TOTP password validation are:

- Time step size.
- Windows size.
- Number of digits (key modulus).

Furthermore, the secret key can be provided in one of the supported representations:

- `BASE32`.
- `BASE64`.

Base32 is the representation used by default by Google Authenticator, and Base64 is offered since it is the default representation used by RFC 6030, *Portable Symmetric Key Container (PSKC)*, since it is commonly used by software platforms and hardware TOTP token providers.

All these parameters can be configured by passing an instance of the `GoogleAuthenticatorConfig` class to the constructor of the `GoogleAuthenticator` class.

4.1.1 GoogleAuthenticatorConfigBuilder

`GoogleAuthenticatorConfig` instances can only be built using `GoogleAuthenticatorConfigBuilder` instances. The following fragment, for example, builds a `GoogleAuthenticator` instance with the following characteristics:

- It uses a time step size of one minute.
- It validate TOTP password checking an interval of 5 time steps centered at the current time.
- It validates and creates TOTP passwords of 8 digits.
- It requests 10 scratch codes.

```
GoogleAuthenticatorConfigBuilder gacb =
  new GoogleAuthenticatorConfigBuilder()
    .setTimeStepSizeInMillis(TimeUnit.SECONDS.toMillis(60))
    .setWindowSize(5)
```

```
        .setCodeDigits(8)
        .setNumberOfScratchCodes(10);
    GoogleAuthenticator ga =
      new GoogleAuthenticator(gacb.build());
```

4.2 GoogleAuthenticatorException

This class is the root exception used by library methods. The exception is a subtype of `RuntimeException` so that library method callers need neither catch nor declare this exception in their calling methods.

4.3 GoogleAuthenticatorKey

This inmutable class is a JavaBean used by the GoogleAuthenticator library to represent a newly-created set of credentials. Currently, this class publishes the following read-only properties:

`GoogleAuthenticatorCofig config`
: The TOTP configuration of this key.

`String key`
: The shared secret.

`List<Integer> scratchCodes`
: A list of scratch codes. By default, 5 scratch codes are provided. There currently is no way to generate scratch codes without creating a new credential.

`int verificationCode`
: The verification code, that is, the TOTP password when at time-step 0 (the Unix Epoch).

4.3.1 GoogleAuthenticatorKey.Builder

Instances of `GoogleAuthenticatorKey` can only be built using `GoogleAuthenticatorKey.Builder` instances. The builder publishes all the properties of the `GoogleAuthenticatorKey` class. The following fragment, for example, builds an instance with the following characteristics:

- It has a default `GoogleAuthenticatorConfig`.
- It has the following secret key: `secretKey`.
- It has the following verification code: `123456`.
- It has an empty list of scratch codes.

```
GoogleAuthenticatorConfig config =
  new GoogleAuthenticatorConfig
    .GoogleAuthenticatorConfigBuilder()
    .build();

GoogleAuthenticatorKey credentials =
```

```
new GoogleAuthenticatorKey
  .Builder("secretKey")
  .setConfig(config)
  .setVerificationCode(123456)
  .setScratchCodes(new ArrayList<Integer>())
  .build();
```

Manually building instances of this class is generally useful to re-create QR codes for an existing key.

4.4 GoogleAuthenticatorQRGenerator

This class is a helper class which provides a way to generate QR codes using the Google Chart API. Please note that the usage of this method is subject to the license and usage restrinctions of the Google Chart API.

This class publishes the following method:

`static String getOtpAuthURL(String issuer, String accountName, GoogleAuthenticatorKey credentials)`
: Returns the URL of a Google Chart API call to generate a QR barcode to be loaded into the Google Authenticator application. The user scans this bar code with the application on their smart phones or enters the secret manually.

`static String getOtpAuthTotpURL(String issuer, String accountName, GoogleAuthenticatorKey credentials)`
: Returns the key URI, that is a `otpauth://` URL as specified by the key URI format, to be loaded in compliant applications.

