An Introduction to How PGP Works

Revision: 0.01

Author: Kenneth Robert Ballou Date 2005-07-21

Document Change History				
Change Description/Reason	**Changed By**		**Date**	**Revision**
Initial Creation of Document	**Kenneth Robert**	**Ballou**	**2005-07-21**	.01

Table of Contents

Figures

I. Introduction

PGP is a powerful application providing security for files and for e-mail. However, it can be difficult to get started with PGP. Unfortunately, there is a lot of unfamiliar terminology in PGP, and it isn't really clear what the beginner needs to know to use PGP effectively.

We'll try to explain important PGP terms here as painlessly as possible. This is intended to be more than a "do this, then do that" explanation. Instead, we hope the reader will come away with enough understanding that PGP actually makes sense and will be able to explain how to use PGP to other newcomers.

In addition to PGP, there is also GNU Privacy Guard, also known as GPG (or gnupg). The concepts in this document apply to both PGP and GPG. For simplicity, we will simply refer to PGP from here on.

II. Important PGP terms

A. Key

At the heart of PGP is something called a "PGP key." In fact, this term is a bit misleading. A PGP key is really a key pair, which consists of a private key and a public key. The term "PGP key" almost always refers to the public key part of a key pair.

A PGP key is either an "RSA key" or a "DSA key." These terms refer to cryptographic algorithms. It's really not important to understand these algorithms (they are based on complex mathematics that mostly just leads to a headache). Each algorithm offers excellent security.

Until the year 2000, DSA was much more popular than RSA because the RSA algorithm was encumbered by a patent. That patent expired September 17, 2000. Since then, RSA has become much more popular than DSA. If you're creating a new PGP key, you probably should choose RSA as the type of key. You should be sure to specify a key length of **at least** 1024 bits for either RSA or DSA. As of 2005, 512 bit RSA keys are considered insecure; 768 bit keys are considered secure, but security experts recommend that new keys should be at least 1024 bits long. (The longer the key, the harder it is for a stranger to "crack" the key, or discover the private part of the key pair. Once someone has the private key, that person can impersonate you without your knowledge.)

B. UID (User Identity)

A public key is basically a big number – a *really* big number. For example, a 1024-bit public key is a 308 digit number. (Think of 1 followed by 307 zeros:
10,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,
000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,
000,000,000. That's ten million trillion. How's that for big?)

People don't remember numbers that big, but computers can handle them quite nicely. People do remember names and e-mail addresses as identities. So, a UID (user identity) answers the question "who does this public key belong to?"

C. Signature

In general, in cryptography, the term "signature" refers to something you've used your private key to generate. As in the pen and paper of the physical world, you "sign" a document by feeding that document along with your private key to the "signing algorithm" (either RSA or DSA). Someone who wants to check your signature on a document feeds the document, your signature, and your public key to the signing algorithm. The signing algorithm will then either say that "the signature is verified" or "the signature could not be verified." If the signature could not be verified, there are two possible reasons. Either someone altered the document after you signed it, or someone tried to "forge" your signature without your private key. (This is why it is **essential** to keep your private key private. Anyone who has access to your private key can generate signatures that appear to have legitimately come from you.)

D. Certificate

Suppose Alice creates a user identity that says "I'm Alice, and my e-mail address is alice@alice.com." Now, Bob knows Alice, so Bob is willing to vouch for Alice's identity. Bob gets Alice's public key from her. Bob then creates a *certificate* that says "I, Bob, state that this identity for Alice is correct and that this public key belongs to Alice." Bob signs this certificate with his private key. Bob then adds this signed certificate to Alice's public key and sends the revised key back to Alice. Now, when Alice gives out her PGP public key to Carol, Carol can check Bob's signature on the certificate. If the signature verifies correctly, then Carol knows that Bob has vouched for Alice's identity.

