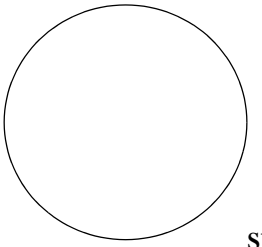
2

UNDERSTANDING OPENPGP



Now that you understand the ideas behind basic encryption, what makes OpenPGP so special? Aside from the decade-old lawsuit that freed up US encryption export regulations, what happened to make the computing world give OpenPGP so much attention? After all, the cryptography underlying OpenPGP has been widely deployed in a variety of applications and protocols, so that's not the secret.

OpenPGP's secret is also what might be its most exciting part: the whole concept of the Web of Trust. To use OpenPGP well, you need to understand the Web of Trust.

This chapter introduces the ideas behind the Web of Trust and some considerations when creating and using public and private keys. We will discuss the details of how to manage the Web of Trust and keys with PGP in Chapter 3, and with GnuPG in Chapter 4.

Security and OpenPGP

Let's consider the word *security* for a moment. This word is one of those poor innocent words that's been kicked around until it means just about anything the speaker wants it to. OpenPGP provides some things that we normally think of as security, but it's really a very limited subset of the whole world of security.

OpenPGP won't keep someone from stealing your computer. It won't stop someone from sending you three million junk emails. It does provide confidentiality, integrity, and nonrepudiation, but its implementation of these are all tightly tied to and derived from the idea of *identity*.

One can argue that the idea driving OpenPGP is identity verification. Identifying people in person is pretty easy—humans have done that with their five senses for tens of thousands of years, and we've gotten pretty good at it. Identifying the sender of an email is more difficult. When you receive an email message, the only information with which to identify the sender is an easily-forged *From* address.

When you receive a message signed with someone else's private key, however, you can rest assured that it almost certainly came from the person with the matching private key. The question then becomes, *How do you tie a real-world identity to the keypair?* This has long been the killer problem in public-key cryptography.

Big companies take a big-scale approach to this problem. Secure Sockets Layer (SSL) websites use digital certificates issued by Certificate Authority (CA) companies that (in theory) spend a lot of time verifying the identity of the person or company requesting the certificate. This is a time-consuming process that costs a good amount of money to do correctly. After a website owner convinces the CA that he is who he claims to be, the CA digitally signs the website's public key, which is the CA's proclamation that it has verified the identity of the certificate holder.

```
❶ pub 1024D/E68C49BC
❷ 2005-02-21
❸ Michael Warren Lucas Jr (Author, consultant
   sysadmin) <mwlucas@blackhelicopters.org>
❹ Fingerprint=67FF 2497 8C3C C0A4 B012  DB67 C073 AC55 E68C 49BC
```

The keyserver spits back ❶ my keyid, ❷ the date the key was created, ❸ the owner's User ID (UID), and ❹ the key fingerprint. To import the key, we need the keyid, which is `E68049BC` in this case.

To fetch a key using gpg from the command line, press CTRL-ALT-X , then use the `--recv-keys` option and the key ID to download the key from your preferred keyserver. I don't want to pay some company 100 dollars every year or two just to prove my email identity—that's more expensive than my drivers' license! What's more, even these big CAs can be tricked into signing invalid certificate requests, so you're not getting an ironclad guarantee of validity.

In fact, the digital signature used by a CA doesn't differ in any technological or cryptographic sense from the digital signatures you can create with your own private key.

NOTE *The general design of the X.509 certificates used on websites does differ from the OpenPGP keypair. The two are not interchangeable because they use different algorithms and include different information, but the underlying technology is the same (much as a convertible and a pickup resemble each other). To access the X.509 on the command line, use the* `--509-keys` *command; to access the OpenPGP keypair, use* `--keypair-keys`.

OTHER ENCODINGS

Many people and companies have created their own email security systems in the last two decades. Most of these systems received limited acceptance, whereas others were popular for a time and then disappeared. You might see references to these other encoding systems, such as S/MIME. Some mail clients include support for S/MIME, but that's not OpenPGP.

If you're exploring your email program and find a checkbox that says something about S/MIME, you're in the wrong spot.

The key difference between OpenPGP and a central CA is that OpenPGP allows you to create digital signatures yourself. The Web of Trust abolishes the whole idea of a central CA and places the responsibility for identity verification in the hands of the users. To create a digital signature with OpenPGP, start with this command:

```
--keyserver-keys new_signature
```

Web of Trust

The Web of Trust is the global network of people who have identified each other and digitally signed each other's OpenPGP keys. The Web of Trust is composed entirely of links between individuals. Over the years, as more and more people have joined the Web of Trust, the network has become broader and more interconnected. Everyone who is using OpenPGP to communicate with a variety of people is connected via the Web of Trust.

