User guide for Sequoia-PGP `sq`

Keeping private things private

The Sequoia-PGP project

v0.1-43-g6930040

Contents

Document status

This document is very much a work in progress. Nothing is finished and final. Parts haven't been written yet. That said, feedback on what is written, or the structure of the document is very much welcome. The source code for this document is in version control on the `gitlab.com` site at:

`https://gitlab.com/sequoia-pgp/sq-user-guide/`

If you find mistakes or missing parts from the outline, please open an issue or a merge request.

Prelude: quick start

If you already understand the core cryptographic concepts in OpenPGP, and you're in a hurry to get started, this chapter is for you. This chapter distils the main content of this guide into examples showing a small number of common use cases. No explanations.

```
$ sq key generate --userid="My Name" --userid="<me@example.com>" --export=key.pgp
$ sq key extract-cert --output=cert.pgp key.pgp
$ ls -l
total 8
-rw-rw-r-- 1 liw liw 1772 Oct 15 16:18 cert.pgp
-rw-rw-r-- 1 liw liw 1967 Oct 15 16:18 key.pgp
-rw-rw-r-- 1 liw liw  476 Oct 15 16:18 key.pgp.rev
```

```
$ sq sign --signer-key=key.pgp --output=foo.pgp foo.md
$ sq sign --signer-key=key.pgp --detached --output=foo-sig.pgp foo.md
$ ls -l foo*
-rw-r--r-- 1 liw liw 1086 Oct 15 16:19 foo.md
-rw-rw-r-- 1 liw liw 1825 Oct 15 16:20 foo.pgp
-rw-rw-r-- 1 liw liw  325 Oct 15 16:20 foo-sig.pgp
```

```
$ sq verify --signer-cert=cert.pgp --output=checked.md foo.pgp
Good signature from 84B292ABCE27285B
1 good signature.
$ sq verify --signer-cert=cert.pgp --detached foo-sig.pgp foo.md
Good signature from 84B292ABCE27285B
1 good signature.
$ ls -l checked*
-rw-rw-r-- 1 liw liw 1086 Oct 15 16:23 checked.md
```

```
$ sq encrypt --recipient-cert=cert.pgp --signer-key=key.pgp --output=bar.pgp foo.md
$ ls -l bar.pgp
-rw-rw-r-- 1 liw liw 2076 Oct 15 16:26 bar.pgp
```

```
$ sq decrypt --recipient-key=key.pgp --signer-cert=cert.pgp --output=decrypted.md bar.pgp
Encrypted using AES with 256-bit key
Compressed using ZIP
Good signature from 84B292ABCE27285B
1 good signature.
$ cmp foo.md decrypted.md
$ ls -l decrypted.md
-rw-rw-r-- 1 liw liw 1086 Oct 15 16:27 decrypted.md
```

Chapter 1

Introduction

1.1 What are Sequoia-PGP and `sq`?

The Sequoia-PGP project works to make use of cryptography for privacy and authentication in communication more commonplace. The project produces and maintains an implementation of the OpenPGP standard that's easy and uncomplicated to use.

OpenPGP is used widely in the IT industry and by free and open source projects to verify the authenticity of software packages, and for encrypting and authenticating messages.

`sq` is the command line tool provided by Sequoia-PGP. It's easy and uncomplicated to use. Sequoia-PGP also provides a library for the Rust programming language, called sequoia-openpgp. However, the library is only of interest to software developers, and this guide is aimed at users of the `sq` tool.

1.2 Why use OpenPGP?

The cryptography in OpenPGP helps with three main problems:

- keeping private things private
- identifying if data has changed
- authenticating communication (make sure who it's from)

The OpenPGP standard also specifies protocols and data formats for managing the ecosystem of OpenPGP users. The standard specifies, for example, what form cryptographic messages take. This allows different parties in a communication to use different programs. One correspondent might use a laptop computer, while another may use a mobile phone. The software running on those will be radically

different, but as long as they follow the same standard, secure communication using OpenPGP is possible.

1.3 Who is this guide aimed for?

This guide is aimed at those who want to communicate privately, and receive and send messages and data safely with other people. The guide does not require any background in cryptography, mathematics, or programming, but does require using a computer via the command line.

However, note that this guide aims to get the reader to a point where they can use cryptography at all. It does not aim to teach the reader how to protect against motivated, targeted attacks from organized crime, or intelligence organizations. Such protection will probably build on top of cryptography, but it is outside the scope of this guide.

You can think of this guide as giving you the first turn of the *security rachet*. Every turn of the ratchet improves security a little bit, but achieving protection against targeted attacks will require a great many turns.

1.4 Scope of this guide

This guide covers the important concepts in using cryptography as specified by the OpenPGP standard:

- keys, certificates, and their management
- certifying that a name or email address should be associated with a key
- signing files and verifying signatures
- encrypting and decrypting data

The guide shows how to use the `sq` command line tool from Sequoia-PGP. It does not cover integrating Sequoia-PGP with mail software, version control, file transfer software, or other applications. (That will be covered by other documentation.)

