Encryption – Part I

By Andy Pepperdine

Encryption is the name for methods of hiding information from some, while allowing others access to it.

Introduction

This part will describe the basics of encryption and show how to answer the two questions:

1. How can I send an e-mail to you so that you can be certain that no one else but me could have sent it (e-mail signing)?
2. How do I send a message to you, so that only you can read it and thus keep it secret between the two of us (e-mail encryption)?

If everyone signed every message they sent, I think that spam would disappear overnight, as you could avoid reading anything unsigned or not trusted. In addition, it could be arranged that any public key could be required to correspond to a real e-mail address, with a check only if there is a complaint. If the check fails, then the key could be black-listed. The job of hiding the originator would be made very much more difficult.

Theory

The Internet is a public space. In particular, there is no way that you can be certain that anything you send cannot be intercepted and read by someone else. Nor can you be sure that an e-mail that arrives in your in-box was in fact sent by the person named in the “Reply-to” or “Sent-by” fields. It is a noisy, insecure, unreliable means of transferring data.

The challenge is to use this untrustworthy medium to transfer reliable data. However, it will always be necessary to use a different and secure method of transferring at least one piece of information between the parties, or it will never be certain that each can know the other is really who they say they are.

In historical times, there have been a variety of methods of substituting letters by others to encode a message, but they were restricted to what a human could easily do, and are not suitable for an age of computers and the Internet[Pip].

The current practice is to use a pair of keys. One is a secret (or private) key and is not given out to anyone. It is kept by the owner of the key and must be kept very safe and never ever divulged to anyone else. The other is a public key that anyone can see. It can be sent in any suitable way to anyone else at any time.

Mathematically, the problem is to find two functions P (public) and S (secret) with certain special characteristics:

1. Given P, it is practically impossible to find S. So anyone can encrypt a message using P, but only the owner of S can decrypt it. These functions are known as *trapdoor* functions, because they can easily be used in one direction, but the other is much more difficult. Note, it is not impossible, just very hard to discover the route back, and practical methods trade off

ease of computation in use with difficulty of computation when trying to crack it.

2. Applying P, and then S to a message, gives the same message: S(P(*m*)) = *m*. Anyone can encrypt using P, and the owner of S can decrypt (secret communication).
3. Also, the reverse, P(S(*m*)) = *m*. If a message was encrypted by the owner using S, then anyone can decrypt using P (signing, or signature verification).

The commonest method in use today is called the *RSA system* after the initial letters of the authors Rivest, Shamir and Adleman [Riv], which depends for its strength on the difficulty of finding factors of large integers. It uses only elementary number theory and is based on Euler's generalisation of Fermat's Little Theorem, which can be found in any introduction to Number Theory, see [Bur] for example, but there are many others. A full treatment of how this theorem is used in the popular RSA cryptographic technique is found in [Til].

Another, increasingly common, system is known as the *El Gamal system* and is based on work by Diffie and Hellman [Dif]. It uses a more involved technique to generate a key common between two people where each holds part of it. Mathematically, it is based on the difficulty of finding the discrete logarithm of an element of a certain type of finite field, and can be found also in [Til]. However, it appears that there have been some problems with earlier versions of the implementations of this in some products, although probably those with whom you wish to correspond with signed messages will also have reasonably up to date products that will not be affected, and they can easily update their software where necessary.

Preparation

I'm using Linux, Ubuntu 9.04 Jaunty Jackalope, to prepare the demonstration. Other versions of Linux will be similar. Windows has access to the same tools as will be described here.

Linux distributions will have installed the relevant package to handle the actual calculations involved in encrypting and decrypting. On Ubuntu, it is called `gnupg` and is usually installed by default.

Thunderbird has an add-on to handle encrypted and signed e-mails called `Enigmail`. It can be obtained from [Eni], or from Tools → Extensions → Get extensions and search for `enigmail`. I will use this to access e-mail that is signed or encrypted, and also as the GUI to create keys etc. On Ubuntu, there is a package with the name *enigmail* that contains and installs the Thunderbird extension.

For Windows users, [Cor] is a useful site to lead you through the download and installation of GnuPG, if you want to use it instead of Thunderbird.

Creating public and secret keys

With the extension installed, Thunderbird shows another Menu list named OpenPGP. Under there is Key Management, and the first time that is called, it may take you to a wizard that will initialise everything and generate a key pair for you. This key pair will be of the El Gamal variety, restricted to 5 years duration, and will not have any associated comment. You may wish to discard it, in favour of making another one from the Key Management window, Generate → New Key Pair where you will have full control of the values of all the variables.