III. What's in a PGP key?

A. The simplest PGP key

The simplest PGP key contains the following items:

1. The public key itself
2. The user identity (UID)
3. A certificate corresponding to the UID.

It's pretty clear why the key contains the public key itself and a user identity. But why is there a certificate following the UID, and who creates the certificate?

Suppose there were no certificate. Then, suppose Alice creates a new PGP key. This key contains the public key portion of the RSA or DSA key pair and the UID that says "This key belongs to Alice (alice@alice.com)." Let's suppose Mallory is a malicious prankster who wants to deface Alice's key. What prevents Mallory from replacing the UID with one that says "This key belongs to Mallory (mallory@mallory.com)"? That's the purpose of the certificate. Alice creates a certificate that says "I'm Alice, and I'm asserting this key belongs to Alice (alice@alice.com)." Alice signs the certificate with the private key. Then, anyone who uses this PGP key can check Alice's signature with the public key to verify the UID hasn't been tampered with. Because Mallory does not have access to the private key, his attempt to deface Alice's PGP key will fail.

When you generate a new PGP key, PGP asks you for a name and an e-mail address to include in the key. PGP uses this data to create the UID for the new key. PGP also creates the certificate for the UID automatically.

B. Creating a PGP key

If you haven't done so yet, this is a good time to generate a PGP key. Start the PGPKeys application. From the "Key" menu, select "New key…". In this example, we've used "Alice" as the name and alice@alice.com as the e-mail address. When PGPkeys is done generating the key, the PGPkeys window looks like this:

Figure 1 Alice's PGP Key

If we click the "+" next to the key, PGPkeys shows the user identity (UID) associated with the key:

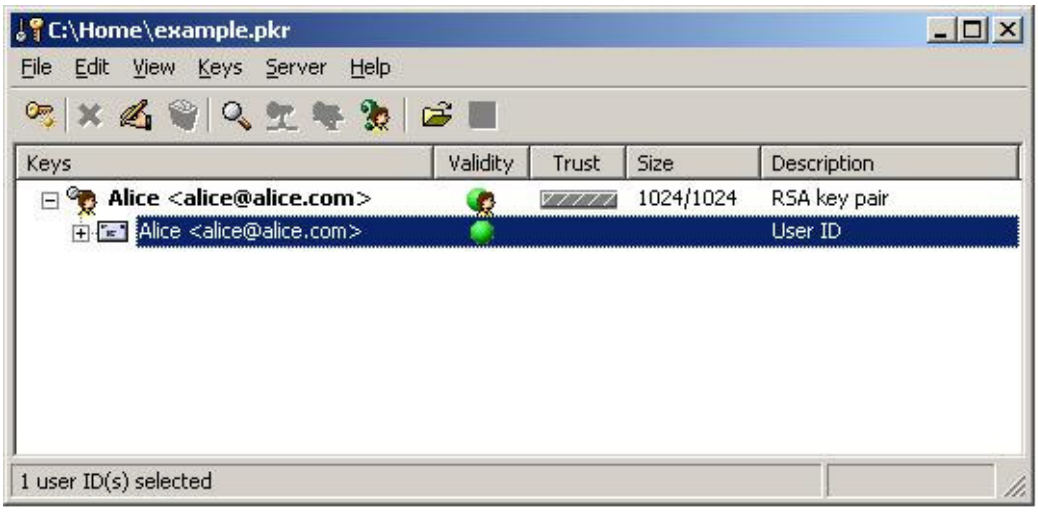

Figure 2 Alice's UID

If we click the "+" next to the UID, PGP shows the signature (certificate) associated with the UID:

Figure 3 Alice's Signature on the UID

Notice that the signature is associated with the UID, **not** with the key. This is a subtle point in PGP. You vouch for identities. Specifically, a signature certificate vouches for a pairing of an identity with a PGP key. (We'll come back to this point later when we discuss having multiple UIDs associated with a key.)

