## Encrypting Email with KMail, Mozilla Thunderbird, and Evolution

# LOCK AND KEY


The leading email applications include new features for helping users secure and authenticate their mail messages, but each tool has a different approach to handling tasks such as signing and encryption. This article describes how to add encryption and digital signatures to the Thunderbird, Kmail, and Evolution mail clients.

**BY FRAUKE OSTER**

Spoofers have an easier time on the Internet than anywhere else. There's no need to forge a signature to dispatch mail under someone else's name; all you need is a spoofed entry in the From header. The mail protocol does not provide any kind of protection against this kind of manipulation. If you want the people you write to to be able to rely on the authenticity of your messages, you should get into the habit of signing your email messages. The same thing applies to encryption: or would you really want a curious mail server admin reading your letters? Anyone with access to one of the computers that relays mail between your outbox and the destination could theoretically read your messages.

GNU Privacy Guard (GnuPG) [1] is a program that protects your messages against monitoring and manipulation. GnuPG is a cryptographic system that uses asymmetric keys. For the user, this means having two keys, a private key and a public key, which are generated as a key pair. The password-protected private key is kept secret. This is the key you use to decrypt and sign email messages.

In contrast to this, you need to distribute the public key to the people you write to. The public key allows people who write to you to encrypt any messages they send to you and to check your digital signature. When you sign a message, GnuPG uses your secret key to generate a hash for the message body and attaches the hash to the message. The recipient can use your public key to verify the authenticity of the message.

## Software You Need

You need two things for secure communications of this kind: the GnuPG software itself and a mail program that supports GnuPG. In this article, we will look into handling GnuPG with KMail, Thunderbird, and Evolution. *gpg --version* tells you if GnuPG is installed on your system, and if so, which version you have (Figure 1). If an error message is displayed instead, you will need to install GnuPG from the distribution media. The package is typically called gpg or gnupg; in Suse 9.1, the package is version 1.2.4.

As not all mail clients are capable of generating a key pair, your best option is

**Figure 1: Most mail programs rely on the gpg command-line tool for encryption and signature. *gpg -version* tells you which version is installed on your system.**

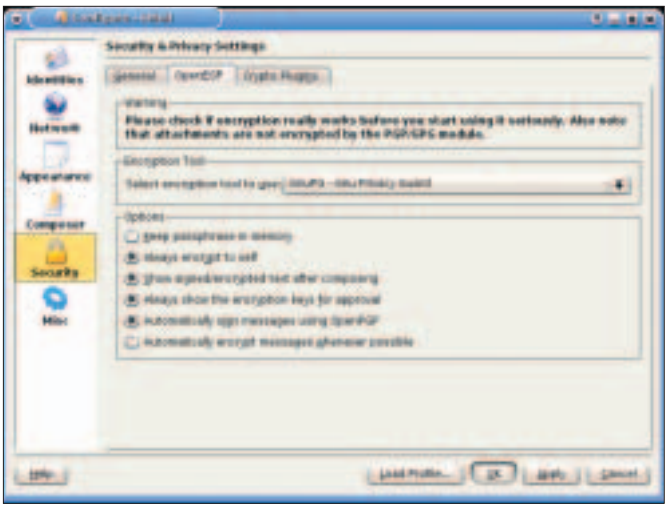

**Figure 2: Select GnuPG as the encryption program in the KMail settings dialog.**

to use the shell-based tool, which will run on any system. Type *gpg --gen-key* to launch into the key generation dialog. GnuPG will prompt you to specify the key type first. There are three options, but the default – ElGamal and DSA – is your best bet. Press [Enter] to confirm. You can then go on to specify the key length. The key length is always a trade-off between security and processing effort. A short key is easier to break, but it is processed more quickly. The default of 1024 bits should be fine for normal email communications. Press [Enter] again to accept the default.