While experimenting and finding your way, it is reasonable that a limit on the lifetime of the key is set in case anything goes wrong and you have to let it expire. Later you may want a more permanent one that you are comfortable letting the world know about.

I think it is useful adding comments to the name for the key as it enables you to tell the world under what conditions it will be used. For example, you may want one for work, and another for private use.

Important: Every key is usually associated with an e-mail address. If you change your e-mail address, you should probably change your key pair. This is not essential, but good practice, otherwise the identification of keys on the public registers can become misleading.

Signing a message

Signing a message can either be done on a message by message basis, or you can set up an identity to always sign when sent from that identity. I shall use the latter method. In any case, signing will have to be enabled for the identity from which you are sending the message.

In Thunderbird, go to Edit → Account Settings and highlight the account name you want to use. Then click on the button Manage Identities to pen the dialog to set up a new identity.

Hit the Add button, and in the Settings tab put in all your own information that applies to this identity. It is a very good idea to make sure that all your identities have unique names so that you can easily identify them. One suggestion is to add the phrase (signed) after your actual name. This will then be seen by the recipient, too.

Then move to the OpenPGP tab to define the signing method, in particular check the box Enable OpenPGP support. Also to sign messages by default, check the box Sign non-encrypted messages and any other types you require. If you click Advanced here, you can change how often you need to enter your pass phrase that protects your keys; typically it should remain short unless you are very secure where your machine is.

Hit OK a few times to close all the dialogs and you have an identity you can use without forgetting to sign your e-mails.

To use it, open a Write window in Thunderbird, and select from the drop down list on the From line the identity that you have set up for signing. Then type your message as you would normally. When you hit the Send button, it will now ask for your pass phrase to open your keyring, so enter it, and it will be sent off with a signature attached.

Telling others what your public key is

If you do all the above, you will discover that your recipients will not be able to verify the signature as they will not have your full public key. Before you go really public with this key, you will probably wish to test it with someone you trust, so you can create a file with the public key in it and attach it to an e-mail to send to whoever you wish to correspond with. Or even post it on a CD, or some other method.

To create the key in a suitable file, you will need to export it by OpenPGP → Key Management, and then highlight the key, and go to File → Export keys. Make sure that you do NOT send the secret key when it asks whether you should (the default is not what I expected!) and save somewhere convenient.

If you send the attachment from the signed identity, then you will be prompted with a dialog that recommends whether you should sign the message only, the whole message including all attachments, or each attachment separately. Only some e-mail clients can cope with the first two methods, and it defaults to doing it separately for each.

What to do when you receive the public key

If you receive a file like this, save it to disk in the normal way. If it came in an attachment to a message, then it can be saved as normal.

Then in Thunderbird go to OpenPGP → Key Management → File → Import keys from file, select the file and click Open to put it onto your keyring as a public key only.

Going back to the message and re-open it, and now Thunderbird will note instead of an unverified signature, it accepts the signature but from an untrusted key (on a blue background).

You can now say the owner is trusted in the Key Management by going to Edit → Change Owner Trust and checking the Fully trusted box. However, it will not change the colour of the background of the notice that Thunderbird displays.

What you can further do, is to “sign” the other's key which in effect is a mark that you have accepted and approved the use of this key by the correspondent (more on this later) and believe it to be accurate and correct. This is accomplished from Edit → Sign key after which you will find that Thunderbird changes the background colour to green.

Encrypting a message

Before you can encrypt any message you must enable encryption for your sending identity by going to Edit → Account Settings and Manage Identities for your account, highlight the identity and press Edit. In the PGP tab, check Enable OpenPGP support and leave the defaults otherwise alone.

Encrypting a message requires access to the public key of the recipient – not yours. When you have someone else's key and wish to encrypt a message when sending it to them, then write a message as normal first, but while writing it sometime go to OpenPGP → Encrypt message which will then look up the key associated with the recipient and encode appropriately.

Signing another's key and why

The notion of “signing” another's key may sound odd, but there are good security reasons for it. When you receive a key (perhaps by e-mail), how do you know that it really belongs to the person you believe sent it? How do you know that the e-mail was not concocted by someone else and the return address faked? Unless you have some other way of checking this key, then you will never be certain.

The way out of this impasse is to “sign” another's public key. In truth, I think a better term would be “witness” another's key, but unfortunately, the terminology is now well-entrenched. If you were to print out the key you think is from them, then the next time you see them, ask them to verify it is theirs, and to sign the page, you can then be certain that it really is a correct key for that sender. In that case, you can then “sign” their key as meaning it is trusted and you can be sure that anything signed with that key is good. You signing a key means that you are willing to put your name behind the accuracy of the association of key and person. This is the start of what is known as your “**Web**

of Trust”.