C. Collecting signatures

Now, suppose Alice gives her PGP key to Bob. Bob decides to *sign* Alice's key. In fact, as we just described, this is sloppy (but unfortunately commonly used) terminology. Bob is actually signing Alice's UID.

First, Alice exports her PGP key by selecting her key in the PGPkeys window and then choosing "Export…" from the "Keys" menu. Alice saves her public key to the file Alice.asc and sends this file to Bob. Bob opens PGPkeys and chooses "Import…" from the "Keys" menu. He opens the file Alice.asc and sees the following window:

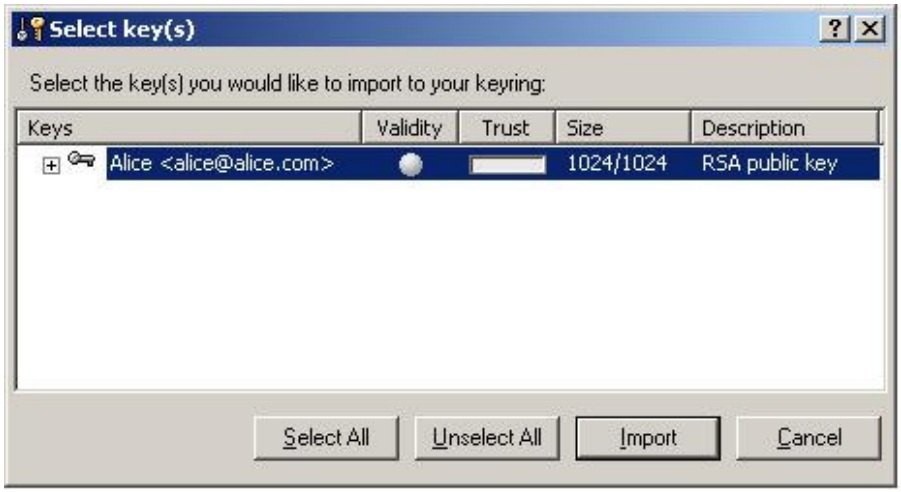

Figure 4 Bob Imports Alice's PGP key

Bob presses the "Import" button to import Alice's key into his keyring. He then right clicks on Alice's UID and chooses "Sign…" from the popup menu. This window appears:

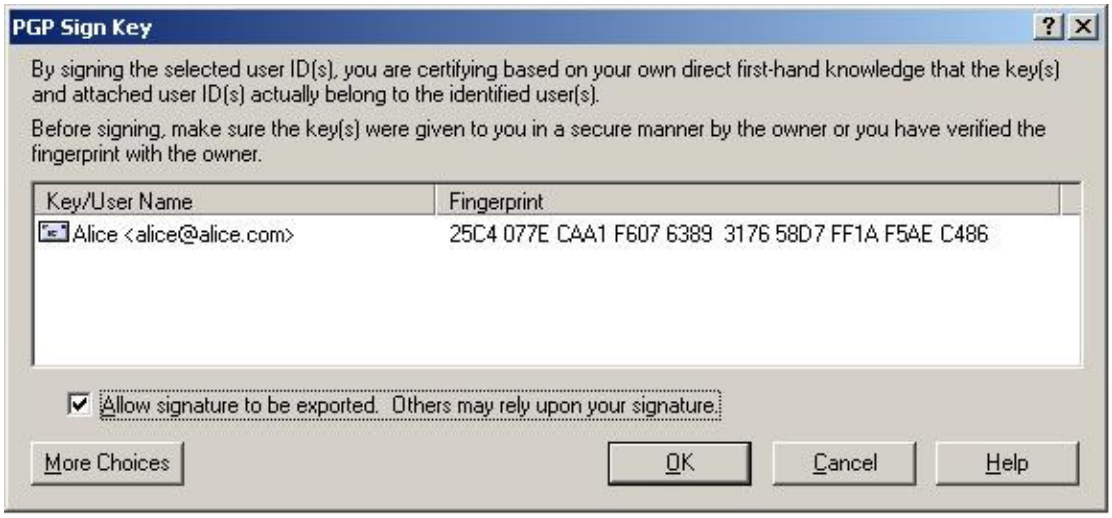

Figure 5 Bob signs Alice's user ID (UID)

Notice that Bob has checked the "allow signatures to be exported" check box. Bob is confident enough in his belief that this key belongs to Alice that he is willing to make an assertion to other PGP users.