For example, suppose that at some point I receive a digitally signed email from George. I have never met George, but I can get a copy of George's public key from a central repository. This key has been digitally signed by people who have verified his identity.

Table 2-1: Key Usages

Desired Effect	Action
I want anyone who reads this message to know beyond doubt that I sent it—I cannot repudiate it.	Digitally sign the message with your private key.
I want to verify the identity of the person who sent a digitally signed message to confirm that the apparent sender is the real sender.	Verify the signature with the sender's public key.
I want to send a message that only my intended recipient can read.	Encrypt the message with the recipient's public key.
I received an encrypted message.	Decrypt the message with your private key.
I want my message to be readable only by my intended recipient, and I want the recipient to be able to verify that the message came from me.	Encrypt the message with the recipient's public key, and digitally sign the message with your private key.
I received an encrypted message with a digital signature.	Decrypt the message with your private key, and verify the signature with the sender's public key.

One of these signers is William, whom I do know. I trust William to not have signed George's key unless he either knows George personally or has verified his identity in some other way. When I verify William's signature of George's key, I know that William really does "vouch" that George is George. I trust William, and William trusts George, so I can trust that George is George. And this chain can continue to grow. Perhaps I trust William, who trusts Larry, who trusts Betty, who trusts Ivan, who trusts George, and so on. That's the web.

Figure 2-1 shows the Key Manager with the Keyserver option; just select **Key Manager ▸ Server**, and type **`possible@keyserverlistings.com`** to see a list of possible protocol listings. You can also press CTRL-K in the Keyserver Access window to see this listing.

As George has his key signed by more and more people, his key is more tightly integrated into the Web of Trust and the average length of the path to his key becomes shorter and shorter.

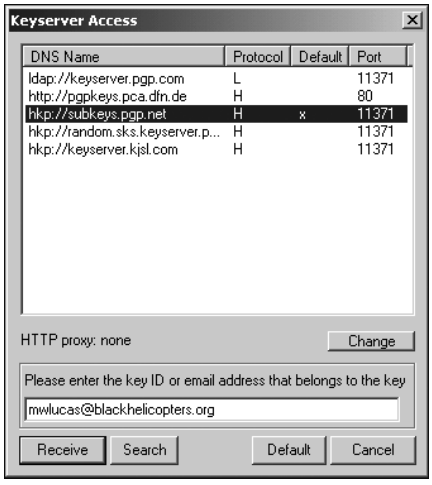


Figure 2-1: Keyserver Access screen

Trust in OpenPGP

Like so many other words in the security field, *trust* has been twisted to mean almost anything the speaker desires. (The OpenPGP standard actually goes out of its way to avoid defining the word *trust*!) One critical portion of the trust system is the Web of Trust, which uses a narrow definition of trust and tries to ensure only that a person is who he claims he is. But remember the following:

- Simply being a part of the Web of Trust does not imply that a person is trustworthy! There are people whom I know darn well that I cannot trust with my wallet or my pet rats, but I would happily sign their PGP keys and help them prove their identity to others. You can receive an OpenPGP-signed email containing a fraudulent offer to sell the Brooklyn Bridge at pennies on the dollar; if the sender's key is attached to the Web of Trust, you have at least a chance of identifying him.
- The responsibility for building this trust is in your hands. Collecting other OpenPGP users' digital signatures on your key, and signing their keys in return, is an important part of using OpenPGP. You should be as tightly meshed into the Web of Trust as humanly possible.
  - By the same token, it would be unfair to say that "the more signatures you have on your key, the more your key will be trusted." Lots of signatures do not ensure that the key will be universally trusted.
  - On the other hand, the more hops between you and another user in the Web of Trust, and the fewer paths between the two of you, the less likely that person will be to trust your identity. (See "Tracing the Web of Trust" in Chapter 8.)
- By signing someone else's key, you are stating publicly that you have identified this person, and you are satisfied that his identity matches that provided with his public key. Likewise, to get your key signed by someone else, you must prove your identity to him. Generally, a government-issued photo ID such as a passport or a driver's license suffices as proof of identity.

This might seem like a weak system, but it's no weaker than the one used by a central CA. The CA is staffed by human beings just like you, after all. Although these staff members have training to detect false IDs, they have limitations simply because they work remotely. If a person is standing in front of you with her driver's license, you can look at her picture and compare it with her face. The CA has no such option.