1.5 Structure of the guide

This guide has the following structure:

- The prelude chapter is a guide for getting started quickly. It's aimed at readers who are already familiar with the relevant cryptographic concepts, having perhaps used other software, and just want to see how to do a few basic things. That chapter provides no explanation, just shows the commands. You can safely skip it if you want to.
- The appendices show how to perform tasks to achieve specific goals. These are expanded versions of the prelude examples, but with detailed, step-

by-step explanations of everything. These "how-to" guides are useful for getting things done without having to wade through long discussions about underlying concepts.

- There is also an appendix with a glossary, which can be helpful for looking up unknown terminology, and another appendix with links to additional material relevant to Sequoia-PGP and cryptography.
- The rest of this guide is discussions of the concepts needed to understand how cryptography works, and how to use it well.

Notably, this guide is not meant to be a reference guide. It does not try to cover every aspect of the `sq` tool in detail. The built-in help, which you can get by running `sq help` or `sq encrypt --help`, is always up to date and a good way to look up details.

Chapter 2

Installing `sq`

This chapter explains how to install `sq` in various ways. It is by necessity always going to be incomplete, but the authors would gratefully accept changes for additional target systems.

2.1 On various platforms

2.1.1 Debian

On a Debian system (version 11 or later):

```
apt install sq
```

Note that on Debian 11 (bullseye), the version of `sq` is rather old. You may want to install a back-ported version from the usual Debian location for them.

2.1.2 Fedora

FIXME.

2.1.3 Arch

FIXME.

2.1.4 FreeBSD

FIXME.

2.1.5 OpenBSD

FIXME.

2.1.6 NetBSD

FIXME.

2.1.7 macOS

FIXME.

2.1.8 Windows

FIXME.

2.2 From source code on all platforms

To build and install `sq` from source, you need to have the Rust toolchain installed, in particular `cargo` and `rustc`. You also need a number of non-Rust build dependencies installed; see the README.md for an up-to-date list.

To build and install the latest released version of `sq`, run the following command:

```
$ cargo install sequoia-sq
```

To build `sq` from the current development version, get its source code from GitLab:

```
$ git clone https://gitlab.com/sequoia-pgp/sequoia.git
$ cargo install --path=sequoia/sq
```

Chapter 3

On cryptography

The science of keeping private communication private (confidentiality), verifying that a message hasn't been modified (integrity), and determining who created a message (authentication) is called *cryptography*.

Cryptography is not just for spies. Cryptography allows everyday activities such as shopping and banking to happen without rampant theft. It also allows journalists working on stories about the rich, powerful, or corrupt to communicate with their sources with less fear of prematurely revealing what they're doing.

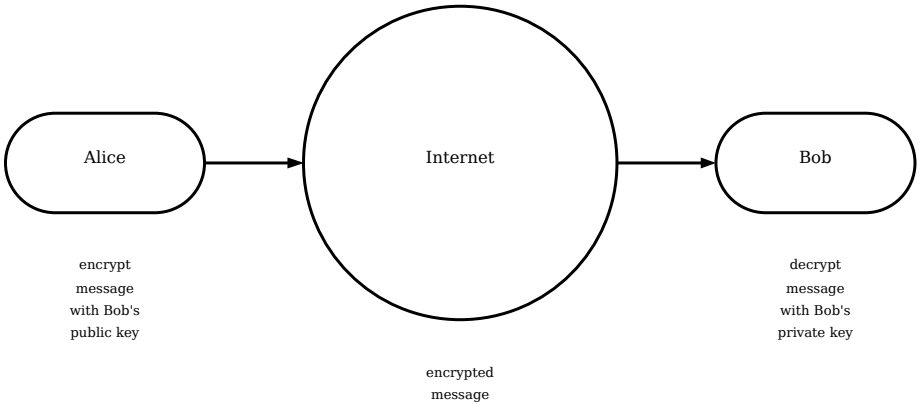
Just as having a seat belt in a car won't help you if you don't use it, or may even hurt you if you use it wrong, you need to understand a few concepts to effectively use cryptography. This chapter presents the essential ideas that you need to understand to not just be safe, but to avoid endangering yourself or others.

If you are concerned about a targeted attack on you or people you communicate with, then this chapter is not enough. You also need training in operational security from a digital security trainer. Freedom of the Press Foundation is one organization that offers training material, and courses.

3.1 Public key cryptography

OpenPGP uses public key cryptography. To use public key cryptography, you need two things: a *public key* and a *private key*.

A public key and a private key form a pair. They work together as follows. Say Alice wants to send a confidential message to Bob. She encrypts the message using Bob's public key, sends him the encrypted message, and Bob decrypts it using his private key:



How Alice sends the message to Bob doesn't matter. Someone who intercepts the message can't decrypt it unless they have Bob's private key. And, even though Alice encrypted the message using Bob's public key, Bob's public key can't be used to help decrypt the message. A public key is a one-way street.

The term public key includes the word public, because for it to be useful, it needs to be widely published: Alice needs Bob's public key to encrypt a message to him.

Likewise, the private key includes the word private, because it should be hidden. If someone else had Bob's private key, they could decrypt the message that Alice sent him.