GnuPG then goes on to prompt you for an expiration data for the key pair. If you intend to set up a large network of trust (see Box 2), you will not want the keys to expire too quickly; this would mean you having to send the new keys to the whole network, and the whole network signing your keys. If in doubt, don't specify an expiration date. In this case, you need a so-called revocation certificate to revoke the keys before the expiration date and remove them from the key servers. Your best option is to create a revocation certificate immediately after creating the key pair – type *gpg -output revoke.asc --gen-revoke key-ID* to do so – and store the certificate in a safe place for future use. Enter your email address as the key ID. After completing this selection, press [y] to confirm the expiration date.

The next step is to type your name, an optional comment, and a valid email address; this is the address you are creating the key pair for. Then press *F* for Finish to confirm your entries. Finally, GnuPG prompts you to enter a passphrase for the keys. As the term passphrase indicates, this does not mean a simple word, but rather a longer string of letters, numbers and special characters. GnuPG security relies on you choosing a good passphrase. If someone steals your private key, the passphrase is the only thing that will stop this person from decrypting your mail or signing messages in your name.

You will find the settings required to use GnuPG in KMail 1.6.2 [2] in *Settings | Configure KMail*. In the *Security* window *OpenPGP* tab, select *GnuPG – GNU Privacy Guard* (Figure 2). Then assign a GnuPG key to your email address in *Identities*.

Two new buttons are displayed in the Composer window. The feather symbol is used for signing, and the lock symbol for encrypting email messages. You can select *Attach Public Key* in the *Attach* menu to add your own or another user's public key to an email message.

If you receive a signed email message, KMail draws a frame around the message (Figure 3). Invalid signatures are framed in red; valid but untrusted signatures are yellow, and valid, trusted signatures are green. This tells you at a glance whether you can trust a message or not. KMail draws a blue frame around any email messages it decrypts.

KMail 1.6.2 has one major disadvantage: it uses inline encryption – that is, it encrypts the message content but not the attachments. Version 1.7 or newer of KMail adopts the OpenPGP/MIME stan-

## Box 1: What's New in KMail 1.7?

OpenPGP/MIME integration has added a few items to the KMail settings dialog in version 1.7 (which is supplied with KDE 3.3 ). There are now five tabs in *Settings | Configure KMail | Security*. The *Composer* tab has more options for signing and encrypting messages. These include options for automatically signing, and if possible encrypting, messages. KMail now has an option for encrypted storage of any encrypted messages you send; this stops unauthorized users from reading the messages on your disk, including any encrypted messages stored as drafts. If you prefer to keep control over what KMail does, you can tell KMail to prompt you to confirm the key that it selects.

In the *Warnings* tab (Figure 4), you can tell KMail to remind you to sign or encrypt email messages. If you do so, a security warning will be displayed whenever you attempt to send a message without a signature or encryption. By default KMail warns you if a key is getting close to the expiration date.

In contrast to KMail 1.6.2, the newer version no longer has an option for selecting an encryption program. Instead, GnuPG support has been added to the *Crypto backends* tab. In the *Reading* tab, you can tell KMail to automatically import public keys attached to incoming messages. This can save you a lot of effort if you tend to receive keys by email rather than from a key server.

You can still define your own keys in the *Security* tab in the *Identity* dialog. The *Crypto backends* tab has new options for selecting the encryption type; this is where you can opt to use OpenPGP/MIME or inline encryption. When composing a message, you can also select one of these methods from a pull-down menu. This feature is useful if you want to send a message to someone who uses a mail client without MIME support, such as KMail 1.6.

dard, which most other mail clients use. OpenPGP/MIME encrypts all parts of an email message, including the attachments, and sends them as individual MIME parts.

The older KMail versions cannot handle OpenPGP/MIME messages, such as the ones generated by most mailers. On the other hand, Evolution, for example, cannot decrypt inline encrypted messages. Thunderbird and the new KMail client support both methods. So it makes sense to update to KMail 1.7. Box 1 gives you an overview of the changes.

As an alternative, you can add OpenPGP/MIME support from the Aegypten Project [3] to KMail 1.6.2. Suse Linux has an OpenPGP/MIME plug-in package, but users of other distributions will need to build the plug-in from the source code. As this involves building and installing six additional packages, an update makes more sense.