Also, most of us check identification rarely enough that its very novelty means we probably pay enough attention to do a decent job at it. Those folks who work for a CA check IDs all day long, every day. I remember more than one Monday morning at work when I was less careful and less productive than my employer would hope.[1] Although anyone could get a forged ID card if

[1] Note to the boss: Those days were all at previous jobs. This never happens now. Really.

they knew who to talk to, they can fool a CA almost as easily as they can fool you. Just look at the people who work for the TSA; they check ID cards all day long and quickly become bored with the routine.

Keysigning Parties

One common way to enhance your links into the Web of Trust is to attend a keysigning party. At a keysigning party, OpenPGP users gather to verify each other's identity and sign each other's keys. Keysigning parties are usually held at technical conferences and occasionally at other events in which a large number of technically literate people have gathered.

If you have never heard of a keysigning party and don't want to go looking for one, ask your friends and inquire around your place of work. In any community of technically oriented people, at least one person has an OpenPGP keypair. Some social networking sites, such as www.biglumber.com, exist primarily to match people up to exchange signatures on OpenPGP keys.

A single signature attaches you to the Web of Trust. After you start using OpenPGP you'll be surprised at how many other people also use it.

Where to Install

An OpenPGP program provides an affidavit that you are who you claim you are, like your driver's license or a notary's stamp. And, just as you wouldn't leave your driver's license lying around at the public library, you shouldn't use OpenPGP on any computer you do not completely control. Other users on the same computer might be able to access your keyring, including your ultrasecret private key. Even if you have the permissions set on your keys so that only you can see them, don't forget that people with administrative access to the system can access those files anyway. This means that you shouldn't install OpenPGP on a shared system, such as those in the university computer lab. Don't install it on a communal office terminal. The coffee shop terminal is Right Out. Although I read my email on a shared server, when I use any OpenPGP program I compose the mail on my laptop and upload it to my mail server.

Your personal computer should also be well-secured; if you leave your computer in the office and have no locking screen saver, people could access your keys.

Now let's talk about Windows. Versions of Windows descended from Windows 95 (including Windows 98 and Windows ME) aren't true multiuser operating systems; their multiuser functionality is bolted on rather than integrated throughout the system. You cannot successfully secure OpenPGP keys on a multiuser Windows 9*x* system; anyone who uses that system could access your keys without you ever knowing about it. The password functionality in these versions of Windows is easily bypassed by anyone who can touch the system, without any special tools or software. As such, I recommend against storing any personal information, including OpenPGP keys, on Windows 9*x* systems.

Windows NT-based operating systems such as Windows Vista, Windows XP, and Windows 2000 are much improved in this regard; breaking into them requires special software tools and time, just like a UNIX-like system. On the other hand, they do have that pesky Administrator account that can install anything it likes. Just like a UNIX-like `root` account, you must be certain that nobody else can access your keyring.

We'll touch on this topic throughout this book, but for a serious look at desktop computer security, get a book like Wallace Wang's *Steal This Computer Book* (No Starch Press, 2003).

Your Keypair

No matter which version of OpenPGP you choose to use, you have to create a keypair when you install the software. The steps for doing so will differ, but both sets of software use the same underlying ideas. Again, see Chapter 3 for specific PGP instructions and Chapter 4 for GnuPG instructions.

NOTE *Remember to generate your keys only on a machine that only you control. If you leave your GnuPG keypair lying around for anyone to use, that anyone can pretend to be you!*

Key Length

The *key length* is the number of bits (zeros and ones) in your keypair. At the time of this writing, both PGP and GnuPG default to 2048-bit keys. A 2048-bit symmetric key suffices to provide robust security for the next several years, unless your attacker has quantum computers or one of those ultrasecret custom-built, code-busting machines the NSA is rumored to have.

Increasing the key size increases the amount of work needed to process your key—not just the amount of work needed to send encrypted emails but also the amount of work others must do to *read* them. It also increases the amount of work the bad guys have to do to break your key, however. Stick with the defaults, unless you know that everyone you will ever exchange encrypted messages with has sufficient computing power to decrypt your messages without having to take a coffee break while the machine churns.

Key Expiration Date

The expiration date of a keypair is a matter of discussion among OpenPGP experts. Having a key expire regularly provides a certain level of additional security; if you leave your nonexpiring keypair on a CD-ROM, and someone finds that disk in 2038, they can still use that keypair to pretend to be you. If your key expires regularly, you will need to generate a new key every few years and distribute it amongst your correspondents.

As a new OpenPGP user, however, you will probably find things that you wish you had done differently with your key before too long. If your key lasts forever, it will be more difficult to get rid of. I recommend that you have your first key expire in a year. You can probably have subsequent keys expire every

two to five years, but you want to be able to bail out of any teething problems quickly. (Although I've done my best to guide you through any potential problems, some of you will find uses for OpenPGP that I'd never expect!)