After Bob clicks the "OK" button, he is prompted for the password for his private key. He enters the password and presses "OK." PGP then creates a signature (certificate) for Alice's UID and inserts it into Alice's key. Bob now sees this in his PGPKeys window:

Figure 6 After Bob signs Alice's user ID

Notice there are now two signatures on Alice's UID. Also, notice that the dot for Alice's UID in the "Validity" column has changed from grey to green. By signing Alice's UID, Bob has asserted his belief that the UID for Alice is correct and that the public key really does belong to Alice. The dot on top (next to Alice's key) is green because at least one UID associated with the key is valid.

After Bob signs Alice's identity, he exports Alice's PGP key (by choosing "Export…" from the "Keys" menu in PGPKeys). He sends the exported key back to Alice. Alice imports the key (by choosing "Import…" from the "Keys" menu in PGPKeys). After Alice imports the key, her PGPKeys window looks like this:

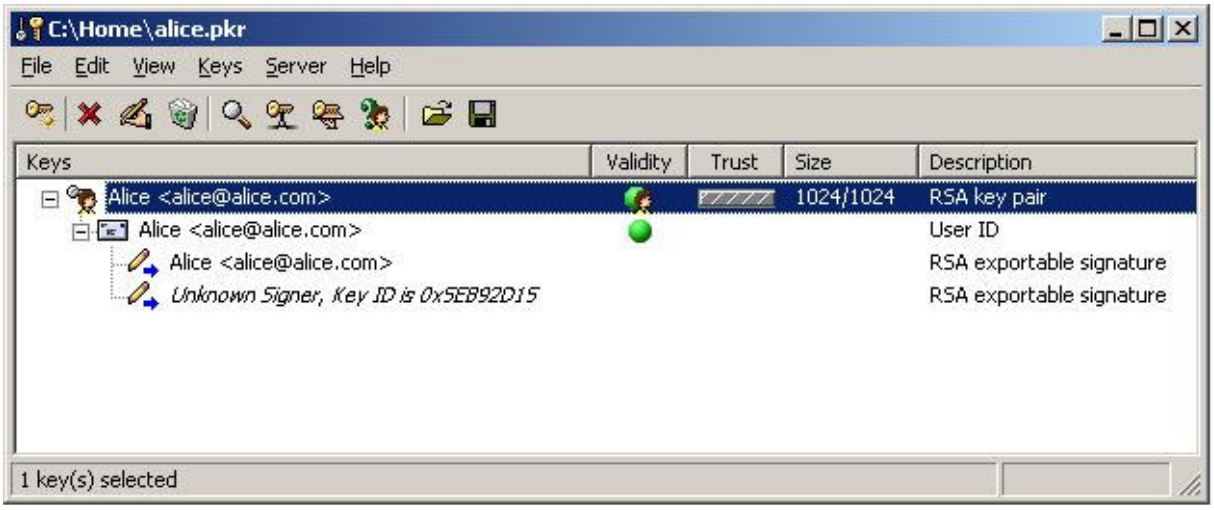

Figure 7 Alice imports Bob's signature

Notice that Alice does not have Bob's PGP key on her public key ring. Bob's signature there appears as "Unknown Signer" with a key ID. After Alice imports Bob's PGP key, "Unknown signer" will be replaced by "Bob <bob@bob.com>".