**Figure 3: KMail draws a colored frame around signed and encrypted messages.**

## Thunderbird with Enigmail

Mozilla Thunderbird [4] needs the Enigmail [5] plug-in to support GnuPG. You need to download the plug-in off the Internet and install it via *Tools | Advanced*.

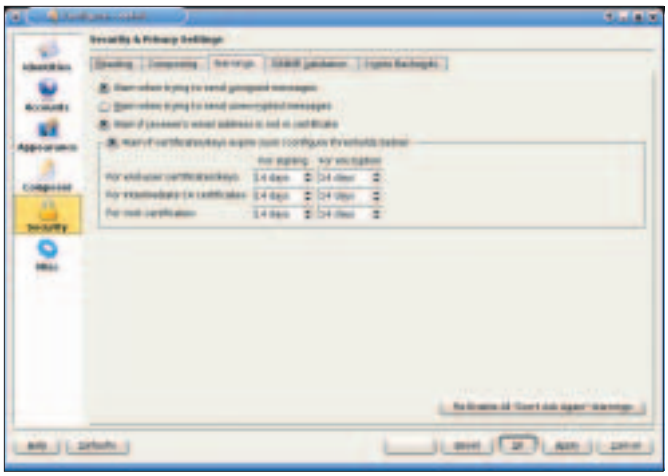

**Figure 4: KMail 1.7 can warn you if you attempt to send a message without encryption or a signature.**

Specify the path to the GnuPG program in *Enigmail | Preferences* (Figure 5) – this is */usr/bin/gpg* for most distributions – and use the defaults for the other fields.

You can use the new *OpenPGP Security* tab in *Tools | Account settings* to specify the keys that Enigmail should use. First, enable GnuPG support by checking *Enable OpenPGP support (Enigmail) for this identity* (Figure 6). If you use the same email address for the account and the key,

Thunderbird will automatically look for the matching key. If not, select the second option and choose the required key from a list of existing keys. Then go on to specify whether Thunderbird should automatically sign and encrypt messages.

When you compose a message, an *OpenPGP* button with a pull-down menu is displayed (Figure 7). You can use the

## Box 2: Distributing and Signing Keys

GnuPG communications require two parties. If you want to send an encrypted message to somebody, the recipient needs your key up front. It only makes sense to send a signed message if the recipient can use a key to check if the message and the signature really belong to you. In both cases, the key exchange is a major issue. After all, anyone could send you a key as a mail attachment using a spoofed email address, so this is no proof that the key actually belongs to the sender.

The key fingerprint allows you to verify the key. The fingerprint is a string of numbers and capital letters that validates the identity of the key. You can generate a fingerprint by entering *gpg --fingerprint key-ID*. Instead of the ID, you can alternatively provide the email address, assuming a one-to-one correlation between the key and the address.

If you are sent a key as an email attachment, or if you download a key off the Web, it makes sense to verify the owner using the fingerprint; this is a sure fire way of making certain you have the right key. You can either phone the key owner to do this or arrange to meet and exchange fingerprints. If the fingerprint matches, you can then type *gpg --import keyfile* to add the key to your keyring.

This is fine for people you know, but what happens if you need to communicate with someone you have never seen before? This is where key signatures are used to build a so-called Web of Trust .

If you have verified a key, you can sign the key. In other words, you vouch for the identity of the owner, and the correlation between the key and the owner, with your good name (and your key). You can then return the signed key to the owner. Another user who does not know the key owner, but knows you to be a trustworthy and responsible person, can base his or her decision to accept the key on the signature you attached to the owner's key.

The following command will sign a key:

```
gpg --sign-key key-ID
```

To remove the need to mail keys back and forth, a number of so-called key servers have been set up on the Internet, and you can download keys from these servers. Key servers synchronize key data and distribute public keys. In other words, any key server should have the key that you need.