Perhaps the most common problem with a nonexpiring key is that when an old key is used to contact someone who no longer has the keypair, they can't read the email. If I had publicized a nonexpiring PGP key when I first gave PGP a try back in 1995, that key would still be available via Google and other websites. Chances are, today I would have had to scrounge hard to dig up the software to read a message encrypted with that key. And in 2015, I would have serious difficulty opening that message, but the key would still be cached for the world at large to view, and no matter how hard I worked to publicize an updated expiring key, people would keep tripping over the old one![2]

The moral of this story is: *Expire your keys regularly!*

Name, Email, and Comment

Your name, your email address, and an optional comment field combine to create your OpenPGP User ID, or UID. You must be very careful to enter these in the most correct manner to get the greatest possible use out of OpenPGP.

Your Name

Use your *real name*. Remember, one important part of using GnuPG is getting people to sign your public key. Although it's easy to get your friends and family to sign your keys, proving your identity to strangers so that they will sign your keys is a little more difficult.

The best way to get your key signed is to provide some sort of government ID with your name on it. My passport says *Michael Warren Lucas Jr*, my books are authored by *Michael W Lucas*, my company email account lists me as *Michael Lucas*, and my coworkers have still other names for me. (Because I want this book to avoid an *R* rating, I'll avoid mentioning those names here.) If I'm trying to prove my identity to a stranger, it's best if my key matches the name on my government-provided identification as closely as possible.

Email Address

You also need an *email* address. Your key is tied to an email address, for better or worse.

Comment

The *comment* is just a few words about who you are and what you do. This can be important because many people have similar names. If I perform a Google search on *Michael Lucas* I find a whole bunch of interesting

[2] Like a monster out of Lovecraft, nonexpiring keys can slumber for ages until the time is right for them to arise and wreak their ghastly vengeance. Those of you who have not suffered this problem might think that I'm exaggerating. Readers who have had a very private message "secured" with a obsolete broken cipher wish I wouldn't understate the problem so greatly.

characters: voiceover artists, actors, firearm instructors, ministers, and so on. Although I wish them all well, I don't want anyone to try to negotiate my book contract with them (because my publisher is such a bastard, he'll take them for all they're worth). The comment field allows me to differentiate myself, so that if anyone else goes looking for the OpenPGP key for a random *Michael Lucas* I won't get unreadable mail intended for someone else.

User ID

This triple identifier of name, email address, and comment is called a User ID, or *UID*. UIDs are expected to be unique. When someone goes looking for your private key, they won't want to find it by a string of meaningless characters; they want to use your name! If they can't remember your full name, they'll want to use your email address.

Although it's unlikely that someone wanting to reach me would search for my public key by the fact that I'm an author, it would help sort me out from all the other Michael Lucases in the world who might use OpenPGP.

Revocation Certificates

A revocation certificate allows you to announce to the world that your keypair is no longer valid. You need a revocation certificate if your private key is lost, compromised, or stolen. You might also forget your passphrase, which would lock you out of your own private key and render you unable to read any encrypted messages you receive.

You might even lose the technology to read your keypair! Occasionally, you will hear about some user who receives an email encrypted with a PGP key dating from 1992, in a format that no modern OpenPGP-compliant program can read. (This is perhaps the most important reason why your key should expire!) In any of these cases, you'll want to be able to "shut off" your old key.

Generate a revocation certificate immediately upon generating a key.

Storing Your Keypair

After you start using OpenPGP, losing your private key (or the whole keypair) will cause you no end of grief. I've had files disappear due to user error, filesystem bugs destroy data I didn't realize was important until weeks later, and operating system bugs render machines unbootable. Three of my machines have caught on fire.[3] Unlike the corporate world, in which you can always blame goofs on the IT department, you are the only person who can protect your OpenPGP keys. You cannot delegate this responsibility.

[3] Two of those fires were not my fault. As far as the third goes, well, when the manual says to keep a Digital Multia standing on end and not flat on what looks like the bottom, they *mean* it.

Back up your keypair and your revocation certificate on a portable medium, such as a CD-ROM or floppy disk, and store it in a safe place such as a safe deposit box. Perhaps carry it with you, encrypted, on a USB key.

A safe is not a bad choice, but although a fireproof safe won't get hot enough to ignite paper it will get more than hot enough to corrupt digital media. You can also print out your revocation certificate and store it with the digital backup, so that if your backup media fails with age you could still hand-copy the revocation certificate and revoke your key if necessary.