If there is someone else who trusts you, then when they see that you have signed a certain key, they can then assume that they can trust that key, even though they have never met the person, or know them at all. As a practical use for this, consider a project with a co-ordinator and several collaborators. If they wanted to have and be sure of all the others' keys, they would have to meet everyone else at some time. But an alternative would be to have the co-ordinator visit each of the collaborators and sign each other's keys. Then put them in a public place. When one of the collaborators, say Alice, receives a signed message from another, say Bob, she can look up his key and see that it is signed by the co-ordinator and can therefore trust the key, even though she has no other knowledge of Bob.

Public key servers and the web of trust

Before you let everyone in the world know about your public keys, make sure you really do want that. It is much easier to control the keys if you let people know who need to know and no one else. But the downside is that later generations may not be able to verify your signatures if the keys are not found in public archives.

A public key server is where keys are recorded for everyone to access. They contain the public keys and e-mail addresses of all the people who wish their keys known throughout the world. There are several of them, and in theory they periodically synchronise with each other, so it should not matter which one you use, it will be transferred to all the others over a period of time. However, my experience differs, and you should not expect this to happen. A key can be recorded on a public server from the Key Management window of Enigmail from the menu Keyserver → Upload Public Keys. There are search facilities available there, too.

You can even send your signature on a key by uploading their key, and it will record it as having been signed by you. That means that anyone else who sees the key, will also see your signature. If they trust you, they can then also trust the key. And so the web of trust expands.

Revoking a public key

Public keys appearing on a key server are for all practical purposes **not removable**. They can however be marked as *revoked* and that is why you should create and keep a revocation certificate around for all your publicly visible keys as you may wish to revoke them if they are out of date or in some way no longer usable [Ros], which also contains details of which key servers are currently active, which to avoid, and which do not share there contents with others.

If for some reason you lose your secret key and your revocation certificate, then [Ros] describes a sequence you can go through to show not to use the key. Everything needed is available from the Enigmail GUI, but there are lots of facilities not there, and the full documentation can found at [Gpg].

Further reading

If you are interested in following current developments and in improving your security, then a very recent article [Wil] shows that the current hash algorithms used to identify keys in a register, and to reduce a message before signing, have significant flaws. Work is afoot to improve matters, and the US Government has mandated phasing out SHA-1 by the end of 2010. This will force changes in signing and key handling by all current products. To make the move now to a stronger hash, then [Gil] shows you how. The message, though, is that the problem is largely theoretical with current computing power, but should be addressed for the long term. As they say, “Don't panic, but walk towards the exit”.

References

[Bur] Burn, R.P., *A Pathway into Number Theory*, Cambridge, 1982, ISBN 0-521-28534-8. A clear introduction by means of exercises and examples to give a good understanding of the subject.

[Cor] *http://www.coresecure.com/v5/gnupg.html*, contains a “how to” covering downloading GnuPG and its installation and use.

[Dif] Diffie, W. and Hellman, M.E., *New directions in cryptography*, IEEE Trans. Inf. Theory, IT-22, pp 644-654, Nov 1976

[Eni] *http://enigmail.mozdev.org/download/index.php*, where the add-on can be downloaded for any system.

[Gil] Gillmore, D.K., *HOWTO prep for migration off of SHA-1 in OpenPGP*, http://www.debian-administration.org/users/dkg/weblog/48 , Debian Administration, 6 May 2009.

[Gpg] *http://www.gnupg.org/documentation/manuals/gnupg/OpenPGP-Key-Management.html* contains the full documentation for gnupg. This URL is the page for the editing commands.

[Pip] Piper, F, and Murphy, S, *Cryptography: A Very Short Introduction*, Oxford, 2002, ISBN 978-0-19-280315-3. A short book that gives some historical information and general description of current practice.

[Riv] Rivest, R.L., Shamir, A., and Adleman, L., *A method for obtaining digital signatures and public key cryptosystems*, Comm of ACM Vol, 21, pp 120-126, Feb. 1978

[Ros] Ross, D., *PGP: Public Key Servers*, http://www.rossde.com/PGP/pgp_keyserv.html has a good description of the services on all public key servers, including information on how to proceed if you ever lose your secret key.

[Til] van Tilborg, Henk C.A., *An Introduction to Cryptology*, Kluwer, 1988, ISBN 0-89838-271-8. A thorough mathematical treatment of several different cryptographic methods. Chapter 8 covers Diffie-Hellman, and Chapter 9 covers RSA.

[Wil] Willis, N., *Dealing with weakness in SHA-1*, http://lwn.net/Articles/337091/, Linux Weekly News, 17 June 2009.