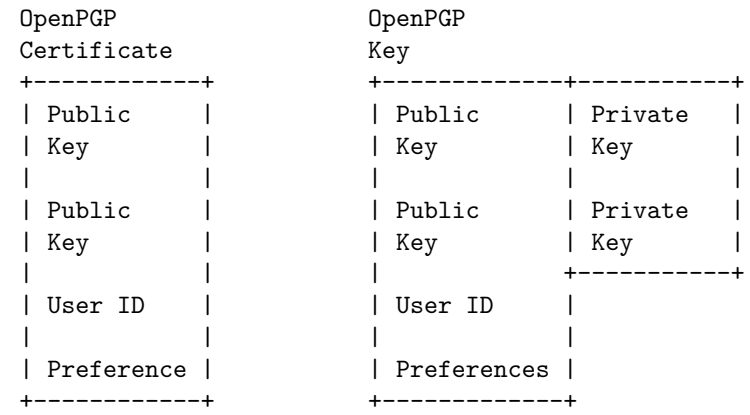
In short: you want people to have access to your public key; it should be public. Your private key, however, is private; like a secret, you shouldn't share your private key with anyone.

Digital signatures work in a similar manner. Alice creates a digital signature using her private key (because no one else should be able to sign a document in her name!). And, to verify a signature, Bob uses her public key, because anyone should be able to verify the signature.

In OpenPGP, your public key is just one part of a thing called a *certificate*. A certificate is a collection of public keys (you need a different one for encryption and signing), some information about you, like your name or alias, and your email address, and information about what features your software supports. A certificate doesn't include your private keys. You can and should share your certificate with people you want to communicate with.

In OpenPGP, private keys are stored in a *key*. A certificate never includes private keys; a key does include private keys. You should share your certificate with other people; you should never share your key with other people.

```
Do Share                Keep Private
```



3.2 Password-based encryption

There is another type of encryption, which uses passwords. This is called symmetric encryption, because you use the same key to encrypt and decrypt a message.

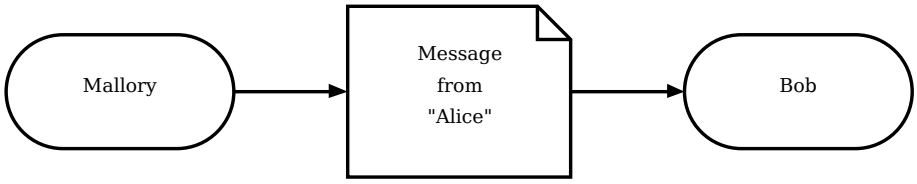
OpenPGP also supports password-based encryption. Oftentimes, your key will be protected with a password so you'll need to enter your password before you can decrypt or sign a message. But, you can also use a password to encrypt a message. Unlike a certificate, if you publish a password, then everyone can decrypt your message. This means passwords are a lot more inconvenient than public keys. Unlike public keys, you can't share them willy-nilly, and you definitely can't publish them in a directory like a telephone book. Passwords have to stay secret to be useful.

3.3 Authentication

Encryption and signing are two of the three essential functions that you need to communicate privately. The last is called authentication. It helps answer the following questions: When Alice sends a message to Bob, does she have the right certificate? And, when Bob receives a message from Alice, can he be sure it really came from her?

Authentication helps prevent two different problems. The first is impersonation. If Alice and Bob communicate regularly, and Bob gets a message that purports to be from Alice, but is written in a different style, then he may become worried that it is not really from Alice. But, if Bob doesn't recognize these social cues, then he might be tricked. This is how phishing works. Today, people are taught to recognize impersonation. This requires schooling, and vigilance. Authentication addresses this problem in a different, more reliable way: if Bob can authenticate Alice's key, and a message is signed using Alice's key, then Bob can be confident

that the message really came from Alice.



The second problem, interception, is more subtle and can't be solved using social cues. If Mallory wants to read what Alice and Bob send to each other, then he can try to eavesdrop on their communication channel.

Encryption is a prerequisite, but it is not sufficient to prevent Mallory from intercepting the messages. Imagine that Alice and Bob send each other their certificates via email. If Mallory is able to intercept these initial, unencrypted messages, then he can replace the certificates with his own. Now, Alice and Bob will have the wrong certificates, and when Alice sends Bob a message, she'll encrypt it using Mallory's certificate. When Mallory intercepts the message, he can decrypt it, since actually Alice encrypted it to him. And, he can even fool Bob by reencrypting it using Bob's real certificate, and forwarding that version to Bob. Bob will be able to decrypt the message as usual and won't suspect a thing!

The only practical way to prevent this type of attack is to authenticate certificates.

Authentication can be done directly. For instance, when Alice and Bob meet in person, Alice and Bob can exchange business cards with their certificates' ID numbers (in OpenPGP, ID numbers are called *fingerprints*). When Bob gets home, he can add what the correct certificate for is Alice to his address book. And, Alice can do the same for Bob. Alice and Bob will now use the right certificate, and will detect an interception attack. This is effective, because it's much harder for Mallory to switch the fingerprints at a physical meeting than to intercept and modify an email.

Another approach is for Alice and Bob to use a trusted third party, which is sometimes called a *certification authority* (*CA*). For instance, if Alice and Bob work at the same company, their IT administrator could record everybody's fingerprint, and publish appropriate certifications in a publicly available directory. Now, Alice just needs to authenticate the IT administrator; she doesn't have to worry about authenticating her coworkers' certificates. A convenient way to run a CA like this is to use OpenPGP CA.