The following command:

```
gpg --recv-keys key-ID
```

downloads the key with the specified ID from your preferred key server. And

```
gpg --send-keys
```

uploads your keys to the key server. You will be prompted to confirm that you want to do this, as the command sends all the public keys on your machine to the key server. This allows you to update any public keys belonging to other people that you have signed. If you want to update all the keys on your system, and to receive signatures from other users, the following command will do just that:

```
gpg --refresh-keys
```

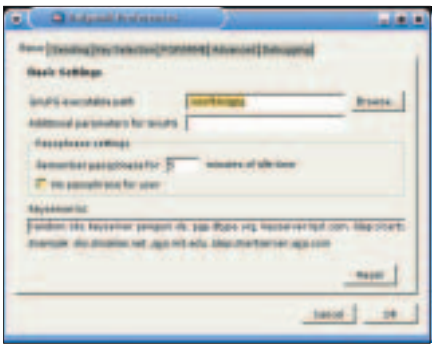

**Figure 5: Before you can use the Enigmail plug-in with Thunderbird, you need to specify the path to GnuPG.**

menu to sign or encrypt the messages. The Enigmail menu also allows you to attach your public key to the message for distribution to you mail correspondents.

If you receive an encrypted or signed email message, Enigmail adds a highlight to the message above the sender address, additionally displaying a pen for signed email messages and a key for encrypted messages. You can click the icon for more detailed information on the signature and encryption. This allows you to discover the origin of the message. Thunderbird can compose and read messages using both the inline method and OpenPGP/MIME.

The main window below *Enigmail | OpenPGP Key Management Window* gives you a key management tool with a few useful capabilities. For one thing, it can list the public keys on your system. You can then opt to sign the keys, create a new key, and add users to a key. The *Key* menu also allows you to create a revocation certificate (Figure 8). Unfortunately, Enigmail does not support key server management, although this feature is due to be implemented in future versions.

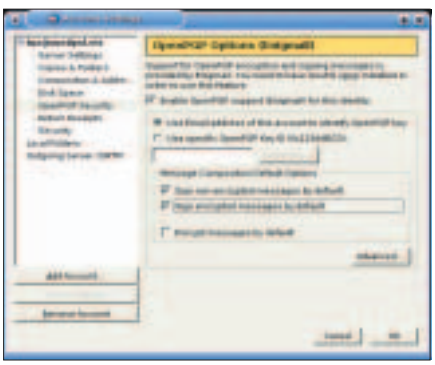

**Figure 6: Use the account settings in Thunderbird to enable Enigmail support for your mail account, and specify the keys that Enigmail should use.**

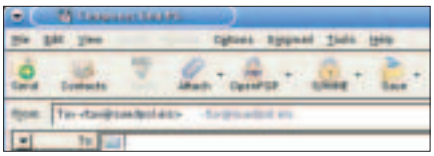

**Figure 7: When you install Enigmail, a new *OpenPGP* button is displayed in the compose box.**

## Evolution: Simple and Secure

To use GnuPG with Evolution 2.0.2 [6], open the settings dialog via the *Tools* menu, and then select the email account for which you will be defining a GnuPG key. Now click *Edit* to open a new dialog where you can enter the key ID in the *Security* tab (Figure 9). To display the ID, type *gpg --list-keys your@mailaddress* in a terminal window. GnuPG will display the key type (e.g., *pub* for a public key), the key length and type (e.g. *1024D* for a 1024 bit DSA key), the required ID, and the creation and expiration dates. Evolution also accepts the email address to which the key belongs instead of the ID.

You can use the same dialog to opt to sign all outgoing messages, to never sign appointment queries, and to encrypt email messages to yourself. The latter is extremely useful, as this additionally encrypts messages with your own key. In other words, you can decrypt the message yourself at a later date and reread the message if required. If you do not enable this option, you will be unable to open any messages you compose and encrypt yourself. You should also enable *Always trust keys in my keyring when encrypting*, as Evolution will otherwise reject unsigned keys.

When composing a new message (Figure 10), you can use the *Security* menu to specify whether you will be signing or encrypting the message. Before signing a message, Evolution prompts you for your pass phrase to ensure that you really are the owner of the key. This means that recipients of email messages can trust the identity of the sender.

If you opt to encrypt a message, Evolution automatically searches your system for the recipient's public key. This will not work if the email address for the key is not the target address. Evolution cancels the action at this point and displays an error message.

