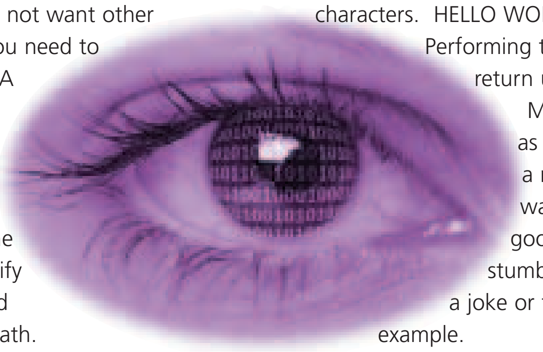
CRYPTOGRAPHY
Protection from prying eyes

BEWARE THE EYES OF MARCH

You may have something that you don't want every Tom, Dick or Sarah knowing about, so encrypt it. John Southern shows us how



If you consider your data to be sensitive – i.e. something that you may not want other people to view – then you need to think about encrypting it. A cryptosystem is a way of disguising a message so that only the intended recipients can view the true data. Only those in the know will be able to identify the false nose and wig and decrypt the message beneath.

Can you rely on encrypted text?

No matter how securely you encrypt your messages there is no absolute guarantee that no one, other than your intended recipient, will get to the information they contain. With a little brute force, enough processing power and a lot of time, anything is crackable. All you need to know is the encryption algorithm. Once someone gets hold of the encrypted text they can find the guarded text through the lengthy procedure of trying every possible key.

Back in the early '70s, it was agreed that a strong cryptographic algorithm was needed. Development started on DES – the data encryption standard – which uses an algorithm called Lucifer. DES has a staggering 2^56 (about 10^17) possible keys. In the mid '70s this was sufficient to thwart all but the most dedicated government agencies. As processing power has increased however, so has the strength of brute force attacks, so the need for more key combinations is always growing. Given enough time all keys can be found but information usually has a finite useful life and so encryption only has to withstand the length of time that the information remains useful.

As all the information can be considered a string of numbers (ASCII symbols are just numbers) modern ciphers use mathematical functions to encrypt data. If the same key value is used to both encrypt and decrypt, then it is known as a symmetrical cryptosystem.

One example of a symmetrical cryptosystem is ROT13. Here we let A=1, B=2, all the way to Z=26. To encrypt we just move each letter along 13 characters. HELLO WORLD becomes URYYBJBEYQ. Performing the ROT13 process again return us to the original message. Many IRC systems include this as a quick method of disguising a message, until the recipient wants to pull the mask off. It's a good way to stop someone stumbling across the punch line of a joke or the spoiler to a film, for example.

Asymmetrical cryptosystems are much more secure, and therefore useful. These use mathematical functions that require two keys, which are not the same. Some ciphers work on a block of data, say one byte, with operations such as addition, transposition and multiplication, then move onto the next. A product cipher performs several block ciphers on each block. Feistel ciphers work on half of the cipher text then swap them round before performing on the next section. Lucifer happens to be a Feistel cipher.

Triple DES works on 64-bit blocks of data using 56-bit keys three times. This gives rise to the public-private key system. By using an asymmetric key system one key will encrypt while the other decrypts and vice versa. If you publish your public key then encrypt a message with your private key. Everyone can decrypt it (with your public key) and they know that only you could have encrypted it. This way you can also prove who you are. Similarly, they can encrypt a message to you, which only you can read. One form of this is the RSA (Patent 4405829). To test how strong this cipher is, RSA Data Securities Inc. post a series of numbers each month, and a cash prize is awarded to the first person to break down the factoring numbers.

Remember: the security and secrecy of your data not only lies with the power of the encryption algorithm, you must also bear in mind the security of your machine. As such, you should be thinking about your system security in general. Basically, we recommend PGP and GPG, as we don't *know* of any

Public and private: public/private key encryption is based on the use of two keys. The public key is freely distributable and can be sent in emails, cut and paste, or saved to floppy disk and handed out; and is used, in part, to encrypt messages that only you will be able to decrypt. The other key is the private key, which should remain secret and should not be spread. This key should only be available to the keyholder. Someone sending you an encrypted message will use your public key and their private key.

cracks in the system.