Interception attacks are a real concern. The Government Communications Headquarters (GCHQ), Britain's intelligence and security organization, has proposed

Ghost, an authentication-layer backdoor that they want secure messengers to implement. Their argument is that subverting authentication allows secure messengers to help governments without actually violating their claim that communication is end-to-end encrypted. While technically true, this is the moral equivalent of building a backdoor into the encryption, and is, in effect, a new attempt at the failed Crypto Wars of the 1990s.

3.4 Advantages of public key cryptography

Using public key cryptography allows some very interesting things:

- I can publish my certificate, and anyone can send me a confidential message by encrypting it using my certificate, and be assured that only I can read the message.

  Examples: Activists in oppressive regimes using this can stop their governments from eavesdropping on them. Journalists can protect communications with their sources. Corporations don't need to worry their trade secrets leak to competitors.

- I can send them a reply, sign it with my private key, and the recipient can be sure it's from me, because they can check the signature using my public certificate.

  Examples: The CFO can't be fooled by forged emails from the CEO telling them to pay a fraudulent invoice. Activists can't be fooled by messages from the secret police to go to the town square at noon wearing a pink rose. Journalists can be sure the message is from their source and not someone who's trying to interfere with their reporting.

- These can be combined: If I have their certificate, I can be sure that the message someone send me is really from them, and that only they can read my response.

- I can publish a software release, and sign the file using my private key. Anyone downloading the release can be sure they get what I published by checking the signature with my certificate.

The mathematical and cryptographic details of how this works are outside the scope of this guide, but see the references for links to explanations.

3.5 Limitations of cryptography

When thinking about cryptography it's important to remember that it has limitations. For example, no cryptography can prevent the intended recipient from willfully sharing an encrypted message they receive. If you send a photo of your safe combination to someone encrypted with their certificate, they can decrypt it, and share the picture with the highest bidder.

Also, no cryptography provides any protection if keys aren't kept private. If I accidentally publish my key as a front page advert on the New York Times, cryptography can't prevent others from using that to decrypt messages intended for me, or publishing messages that claim to be from me.

Further, cryptography doesn't protect against violence used to coerce either party in a secure communication from disclosing secrets.

Finally, cryptography relies on some assumptions of what kind of attacks are feasible, whether they're based on mathematics or raw computing power. Over time, attacks on cryptographic protocols, algorithms, and implementations only get stronger. Every few years, attackers have a breakthrough, and some classes of cryptography suddenly become so weak normal people can break them. Then the purveyors and users of cryptography move to newer, stronger alternatives.

For most people, these are quite unlikely scenarios. Most people do not actually have enemies who are a threat specifically to them. If you do, or you suspect you do, be very careful what you do and what advice you follow. You need to seek advice beyond this guide. In particular, you need training in operational security. A digital security trainer can help you. Freedom of the Press Foundation is one organization that offers training material, and courses.

Chapter 4

General principles of the `sq` interface

`sq` is a command line tool using subcommands and options. Global options come before the subcommand on the command line, and options specific to the subcommand come after.

Some options have both long and short forms. Thus, for example, `--output` may be shortened to just `-o`. The examples in this guide use the long form, as that's clearer and easier to understand without explanation. In practice, the short form is often more convenient to type.

Some options are "flags", and the use of the option is enough. For example, to request binary output, the option`--binary` is enough. Other options require more information to be provided. For example, the option `--recipient-key` to specify what key to use for a recipient when encrypting needs to be provided the name of the file in which the key is stored. Such option values can be specified as the command line argument after the option, or appended to the option itself using `=foo` syntax. Thus, the following are equivalent:

```
sq encrypt --recipient-cert cert.pgp --output bar.pgp foo.md
sq encrypt --recipient-cert=cert.pgp --output=bar.pgp foo.md
```

This guide uses the latter syntax to make it clearer when an option is given a value without the reader having to look up each option.

The `sq` command has built-in help text that can be accessed using the `help` command or the `--help` option:

```
sq help
sq --help
sq help key
sq key --help
```

```
sq help key generate
sq key generate --help
```

Use the help feature liberally to find out all the subcommands and options, and whether an option is a flag or takes a value, and what other arguments the command accepts.

Chapter 5

Managing one's own key

This chapter concentrates on creating and managing a private key for oneself. Please see the glossary for definitions of terms. Some of the terminology `sq` uses is specific to cryptography in general, or to OpenPGP, and some is specific to `sq` itself.

5.1 Why use keys and certificates?

Your key is, in the context of public key cryptography, you. It's a digital artifact that represents you to everyone else. Nobody else has your key. You and your key are inseparable, and to everyone else, your key is you, and you are your key.

That is, of course, romantic balderdash. A key is a large random number. It has no free will, it has no agency, it can't think, it doesn't feel, it can't act, it can't enjoy a cup of hot tea in the morning while writing a book, it's not alive. In no real way is it you. Except when it comes to secure communication, your key stands for you. When someone wants to send a confidential message to you, they encrypt it using your certificate, which is mathematically, inalienably linked to your key. When you want to prove a message comes from you, you encrypt it with your key.

You can have as many keys as you want. You can have one for work, another for school, a third for your family and friends, and a fourth one for publishing poetry online. These keys may be linked or kept separate, as you prefer.

5.2 Types of keys and algorithms

Over time, as cryptographic attacks have weakened the protections of cryptographic defences, different types of keys and algorithms have been developed

and enhanced. While a through discussion of these is beyond the scope of this guide, the list below gives a summary.

