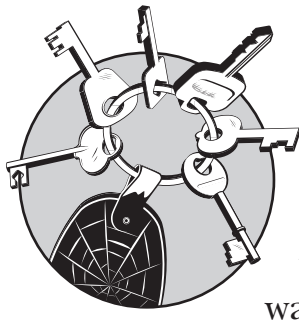
8

OPENPGP AND EMAIL



Learning how OpenPGP works and how to use it to manage your keyring has just been a warm-up to the real meat of this book: using OpenPGP with email.

Both PGP and GnuPG extend mail programs to include OpenPGP functions. As you'll soon see, OpenPGP operations in all mail programs are very similar after you learn where your particular client puts the "encrypt" and "sign" buttons. After you can use one OpenPGP program with a particular email client, you can extend that knowledge to cover other email clients.

Back in Chapter 1, we had a table that listed all the actions you could perform with OpenPGP. That was before you knew words such as *nonrepudiation*, however, and so it might not have meant as much to you as it does now. Take a look at it again here (Table 8-1) with your newfound cryptographic wisdom, and see how public keys, private keys, and digital signatures all

tie together to create a chosen level of privacy. After you begin working with OpenPGP on a daily basis, it won't take long to learn when to use the six features in Table 8-1.

Table 8-1: Key Usages

Desired Effect	Action
I want anyone who reads this message to know beyond a 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 see whether 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 want to decrypt a message that I received.	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 want to decrypt and verify a message that includes a digital signature.	Decrypt the message with your private key and verify the signature with the sender's public key.

Message Encoding

When used with email, OpenPGP uses two different methods to encode messages. PGP/MIME is the more modern choice, being attachment-based, but not all mail clients support it. The older *inline encoding*, also known as *clearsigning*, works well for basic email messages. The choice of encoding varies with different email clients and OpenPGP programs, and I'll discuss the options in the following sections.

Inline Encoding

Inline encoding occurs directly within the body of the email message. When you sign a message with inline encoding, the message body is edited to include an OpenPGP signature at the very end of the message. When encrypting and signing a message, the encrypted message replaces the original message body completely.

If you open an inline-encrypted message without using an OpenPGP program, it will start off looking much like the following:

```
-----BEGIN PGP MESSAGE-----
Version: PGP Desktop 9.0.2 (Build 2433)

qANQR1DBwU4D2jTKQaZxmacQCACbxrL+clBol8wB1R16tr5vXFFLurHsug9Qk6Cq
...
```

Not much to look at if you can't decode it.

Inline Encryption Trade-Offs

Inline encryption is about as simple as you can get, but as time has passed, its limitations have become more and more apparent. Inline encryption can have trouble with non-English character sets, attachments, and binary documents.

Non-English character sets (such as the symbols used for Chinese or Russian text) can cause problems for inline encryption. Email was designed by English speakers for English speakers, and mail programs already jump through hoops to manage non-English character sets.[1] Combining these characters with OpenPGP can cause unpredictable effects, depending on your combination of email client and OpenPGP software.

Similarly, attachments can be problematic with inline encryption. Your email client might encrypt your message body but leave the attachment unencrypted. To use inline encryption in such a case, you would need to encrypt your attachment separately and attach the signature and the encrypted attachment to the message. Also, when using inline encryption, you cannot encode binary data, such as PDFs, Microsoft Word documents, digital photos, and so on.

Finally, as if all this weren't enough, mail servers can corrupt clearsigned messages.

On the other hand, in spite of these challenges with inline encryption, OpenPGP-signed messages that use inline encoding can be read by any mail client.

When you use inline encryption, you must be aware of how your mail client interacts with your OpenPGP software in these circumstances. We discuss these interactions in the next two

[1] I like to think that if the original creators of email realized that it would be used by people all over the world instead of just a handful of highly skilled engineers, they would have anticipated these problems and designed around them. Then again, if the creators had realized what their innocent tool would become, they might have just given up the whole thing as a bad idea and bought extra stamps.

chapters. In a nutshell, PGP Corporation puts a lot of work into making inline encryption work properly and easily for everyone, whereas GnuPG varies with different clients.