Encrypted data might be more than trivial to crack but other means can be used to attack your security and piece of mind. Some time back, a Trojan horse was found, which rooted about in systems searching for secret PGP keys and FTPd them away to some ne'er do well! The quality and integrity of your password obviously play as important a role.

How to encrypt under Linux

The two most frequently used means of encrypting files on Linux are using GPG (GNU PrivacyGuard, which is based on PGP – Pretty Good Privacy – but without the patent issues) and RIPEM. In this article we'll concentrate on GPG, which is shipped with all of the main distributions of Linux

First you need to generate two keys, **public and private**.

Generate your key

```
gpg  --gen-key
```

Here you are asked for the type of algorithm you wish to use. Lets use the default DSA/ElGamal, because it's not restricted by patents.

Next you are asked for a key length. The minimum is 768 and the DSA minimum is 1024, but lets think about this for a moment. The decision lies between choosing security and the amount of time you want to spend on encrypting messages. The greater the key length the less likely your message will be cracked open. We will run with the default of 1024-bits.

We are now asked for details such as our name, email address and a comment. Finally, we now need to enter a pass phrase to help generate our keys. You'll need to make this something you can remember so that you can decrypt your files, but not something that can be easily guessed.

Before starting to generate your keys, GPG will get some random numbers from the system, so working with lots of windows helps to generate randomness.

Getting and using your keys

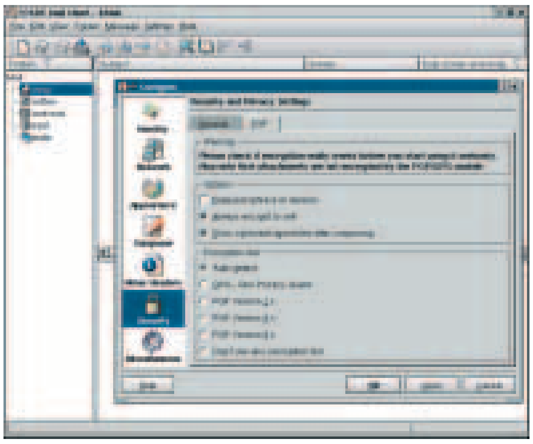
In order to have your key in a form that you can conveniently use, you'll need to export the key to a file:

```
gpg --export -ao public-key
```

The –a switch will produce your key in 7-bit ASCII, so it's easier on the eye, while the o switch sends it to the file 'public-key' so that we have something to handle.

When you receive someone else's key, you need to put this on your keyring. This is done with

```
gpg --import public-key
```


KMail ready to be configured with your newly created GPG keys

Key distribution

To make use of your public key, other people need to have a copy of it. This can be done by attaching the file in emails or by sending it on floppy disks. You can even use servers like *http://www.keyserver.net/*, which enables you to post your keys for other people to search for and pick up.

There is a problem of trust with this however. Who is to say that someone with evil intent won't give away public keys pretending to be you? The way around this is to use key signing. Your public key can be signed by other people who have verified it really is you, and who have been verified themselves. In practice you meet someone at a keysigning party for a quiet drink and exchange keys on floppies. Now their friends can accept your key because your friend says yes that really is you. In this way, a Web of trust builds up.

Locking the door

Now that we have our keys, and the public key of the person we want to send something encrypted to, we run with the command:

```
gpg -e -r LinuxMag test.txt
```

To be safe we had better sign the file as well

```
gpg -s data_file

gpg -d test.txt.gpg
```

will decrypt.

There are gui front-ends to help you through all of this (see *http://freshmeat.net/projects/*) but they are often not needed. The real trick is to set up your email client to automatically encrypt whenever you want – see the KMail screenshot for an example.

Info

GPG HOWTO	*http://www.dewinter.com/gnupg_howto/english/GPGMiniHowto-1.html*
Public key server	*http://www.keyserver.net/en/*