- Encryption algorithm **RSA**: this is the oldest known public key encryption algorithm. It has stayed strong, except as attackers are gaining faster computers and better attack software, the keys used for RSA have needed to become much longer. Where a 384 bit key was OK in the early 1990s, a 2048 bit or even a 4096 bit key is preferred thirty years later. The longer an RSA key is, the more computing power it takes to use it, and this makes strong cryptography with RSA slower.
- **Elliptic curve cryptography**: where RSA relies on the computational difficulty of factoring large numbers, elliptic curve relies on certain operations in geometry involving elliptical curves in two-dimensional space being hard to undo. There are several curves in popular use, and some of the older ones are now considered to provide only weak protection against attackers. New curves get developed from time to time, to provide more strength. With elliptic curves, the choice of curve also specifies the size of the key.
- **Hash** algorithms are one-way mathematical functions, where if you have a file you can compute a result from its contents easily, but if you have the result, it's really hard to get back the file. A *cryptographic hash* is one where it's really hard to find any file with the given hash value. As with encryption algorithms, hashes need to be made stronger as time goes by. OpenPGP originally mandated the use of the MD5 algorithm, but that is no longer suitable. Currently SHA256 is the preferred hash algorithm.

Because key types and algorithms need to be improved over time, the OpenPGP standard allows replacing them in newer versions of the standard without fundamentally changing the structure of the OpenPGP protocol. Each version of OpenPGP supports multiple key types and algorithms, which allows for a managed migration towards stronger security, and without losing access to older files and messages.

5.3 Why use subkeys?

Even someone having only one cryptographic key may benefit from having other keys for specific purposes. For example, they might have a very strong primary key as their primary key, and additional, auxiliary keys for encryption or digital signatures. Such auxiliary keys can be tied to the primary key using *certifications*, which we'll cover in more detail later. For now, a certification uses the primary key to declare that the auxiliary key can be used instead of the primary key for a specific purpose. The auxiliary key then becomes a *subkey*, and other users of OpenPGP will use it automatically, if they have your certificate. This setup has several benefits:

- you can have separate subkeys for encryption, signing, or authentication
- it's harder to leak or misuse the primary key, as it's only used rarely
- you can use a smaller key when less security is OK in exchange for faster use
- you can have a separate subkey for each device you have, or put them on a hardware security token
- you can replace the subkeys easily: others trust your certificate, and will happily use any subkey they can verify using your certificate

All of this is managed pretty much automatically using OpenPGP software.

5.4 Why would keys expire automatically?

A key, whether a primary key or a subkey, can be set to expire at a given time. This is a precaution against you losing access to the primary key: if the key expires, others won't use it anymore. You can extend the expiration as often as you wish, although that requires getting your updated certificate to everyone who needs to use it.

Another, more subtle benefit of expiring keys is that a short expiration time (of, say, one year) forces everyone else to refresh their copy of your certificate. This routine means they will also get a revocation update for the key, if there's ever a need for that.

You can also set subkeys to expire. This has the same benefits as expiring the primary key.

Changing expiration times can be a chore. There's a security benefit to it, but if it's inconvenient for you, you may want to consider not expiring keys, or only expire subkeys. Despite the benefits, it's better to have a non-expiring key than not have a key at all.

5.5 Generating a key

To generate a key with `sq`:

```
sq key generate --userid="My Name" --userid="<me@example.com>" --export=key.pgp
```

A key can have any number of *user identifiers* (or *user ids*). The Sequoia project suggests that it's best to have separate user ids for name and email address to allow them to be certified separately (we'll discuss what that means later). Traditionally they have been combined into one id, and that still works.

When a email program looks up a certificate for a recipient, it uses the email address to do so. At least one user id should contain the email address for the lookup to work.

You can set an expiration time at the time of creating a key, if you want. See the `--expires` and `--expires-in` options.

Generating a key with `sq` results in two files. The key is put in the file you name as the argument to the `--export` option. A *key revocation certificate* is written to a file with that name and `.rev` appended to it (or you can specify the name yourself, with the `--rev-cert` option). The revocation certificate tells others that your key is longer usable. If, for example, you lose the file with the key, you can share the revocation certificate with others, and they (or their OpenPGP software) will know to not use that key anymore. We'll cover key revocation in more detail later.

You can choose the cryptographic algorithm, and whether the key should have subkeys for signing or encrypting messages. See the `--help` output for a list of options.

5.6 Extracting a certificate from a key

Given a key, you can extract the certificate linked to it:

```
sq key extract-cert --output=cert.pgp key.pgp
```

The `cert.pgp` file is the certificate (choose whatever name you want for it). You need to re-extract the certificate every time you make a change to the key that would shared with others: user ids, expiration times, subkeys.

Note that while you can extract a certificate from a key, the other direction is not possible.

5.7 Sharing your certificate with others

A certificate contains no secrets, and you can safely share it with anyone: include it as an attachment in every email you send; put it on your web home page; put it on your profile on social media sites such as Facebook, Twitter, Mastodon, or GitHub; publish a photo of it on a photo sharing site; print it on business cards. We'll cover more options later in the chapter on managing keys in a community.

User ids are tied to the primary key, subkeys inherit them from their primary.