It is very important that Bob is certain the key belongs to Alice before he signs Alice's UID. When we generated a new PGP key, we saw there is nothing stopping us from entering anything whatsoever in the "full name" and "e-mail address" fields in the dialog box. So what stops us

from entering "Bill Gates" in the "full name" field and billg@microsoft.com in the e-mail address field? Nothing at all stops us. But it's unlikely we'll get anyone who is honest to sign this UID.

D. Multiple user IDs (UIDs) on the same key

Now, let's assume the e-mail address alice@alice.com is Alice's personal home e-mail address. Alice has been using PGP happily for a while. In fact, Alice is so happy with PGP that she decides she would like to use PGP for her work e-mail as well as for her personal e-mail. Alice has a choice. She can generate a second PGP key and specify her work e-mail address (say, alice@xyzcorp.com) for the UID associated with this new key. Or, Alice may think to herself, "I am the same person whether I use the e-mail address alice@xyzcorp.com or alice@alice.com, so I want to use the same PGP key in both cases."

Let's suppose Alice decides to add a new UID to her existing PGP key. Alice opens the PGPKeys window and clicks on her key to select it. Alice then chooses "Add…" from the "Keys" menu. Another menu appears, and Alice chooses "Name…" from this menu. Alice then sees this dialog box:

Figure 8 Alice adds a UID to her key

Alice then clicks the "OK" button. PGP prompts Alice for the password for her private key. After Alice enters her password, PGP adds the new user ID to her key. Alice's PGP window now looks like this:

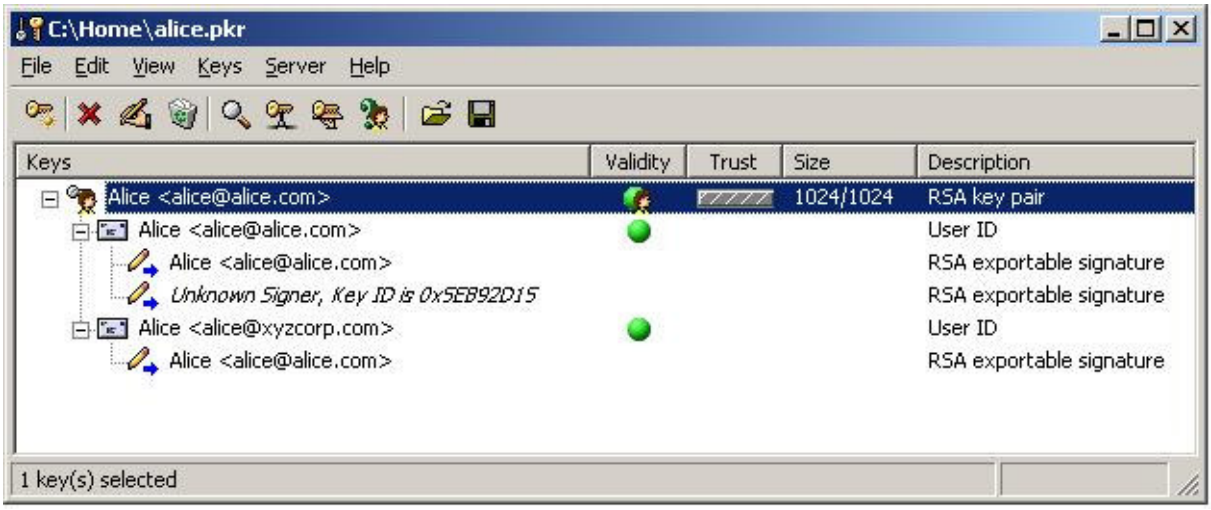

Figure 9 Alice now has two user IDs

Was it a surprise that Alice needed to enter the password for her private key when she added the new UID to her key? Remember that only the owner of the PGP key should be able to add a UID to the key. PGP enforces that restriction by adding a *self-signature* certificate (a certificate signed with the private key corresponding to this PGP public key) to each UID. Because this self-signed certificate can not be created without the private key, no one other than Alice can add identities to her PGP key.