PGP/MIME was designed to address all these issues and handle these cases without trouble.

PGP/MIME

If you're a computer geek, you've probably seen the expression *MIME type* before. MIME, or Multipurpose Internet Mail Extension, is a whole set of standards for encoding email. PGP/MIME is the encoding method designed for OpenPGP email. PGP/MIME gets around the problems with inline encryption by treating absolutely everything as an attachment: The encrypted message is sent as an attachment, the signed message and signatures are sent as attachments, and anything you attach is encrypted and attached.

Mail servers and mail clients treat attachments and email messages differently: Mail servers never modify attachments, and mail clients treat attachments as separate objects. Because attachments are left alone, PGP/MIME makes it much simpler to encrypt messages that use different character sets or binary files.

Generally speaking, all email clients, as well as OpenPGP implementations, can read both inline and PGP/MIME encoding unless otherwise noted in the program. PGP will handle each type of encoding, whereas GnuPG has limitations depending on the mail client you're using. Some mail clients work better with GnuPG than others.

OTHER ENCODINGS

Many people and companies have created their own email security systems in the last two decades. Most of these systems have 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.

Email Client Integration

You can integrate OpenPGP with your email client using either proxies or plug-ins.

Proxies

A *proxy* is a small program that runs on your computer and sits between your email client and your mail server. The proxy sends and receives email sent to your mail server, and the mail client sends and receives mail only through the proxy.

Proxies work with any mail client, but they are not as tightly integrated with the client as most people want; you configure signing, encryption, and decryption in the proxy program rather than in the mail client. Too, when using a proxy, you won't get an "encrypt and sign" button or menu option in your email client; instead, you'll have to open the proxy program and say "Encrypt all messages now" or "Encrypt messages to this email address." (PGP uses a proxy to handle messages, as I'll discuss in Chapter 9.)

Plug-Ins

The other option is *plug-ins*, which are used by GnuPG. A plug-in integrates with your email client, providing "sign" and "encrypt" buttons directly within the client.

Each mail client plug-in is unique, which means that a plug-in designed for Microsoft Outlook will not work in Mozilla Thunderbird.

Because plug-ins are written separately, each behaves slightly differently and has a different interface. Usually, the plug-in is written to look like it's part of the mail client program; integration is the whole point, after all! (I'll discuss GnuPG plug-ins for the three most popular Windows email clients in Chapter 10.)

Saving Email—Encrypted or Not?

Like many people, I save all my old email[2] because I find it very useful for reference. In fact, I've worked at companies in which saving every piece of email from your manager was the only way you could keep your job longer than a month.

OpenPGP presents some interesting problems when archiving mail, however. For example, when you send someone encrypted email, the reader must use the recipient's private key to read it. However, because you don't have the recipient's

[2] I have a complete archive of my email since 1985, minus only pieces that I have deliberately chosen to delete. I suspect that the mere existence of this archive will be submitted as evidence at my eventual sanity hearing. That or my publisher will publish it all. I mean, the guy published a book containing nothing but messages written *to* spammers! I can't imagine who let him out of his cage, let alone gave him a job.

private key, you can't read the mail that you sent, even though you created it!

Saving Unencrypted Email

Some email client plug-ins allow you to save mail as unencrypted. The problem with choosing this option is that it will protect your email during transit and while on the recipient's computer, but not on your hard drive. Anyone who can access your computer will be able to read those encrypted messages.

Although this option might be fine in a corporate environment, if you're in a totalitarian country being threatened by some ugly man with a rusty machete and serious anger issues, those messages just might mean life and death (for your correspondents, if not for you). (I wish this were a joke; in fact, OpenPGP has saved lives in exactly this situation.)