A certificate should only contain User IDs for identities that you want linked together. If you want to compartmentalize your online identities, then you should use a separate certificate for each set of pseudonyms, which should be separate from the others. For instance, you might have one certificate for your activities as an activist, and another for your normal, day-to-day activities.

Chapter 6

Using digital signatures

6.1 Why use signatures?

Digital signatures are used to show who sent a message and that it hasn't been changed. See the chapter on cryptography for a longer discussion.

It's important to note that signatures are good not just for messages, but for any kind of data, including files and cryptographic keys.

- You can sign a file to prove that you've seen it. Others can then verify that the signature and file match, and trust that the file they have is the same one you signed.

  This does not prove that you created the file, only that you had the file at the time you signed it.

- When a software distributor, such as Debian, Fedora, Red Hat, or the Apple app store, prepares a software package for distribution, they sign it using the their key. When a system downloads a package from the distributor, it verifies the package signature using the distributor's public key, which it already has, as it comes pre-installed in Linux distributions and on Apple devices. If the package fails to match its signature, the package won't be installed.

  This is done to protect the system from installing software that has been modified maliciously. It can also be used as a mechanism for preventing software from being installed that isn't approved by the distributor. Linux distributions don't do that, but big corporations seeking a dominant market position tend to like it.

- When you add a User ID to a key, any user ids attached to it are signed by the key. This prevents other people from adding their email address to

your key. If they could do that, they would get your email. While they can't read it, if it's encrypted, they can at least make sure you don't get it.

- When you add subkeys, they are signed by the primary key to prove that you, the key holder, wants the subkey to be used.

6.2 Making a signature

To sign a file with `sq`, you need your key (not the certificate). The command to sign is:

```
sq sign --signer-key=key.pgp --output=foo.pgp foo.md
```

This signs the file `foo.md`, and writes the signed file to `foo.pgp`. That file contains both the contents of `foo.md` and a signature.

Having the signature be part of the file can be convenient, but it can also be inconvenient. Sometimes it's easier to have the signature separate from the data. That's called a *detached signature*. If nothing else it means that you don't have two copies of the data, which can be costly in terms of disk space. A detached signature is also handy if someone else already has a copy of the data and you just want to prove your copy is identical.

To make a detached signature:

```
sq sign --detached --signer-key=key.pgp --output=foo.sig foo.md
```

Note the `--detached` option. The signature, but none of the original data, is written to `foo.sig`. The detached signature is small, and its size does not depend on the size of the signed data. That's because the detached signature only contains the *cryptographic hash* of the original data.

6.3 Verifying a signature

Verifying the signature of a signed file is done like this:

```
sq verify --signer-cert=cert.pgp --output=checked.md foo.pgp
```

The output will say something like this:

```
Good signature from 84B292ABCE27285B
1 good signature.
```

The mysterious number `84B292ABCE27285B` is the key identifier of the key that made the signature. It should match the certificate you've provided.

If the signature doesn't verify correctly, the error message is clear (see #768):

```
thread 'main' panicked at 'It is an error to consume more than data returns: Custom { kind:
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
```

(That output is a bug. See issue #768. This part of the guide will be updated once the bug is fixed.)

A detached signature is verified in a similar fashion, but you need to be a little more verbose and give the names of both the data file and the signature file:

```
$ sq verify --detached --signer-cert=cert.pgp foo.sig foo.md
```

6.4 Authenticating a certificate

If you have a suspicious mind set, you may have spotted a glaring error in the above discussion of verifying digital signatures: you must have the certificate of the sender, and you need to be sure it's actually their certificate.

Sometimes that's easy. If you're downloading software to install using a "package manager" (Linux distributions), or the app installation program on a mobile phone, the certificate was pre-installed.

If you're downloading software directly from its publisher's website, they probably provide the certificate on their website as well. If you're downloading things over HTTPS (instead of the unencrypted, unprotected, un-lamented plain HTTP), you can *probably* trust the certificate. It's possible to spoof that, but it's not easy.

You may gain more trust in the certificate by verifying that the one you have is one that a lot of other people have, and have had a long time. You can do this by asking a lot of people. We'll return to this topic later in the chapter on managing keys at a community level.

Chapter 7

Using encryption

7.1 Encrypting a file

Encrypting things is how you keep your private data private. There's a large, global debate about privacy, and who deserves to keep what private, and how to still catch bad guys who harm others. We shan't get into that here.

To encrypt a file using `sq`:

```
sq encrypt --recipient-cert=cert.pgp --output=bar.pgp foo.md
```

This encrypts the file `foo.md`, using the certificate in `cert.pgp`, and writes the result into `bar.pgp`.

Note that the encryption is done only for the explicitly specified recipients. If you want to read the encrypted output later, you need to add yourself as a recipient.

The output file has a name different from the input file so that the filename, which is not encrypted, does not reveal anything about the contents to someone who happens to see it.

You can optionally also sign the data by adding the `--signer-key=key.pgp` option to the encryption command.

7.2 Decrypting a file

To decrypt an encrypted file:

```
sq decrypt --recipient-key=key.pgp --output=decrypted.md bar.pgp
```

The output is written to `decrypted.md`. If the encrypted data was also signed, and you add the `--signer-cert=cert.pgp` option, the decryption will check

the signature. If the signature fails to match, the data is not written into the output file to avoid anyone trusting an unauthenticated message.