A lock icon is displayed at the bottom of encrypted messages. Clicking on the lock icon takes you to a dialog with encryption or signature details for the message. However, Evolution does not tell you anything about the key used to sign the message, and that makes it difficult to tell if the message was sent by a trusted person. Evolution does not support inline encryption and cannot open inline messages. In this case, you will need to take a detour via the command line to decrypt the message.

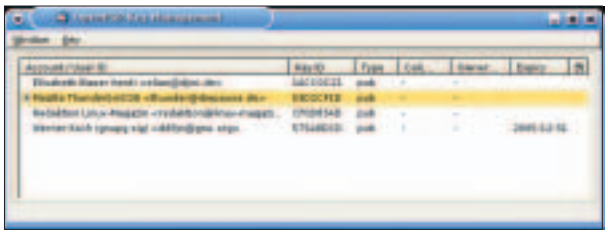

**Figure 8: Thunderbird with Enigmail is the only one of the three mailers to provide key management, meaning users do not need to resort to command-line programs.**

## Which Program to Choose

Although all three programs support GnuPG, the three programs are quite different. It is quite easy to configure Evolution 2.0.2 to encrypt your messages; this makes it useful for people who want to secure email messages quickly and

**Figure 9: Use the Evolution Account Editor to specify the keys that Evolution should use. Instead of the key ID that Evolution prompts you for, the program will also accept the email address of your key.**

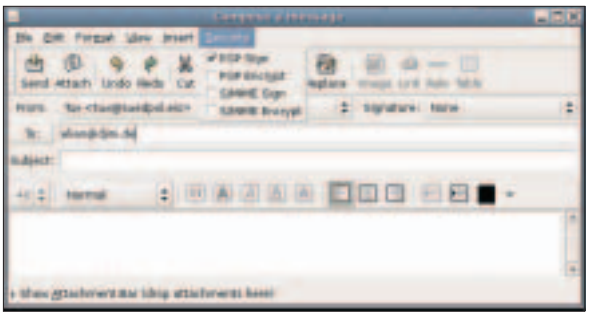

**Figure 10: Use the *Security* menu to specify if you will be encrypting or signing the message.**

with a minimum of effort. The program supports the modern OpenPGP/MIME standard; on the downside, it can not decrypt inline PGP. This means you have to store inline messages and decrypt them in the command line using the *gpg* tool – this is too much like hard work for most people.

Mozilla Thunderbird 0.9 does not support GnuPG by default, but you can easily install the Enigmail plug-in. Enigmail does not locate *gpg* automatically after the install; instead, it expects you to specify the path. The other settings are quite simple. One good thing about Thunderbird is that it supports both inline and OpenPGP/MIME encryption. Thunderbird scores extra points for its integrated key management, although key server management is still missing, as previously mentioned.

The biggest disadvantage with KMail Version 1.6.2 is the lack of support for OpenPGP/MIME; the only way to add OpenPGP/MIME support is to build the Aegypten project and install the plug-in. If this sounds too much like hard work, your only option is to update to version 1.7. A variety of KMail settings have been further extended in KMail 1.7. Highlighting of signed and encrypted email messages is a good thing, allowing users to identify signed and encrypted messages at a glance, instead of losing valuable time identifying trusted messages. ■


## THE AUTHOR

Frauke Oster studies applied computer sciences and has been involved with a number of KDE projects, including the KDE Women Project. When Frauke takes a break from hacking on her computer, and programming in C/C++/Qt/Java, she likes to go to the moves, the theater or museums on her leisure time.


## INFO


[1] GnuPG: *http://www.gnupg.org*

[2] KMail: *http://kmail.kde.org*

[3] Aegypten project: *http://www.gnupg.org/aegypten*

[4] Mozilla Thunderbird: *http://www.mozilla.org/products/thunderbird*

[5] Enigmail: *http://enigmail.mozdev.org/*, *http://www.thunderbird- mail.de/extensions/enigmail/enigmail.php*

[6] Evolution: *http://www.gnome.org/projects/evolution*