One popular option is to save all your email unencrypted, but on an encrypted disk partition. We discuss this briefly in Chapter 11.

Encrypt to Self

Another popular option is to also "encrypt to self" (in other words, to encrypt the saved email with your public key so that you can open it using your private key and passphrase). Using this option will stop people from getting the document even if your computer is stolen, and it is often a good middle ground for many people. (You can get the same effect in an email client that saves sent mail as encrypted by Cc-ing yourself when you send the original message.)

NOTE *If you're using a proxy program to provide OpenPGP services, the mail client will see only unencrypted emails. This means that your mail is always saved unencrypted.*

We'll discuss which options each piece of software provides (if any) in the next two chapters.

Email from Beyond Your Web of Trust

People across the world use OpenPGP, and you don't know all of them. Chances are that your keyring will start off populated with keys for friends and coworkers, and slowly grow as you communicate with more OpenPGP users. If you receive an encrypted email from a country on the far side of the world, however, it's quite possible that you will have nobody in common and hence you won't really be able to truly verify their identity. What do you do?

One possibility is to use only the corporate PGP keyserver and only correspond with people who use that keyserver. PGP Corporation's keyserver signs public keys after it verifies the email address they're attached to.

However, OpenPGP is called "open" because anyone can implement it, and you can't control who will send you email any more than you can control who sends you postcards. I correspond with people all over the world who use OpenPGP, and quite a few have public keys that aren't even vaguely hooked into my Web of Trust. How can I trust them? Here are my three choices:

- Expand my Web of Trust
- Trace the Web of Trust to that person
- Use the key but limit my trust of the sender

Expanding Your Web of Trust

The most correct answer is to expand your Web of Trust. Exchange signatures with more people, even people with whom you're not likely to exchange encrypted mail. More people than you suspect travel between companies, countries, continents, and cultures. Sign their keys and have them sign yours, which will embed you more deeply in the Web of Trust, making it easier for you to reach others and for others to reach you. This takes time, however, and if you receive a mysterious email you don't want to wait weeks or months to read it.

Tracing the Web of Trust

Search Google for "PGP pathfinder" and you'll find any number of websites in which you can trace the path through the Web of Trust between any two OpenPGP keys available on public keyservers. These sites use the keyid for the two keys involved (remember, the keyid is just the last eight characters of the fingerprint). The more paths that exist through different people, the more likely I am to trust that key. Having had my key signed at a couple of different keysigning parties, I would expect to have several paths to anyone in the Web of Trust.

For example, suppose that after publishing this book I get an email from someone who claims to be Phil Zimmermann, the original creator of PGP. The keyid of the message sender is B2D7795E. I can grab Phil Zimmermann's public key from a keyserver, or from his web page, but it's possible that someone uploaded a bogus key for him just to fool people like me.

I visit the Web of Trust pathfinder at www.cs.uu.nl/people/henkp/henkp/pgp/pathfinder (Google's first result) and enter the keyid of the message I received and my keyid. This server tells me that there are eight *disjunct* paths between this key and mine. In other words, my key is linked to the other key by eight different paths that have *no people whatsoever in common*. For that key to be fake, the faker would have had to fool a whole lot of people. Although I have never met Phil Zimmermann, I would believe that this key is legitimate. (If the only path had been through one of my incorrigible practical joker friends, or if there had only been one path, I would have been far more suspicious and infinitely less trusting.)

Most of these Web of Trust tracing programs are based on *wotsap*, a freely available Python program designed to trace relationships between keys. Wotsap is available at many Internet sites; if you're seriously interested in analyzing the Web of Trust, I suggest you start there.

Repeatable Anonymity

No matter what you do, one day you will receive an OpenPGP-encrypted message from someone you don't know, whose key you cannot verify, and who is not in the Web of Trust. What do you do?

Well, you have three choices. One, you can delete the message unread. Two, you can leave the message sitting around until you can verify the owner's identity. Three, you can install the sender's public key and read the mail, which is actually perfectly safe.