Chapter 8

Managing digital keys and certificates on a community level

FIXME: This chapter will discuss how to manage keys and certificates among large groups of people. It will discuss how to build strong trust that a key belongs to a specific person or organization. It will discuss various ways of distributing certificates.

- how do I find someone's key?
- how do I check it's their key?
- how do I get updates to their key?
  - user ids
  - subkeys
  - expiration changes
  - revocation of primary key or subkey

Appendix: How to. . . ?

This appendix has task-oriented guides for achieving specific goals. Each how-to guide will explain every step, giving command line examples, but will not go into detail about what is happening or why. These how-to guides are aimed at people who need to achieve a specific goal, have some understanding of OpenPGP concepts, but don't currently care to understand deeply. As such, the how-to guides will repeat specifics that have been covered in the rest of the book.

8.1 How to verify that a downloaded file is the one its author made

8.2 How to sign a file to share with others

8.3 How to decrypt a message from someone else

8.4 How to encrypt a message for someone else

8.5 How to generate a key, with subkeys, and a certificate

8.6 How to distribute certificate to others

8.7 How to certify someone else's user id

Appendix: Switching from GnuPG to Sequoia-PGP

This appendix is aimed at people who already know how to use `gpg`, the command line tool from GnuPG that roughly corresponds to `sq`. It shows how to do specific tasks using either `gpg` or `sq`. It will

GnuPG stores keys and certificates in the `~/.gnupg` directory, or the directory named in the `GNUPGHOME` environment variable. They're not easily accessed directly as files, and are referred to via the user id or using a hexadecimal key identifier or key fingerprint. The set of keys and certificates in that directory is called a *keyring*. possibly be a comparison table, for easy review.

GnuPG typically outputs binary files. The `--armor` option tells it to write a textual representation. That representation is still not human-readable, but can be easier to transmit over various channels that expect text instead of binary data.

8.8 Generate a key and certificate

```
gpg --quick-gen-key "Tomjon <tomjon@example.com>"
```

8.9 Export certificate into a file

```
gpg --export --armor tomjon > tomjon.asc
```

8.10 Import a certificate into your keyring

```
gpg --import certificate.asc
```

8.11 List all certificates in your keyring

Either all keys, or keys with a user id that contains a string:

```
gpg --list-keys
gpg --list-keys tomjon
```

Output for one key looks something like:

```
pub   rsa4096 2015-03-01 [SC] [expires: 2025-01-10]
      DBE5439D97D8262664A1B01844E17740B8611E9C
uid           [ unknown] Lars Wirzenius <liw@liw.fi>
uid           [ unknown] Lars Wirzenius <liw@iki.fi>
uid           [ unknown] Lars Wirzenius <lwirzenius@wikimedia.org>
sub   rsa4096 2015-03-01 [S]
sub   rsa4096 2015-03-01 [E]
```

The first word of the line tells you what the line contains:

- `pub`—a public key (i.e., certificate)
- `sec`—a secret key (i.e., key, in `--list-secret-keys` output)
- `sec#`—a secret key is known to exist, but isn't actually in the keyring
- `uid`—a userid attached to the key
- `sub`—a subkey

The `unknown` tells you how much you've told GnuPG you trust that user id. There's also information about type and length of a key, what it's used for (signing, certifying, encrypting), and key fingerprint (`DBE5439D97D8262664A1B01844E17740B8611E9C` above). The fingerprint is the strongest way to refer to a key.

8.12 List all private keys in your keyring

```
gpg --list-secret-keys
```

8.13 Sign a file

Drop `--armor` for binary output. Output goes to `hello.txt.asc` with `--armor`, or `hello.txt.pgp` without.

```
gpg --sign --armor hello.txt
```

8.14 Check a file's signature

```
gpg --verify hello.txt.gpg
```

8.15 Sign a file—detached signature

Drop `--armor` for binary output. Output goes to `hello.txt.asc` with `--armor`, or `hello.txt.sig` without.

```
gpg --detach-sign --armor hello.txt
```

8.16 Check a file's detached signature

```
gpg --verify hello.txt.sig hello.txt
```

8.17 Encrypt a file

Output goes to `hello.txt.pgp` (with `--armor` to `hello.txt.asc`). The `--recipient` option can be shortened to `-r`. By default, this encrypts *only* for the explicitly named recipients, so if one wants to decrypt the file later oneself, one needs to remember to encrypt it for oneself.

```
gpg --encrypt --recipient liw -r tomjon hello.txt
gpg --encrypt --armor --recipient liw -r tomjon hello.txt
```

8.18 Decrypt a file

Output goes to the standard output unless `--output` is used. Note that GnuPG may output the cleartext, even if the signature fails.

```
gpg --decrypt hello.txt.gpg
```

Appendix: Glossary

This appendix explains all the specialist terminology related to OpenPGP and Sequoia-PGP. It includes both the terms Sequoia prefers (e.g., "certificate") and the older terminology for the same thing ("public key").

