Notes on GnuPG (The GNU Privacy Guard)

Matt Giuca

From GNU Privacy Handbook:

http://www.gnupg.org/gph/en/manual.html

What is it for?

Signing, encryption. Debian packages, signed and encrypted email.

Non-GPG examples: SSH, Bitcoin.

Keys and algorithms

Three algorithms: DSA, RSA and ElGamal. RSA can be used to encrypt or sign, DSA only for signing, ElGamal only for encryption.

You can choose a) RSA for signing and encryption, b) DSA for signing and ElGamal for encryption, c) DSA for signing; no encryption, c) RSA for signing; no encryption. (a) is the default.

(GPG Manual says that ElGamal can also be used for encryption, but I assume they took that out.)

What is a subordinate key? Keys can be for either signing or encryption. Used to be able to use a single ElGamal key for signing and encryption. I'm not sure why, but when you use RSA for both, you need a separate signing key to the encryption key (perhaps because the encryption key should be changed periodically). Typically, you will have a signing RSA key as your primary key, and you can then attach an encryption RSA key as a subordinate key. The subkey is signed by the primary key – if you trust the owner of the primary key, then you can trust the subkey.

All of the other metadata (email addresses, photos, etc) is also signed by the primary key. Therefore, the only thing that needs to be verified about a public key file is the key fingerprint – if that checks out, then every other piece of data in the file is legitimate.

Always put a passphrase on your GPG key, so if the file is lost, it can't be used (you would still consider this compromised just in case, but it would give you extra time to revoke, etc). Have a really good passphrase that isn't one you use elsewhere.

Always put an expiry time on your GPG key, so if you are unable to issue a revocation, at least it will expire eventually.

Revocation

A *revocation certificate* is a special file which says “I declare this keypair no longer valid.” It is signed by the private key in question. If your private key is compromised, you should issue a revocation to make it unusable. This isn't ideal, because you need to propagate the revocation around as much and as fast as you can.

Once revoked, the key cannot be used to sign or encrypt. It can be used to verify or decrypt, but it gives a big warning.

One thing the manual suggests is that you print a revocation certificate *in advance*. You can generate one at any time (it is about 8 lines of base-64 text), print it out, and keep it in a safe place. That way, if you ever lose your key, you can issue a revocation.

Changing keys

Keys should (but don't have to) have an expiry date – usually a couple of years into the future (mine expires in July 2012, two years after I created it). What happens when they expire? Don't you have to start again and have everyone sign your new key?

You can use the *expire* command to extend the key's expiry time (as long as it hasn't yet expired!) The key signs itself with the new expiry time. This basically says “the owner of this key is still in control and has recommended the expiry” – the implication being that if it doesn't expire, the owner is no longer in control.

There is a different expiry model for signing keys and encryption keys. Because if a signing key expires, it cannot be used to verify. But if an encryption key expires, whoever has the key can still read the encrypted documents. Therefore, you will not want to *update* the expiry on encryption keys – rather, create a new one and let the old one expire. But you will want to update the expiry on signing keys, since it is too much hassle to create a new one.

(In other words it is much more important to periodically update encryption keys than signing keys, because a compromised encryption key breaks all *past* encryption whereas a compromised signing key breaks all *future* encryption.)

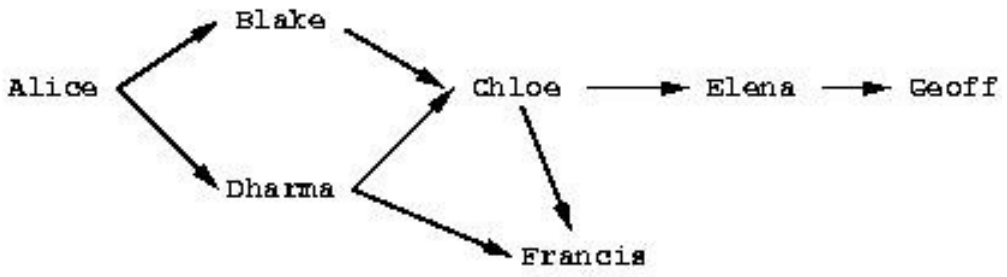
Trust model

Trust (unknown, none, marginal, full) is a *private* setting you assign to each contact. It is “how much do I trust this person with the PKI?” So it isn't about how sure are you that they are this person, or would you lend them money, but specifically, if you were to see their digital signature on another key, would you believe they have checked thoroughly. You directly set the trust level for each person.

Validity is a *subjective* measure of how much you can believe that a key belongs to a person. It is not something you set – it is computed by GPG. Keys are either valid or invalid. A key is valid if:

1. it is signed by enough valid keys, meaning
    - you have signed it personally,
    - it has been signed by one fully trusted key, or
    - it has been signed by three marginally trusted keys; and
2. the path of signed keys leading from *K* back to your own key is five steps or shorter.

Figure 3-1. A hypothetical web of trust



trust		validity	
marginal	**full**	**marginal**	**full**
	Dharma		Blake, Chloe, Dharma, Francis
Blake, Dharma		Francis	Blake, Chloe, Dharma
Chloe, Dharma		Chloe, Francis	Blake, Dharma
Blake, Chloe, Dharma		Elena	Blake, Chloe, Dharma, Francis
	Blake, Chloe, Elena		Blake, Chloe, Elena, Francis

How can we make it more widespread?

- Put your public key on your web page / social networking profile / etc.
- Sign other peoples' keys.
- Send mail signed (hard to do with webmail – I don't).