[Generating a QR Code], page 8, provides a detailed description of this method's functionality.

4.5 ICredentialRepository

The `ICredentialRepository` is a service interface which can be implemented by a library user to provide a callback that establishes a relation between a user name and its shared secret. The methods of the API that accept a user name instead of the shared secret will use this callback to retrieve that user's shared key. This interface is optional and needs not be implemented, unless you want the library code to perform basic user management tasks. Methods accepting user names are used when no `ICredentialRepository` service is available will fail.

This interface defines the following methods:

`String getSecretKey(String userName);`
: This method returns the shared secret of the specified user. If the specified user does not exist, the method shall *fail* throwing a `GoogleAuthenticatorException`.

`void saveUserCredentials(String userName, String secretKey, int validationCode, List<Integer> scratchCodes);`
: This method saves the credentials of the specified user.

The service lookup mechanism used by the library is the `ServiceLoader` facility provided by Java SE. Implementors needs creating a file named after the service interface, that is

`com.warrenstrange.googleauth.ICredentialRepository`

and one line with the name of each implementing class. The library will use the `ServiceLoader` class to discover the available implementations and will use the *first* available.

4.5.1 Setting a credential repository on a per-instance basis

Since 1.0.0, the `ServiceLoader` API is not the only way to set the credential repository: the `setCredentialRepository(ICredentialRepository)` method can be used to explicitly set the credential repository used by an `IGoogleAuthenticator` instance, effectively disabling the autmatic service discovery described in the previous section.

4.5.2 Disabling the credential repository feature

Sometimes it may be desirable to disable this feature on specific instances. This can be achieved by passing `null` to the `setCredentialRepository` method.

4.6 `IGoogleAuthenticator`

This interfaces publishes the main library API:

`GoogleAuthenticatorKey createCredentials();`
: This method generates a new set of credentials including:

  - Secret key.
  - Validation code.
  - A list of scratch codes.

  The user must register this secret on their device.

`GoogleAuthenticatorKey createCredentials(String userName);`
: This method generates a new set of credentials invoking the `createCredentials` method with no arguments. The generated credentials are then saved using the configured `ICredentialRepository` service.

  The user must register this secret on their device.

`boolean authorize(String secret, int verificationCode);`
: Checks a verification code against a secret key using the current time. The algorithm also checks in a time window whose size

determined by the `windowSize` property of this class. The default value of 30 seconds recommended by RFC 6238 is used for the interval size.

`boolean authorize(String secret, int verificationCode, long time);`
: Checks a verification code against a secret key the specified time. The algorithm also checks in a time window whose size determined by the `windowSize` property of this class. The default value of 30 seconds recommended by RFC 6238 is used for the interval size.

`boolean authorizeUser(String userName, int verificationCode);`
: This method validates a verification code of the specified user whose private key is retrieved from the configured credential repository. This method delegates the validation to the `authorize(String, int)` method.

`boolean authorizeUser(String userName, int verificationCode, long time);`
: This method validates a verification code of the specified user whose private key is retrieved from the configured credential repository. This method delegates the validation to the `authorize(String, int, long)` method.

`ICredentialRepository getCredentialRepository();`
: This method returns the credential repository used by this instance, or `null` if none is set or none can be found using the *ServiceLoader* API.

`void setCredentialRepository(ICredentialRepository repository);`
: This method sets the credential repository used by this instance. If `null` is passed to this method, no credential repository will be used, nor discovered using the *ServiceLoader* API.

`getTotpPassword(String secret);`
: This method generates the current TOTP password using the specified secret.

`getTotpPassword(String secret, long time);`
: This method generates the TOTP password at the specified time using the specified secret.

`getTotpPasswordOfUser(String userName);`
: This method generates the current TOTP password using key of the specified user.

`getTotpPasswordOfUser(String userName, long time);`
: This method generates the TOTP password at the specified time using key of the specified user.

Appendix A 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 `http://www.gnu.org/copyleft/`.

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.

Index

(Index is nonexistent)