authenticate to verify the origin of a message, using a digital signature

certificate the public key in public key cryptography; meant to be distributed widely; *see* public key

certification assuring that the user id and key belong to a specific person, using a signature on the user id

cipher an encryption algorithm

cleartext data that has not been encrypted, even if it's not text

decrypt convert encrypted data into cleartext

encrypt convert cleartext data into a form nobody can read unless it was encrypted for them

identity a deep philosophial problem; for OpenPGP, the user id attached to one's key

key a key for encrypting and decrypting data; *see* private key, public key

key pair in public key cryptography, the two parts of a key; *see* private key, public key

phishing trying to trick humans to do something they shouldn't, often to get them to reveal information they shouldn't

private key the private part of a key in public key cryptography; this is meant to not be shared with anyone

public key the public part of a key in public key cryptography; *see* certificate

sign to encrypt data with one's private key, to be decrypted using one's certificate

signature the result of signing data; *see* sign

user id a name and/or email address attatched to a key; there can be many such user ids attached to one key

verify to check that signed data matches its signature

Appendix: References

- OpenPGP on Wikipedia
- PGP on Wikipedia
- Public key cryptography on Wikipedia
- RFC 4880, the Internet standard specification for OpenPGP
- Sequoia-PGP website

Appendix: Copyright license

Preamble

Creative Commons Corporation ("Creative Commons") is not a law firm and does not provide legal services or legal advice. Distribution of Creative Commons public licenses does not create a lawyer-client or other relationship. Creative Commons makes its licenses and related information available on an "as-is" basis. Creative Commons gives no warranties regarding its licenses, any material licensed under their terms and conditions, or any related information. Creative Commons disclaims all liability for damages resulting from their use to the fullest extent possible.

Using Creative Commons Public Licenses

Creative Commons public licenses provide a standard set of terms and conditions that creators and other rights holders may use to share original works of authorship and other material subject to copyright and certain other rights specified in the public license below. The following considerations are for informational purposes only, are not exhaustive, and do not form part of our licenses.

- **Considerations for licensors:** Our public licenses are intended for use by those authorized to give the public permission to use material in ways otherwise restricted by copyright and certain other rights. Our licenses are irrevocable. Licensors should read and understand the terms and conditions of the license they choose before applying it. Licensors should

also secure all rights necessary before applying our licenses so that the public can reuse the material as expected. Licensors should clearly mark any material not subject to the license. This includes other CC-licensed material, or material used under an exception or limitation to copyright. More considerations for licensors.

- **Considerations for the public:** By using one of our public licenses, a licensor grants the public permission to use the licensed material under specified terms and conditions. If the licensor's permission is not necessary for any reason–for example, because of any applicable exception or limitation to copyright–then that use is not regulated by the license. Our licenses grant only permissions under copyright and certain other rights that a licensor has authority to grant. Use of the licensed material may still be restricted for other reasons, including because others have copyright or other rights in the material. A licensor may make special requests, such as asking that all changes be marked or described. Although not required by our licenses, you are encouraged to respect those requests where reasonable. More considerations for the public.

Creative Commons Attribution-ShareAlike 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution-ShareAlike 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. **BY-SA Compatible License** means a license listed at creativecommons.org/compatiblelicenses, approved by Creative Commons as essentially the equivalent of this Public License.

d. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

e. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

f. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

g. **License Elements** means the license attributes listed in the name of a Creative Commons Public License. The License Elements of this Public License are Attribution and ShareAlike.

h. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

i. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

j. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

k. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

l. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

m. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

Section 2 – Scope.

a. ***License grant.***

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

   A. reproduce and Share the Licensed Material, in whole or in part; and

   B. produce, reproduce, and Share Adapted Material.

2. **Exceptions and Limitations.** For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. **Term.** The term of this Public License is specified in Section 6(a).

4. **Media and formats; technical modifications allowed.** The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. **Downstream recipients.**

   A. **Offer from the Licensor – Licensed Material.** Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

   B. **Additional offer from the Licensor – Adapted Material.** Every recipient of Adapted Material from You automatically receives an offer from the Licensor to exercise the Licensed Rights in the Adapted Material under the conditions of the Adapter's License You apply.

   C. **No downstream restrictions.** You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. **No endorsement.** Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. ***Other rights.***

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. ***Attribution.***

1. If You Share the Licensed Material (including in modified form), You must:

   A. retain the following if it is supplied by the Licensor with the Licensed Material:

      i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

      ii. a copyright notice;

      iii. a notice that refers to this Public License;

      iv. a notice that refers to the disclaimer of warranties;

      v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;

   B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.

3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

b. ***ShareAlike.***

In addition to the conditions in Section 3(a), if You Share Adapted Material You produce, the following conditions also apply.

1. The Adapter's License You apply must be a Creative Commons license with the same License Elements, this version or later, or a BY-SA Compatible License.

2. You must include the text of, or the URI or hyperlink to, the Adapter's License You apply. You may satisfy this condition in any reasonable manner based on the medium, means, and context in which You Share Adapted Material.

3. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, Adapted Material that restrict exercise of the rights granted under the Adapter's License You apply.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material, including for purposes of Section 3(b); and

c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include

other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

a. **Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.**

b. **To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.**

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

   1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or

   2. upon express reinstatement by the Licensor.

   For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the "Licensor." The text of the Creative Commons public licenses is dedicated to the public domain under the CC0 Public Domain Dedication. Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at creativecommons.org/policies, Creative Commons does not authorize the use of the trademark "Creative Commons" or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized

modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.

Creative Commons may be contacted at creativecommons.org.