"Safe? How can it be safe, when you can't verify the identity of the sender?" Because installing any public key is harmless: An unknown key on your public keyring won't cause your machine to be insecure, can't leak your passphrase to the world, and won't allow Mallory to steal your bank records. (Yes, Mallory could find a "magic public key" that would expose a software bug in your OpenPGP implementation and use it to crack your machine wide open, but both PGP and GnuPG handle keys very cautiously and are audited for just such problems.)

Untrusted, unknown keys that aren't attached to your Web of Trust can actually be extremely useful for anonymous or pseudonymous communications. For example, when I began this book I asked for technical reviewers on various OpenPGP-related mailing lists. A person who regularly made valuable contributions to a GnuPG list under a fairly obvious

pseudonym offered to read the book. I have no idea who this person is in the real world. I have no idea of his credentials, other than his intelligent contributions to the list. I don't even know if he is really a he, she, it, or perhaps even a hyperintelligent mouse who is attempting to engage me in his latest overcomplicated plan to take over the world. I have this person's OpenPGP public key, however. When I receive an email signed or encrypted with that key, I know that the sender has the matching private key and passphrase. The author is pseudonymous, and yet I know that he's the same person every time. An impostor who wanted to assume this person's identity would have to get the passphrase and private key from the real person, and then be able to write email in the same style and with the same knowledge as the original person.

This feature is very useful for anyone who has to deal with anonymous sources on a regular basis or who wants to remain anonymous for the present but reserve the option of proving their identity later. For example, imagine a corporate whistleblower who wants to anonymously leak information. A string of OpenPGP-signed messages is a perfect way to do this. The whistleblower can distribute incriminating documents, all signed with OpenPGP. After the corporate giant is destroyed and the CEO carted off to jail for being really, really naughty, the whistleblower can prove that he was the document source by producing the private key and passphrase—or not.

The real question here is "How far do you trust?" Adding a key to your keyring doesn't mean you have to sign it or believe what the email's author writes; it only means you can read the sender's message and make up your own mind. If you decide that the person is full of baloney, you can remove the key from your keyring and have your email client delete further messages unread. If you decide to correspond with the person further, at least you know that subsequent messages are from the same person.

If you can use keys disconnected from the Web of Trust, does this make the Web of Trust less useful? Not at all; the Web of Trust is wonderful for verifying the identities of correspondents, especially when you use path tracing. The Web of Trust is a great way to verify that you are who you claim to be.

By the same token, the Web of Trust is not a straitjacket; using OpenPGP without the Web of Trust provides different possibilities. Without the Web of Trust, trying to prove someone's identity is like searching at midnight in a coal cellar for a black cat that isn't there. The Web of Trust gives you a net and a flashlight, or at least tells you that the cat might be a tiger.

Unprotected Email Components

OpenPGP goes through any number of hoops to protect the contents of the message, but you should remember what it doesn't protect. In particular, OpenPGP does not encrypt the subject lines in email. A subject such as "Ransom pickup at 8PM, City Park" doesn't leave many questions for anyone who intercepts your email. Email messages sent with PGP should have innocuous subjects (or perhaps no subject at all).

Also, your mail client might default to storing unencrypted versions of the OpenPGP emails that you send. Be sure that you know how your OpenPGP system stores messages. My preferred OpenPGP mail client (mutt) stores the sent messages in encrypted format, which means that I cannot read messages that I have sent unless I Cc myself. Some people find this irritating, and configure their mail clients to store sent messages in unencrypted format.

This really is a matter of what best suits your needs. If you are unconcerned about someone who gets access to your computer being able to read your sent mails, fine. If you are in a life-threatening situation, however, it is best to keep your sent messages encrypted, as I'll discuss in the next two chapters.

You now know enough to actually use OpenPGP and the Web of Trust in day-to-day work. Let's see how to configure email clients under both GnuPG and PGP.