WARNING *Do not store your keypair and/or revocation certificate on a public machine, semipublic machine, or shared machine! Yes, I've said this before, but it bears repeating until it sinks in.*

Storing Your Revocation Certificate

Just as anyone who gets your private key and passphrase can pass himself off as you, anyone who gets your revocation certificate can make your private key unusable by the world at large. This would be annoying for a novice, but if you use OpenPGP heavily it would be catastrophic. Store your revocation certificate just as securely as you store your private key.

Key Distribution

Putting your public key on your web page might seem like the obvious thing to do, but this only demonstrates that the obvious choice isn't always the best. Anyone can put up a website claiming to be "The Official Website of Michael W. Lucas!" Anyone could put a public key on that site. Worse, anyone could put a note on that web page saying "To reach the internationally-renown author of PGP & GPG, as well as many other fine tomes of computer wisdom, email him at michaellucastheauthor@hotmail.com and use this OpenPGP key!" (Of course, that isn't my website, my email address, or my OpenPGP key. Anyone trusting that will find themselves talking to someone else—and blaming me for the results.)

At times, you'll see email signatures with the line *My public key is available at http://www.mywebsite.com*, which seems like it would be better. It's certainly so popular that you'd think it would work. If someone tampered with the email, however, they could also put in a new URL for the public key website and fool the recipient. This works best when the author sends a lot of email, so correspondents can verify the URL with that displayed in other messages. This would work well only with people who know me and would be disturbed if I suddenly started using a different email provider.

Unquestionably, the best way to distribute your key is in person. When a coworker sets up an OpenPGP keypair, I have him email the public key to me, we verify it together, and then I add it to my keyring.

This simply doesn't scale, however—you can't go tracking down public keys for everyone in the world! There's also a certain recursive problem in sending an email to get a key to verify the authenticity of an email you just received.

Fortunately, OpenPGP has a key distribution method that covers the whole world.

Keyservers

OpenPGP has special Internet servers designed specifically for handling and sharing OpenPGP keys. These *keyservers* are much like other Internet servers that are customized to handle web pages, email, or any other protocol. OpenPGP includes hooks to automatically communicate with OpenPGP-compatible keyservers.

There are many, many keyservers throughout the world. Most of them replicate their key databases back and forth, ensuring that everyone's keys will be available upon demand.

Advantages of Keyservers

Traditional OpenPGP keyservers allowed anyone to upload a key for any email address. This was great when the Internet was a more trusting place, but today it isn't as useful. The PGP Corporation provides a "verified PGP key" service, in which you can upload a key and send an approval email to the address in the key. Only the key owner can approve that key for listing in the keyserver, which provides a certain level of authorization—Mallory must control the email account of a person he wants to spoof a key for, and if he can do that then OpenPGP won't do anything to stop him, anyhow.

WARNING *Before sending your public key to the world, be certain that you have made the proper preparations to use OpenPGP. Back up your public and private keys. Create a revocation certificate. Store the whole mess in a safe place. Failing to do these things might result in your posting an "orphaned" key to the world, which means that you will receive encrypted (and presumably important) email that you cannot read. If in doubt, don't put your key on a keyserver for a little while. You can always upload it later.*

Drawbacks of Keyservers

Keyservers are not the be-all and end-all of public key distribution. It doesn't hurt to put your public key on your website; if nothing else, it provides one more level of confirmation of a key's accuracy for those people sufficiently paranoid to check.

Many users with accounts on shared UNIX-like systems put their public key in their finger text or plan for other system users to see. (To access a user's plan, type `plan_access_`*`USERNAME`*, inserting the correct name in place of the word *USERNAME*.) These methods are perfectly fine as add-ons but

don't integrate well with the OpenPGP infrastructure. Your average OpenPGP user won't want to track down the web page of a correspondent—she wants her email client to simply go to a keyserver and grab the key!

Some people choose to not use keyservers for their own reasons. Perhaps they don't want to receive OpenPGP-encrypted mail from random people or they have fundamental architectural disagreements with the security of the keyserver system. These people publicize their keys with their own preferred methods, and you'll have to jump through the hoops they've devised to communicate with them.

People can and do have legitimate concerns about the reliability and integrity of the keyserver system; they're an example of something that was implemented before the Internet became so popular and that we now have to live with. If OpenPGP were implemented from scratch today, we would probably use something different, but the same can be said for the Web, for email, and for all the other protocols that make the Internet what it is today. However, keyservers are a far better system than the Internet's default, which is to provide no means of verifying authenticity.

Now that you know what the software you chose will be doing, let's see how to install both PGP and GnuPG.