RTC Installfest Manual

About the RTC

The Radical Technology Collective is an anonymous organization dedicated to the furtherance of the free software movement and of digital security practices in the anarchist community. The RTC believes that as anarchism enters the 21st century, it will have no option but to embrace these concepts, or perish or mutate into something fundamentally opposed to what it has stood for for centuries.

The RTC sees the Free Software movement as a logical extension of anarchist ideals applied to software. Free Software supports community involvement in the use, distribution, and production of software over corporate involvement, and ensures the user has liberty in the digital realm. Without liberty, and without community, how can we be anarchists?

Additionally, the RTC supports the safe and secure use of the Internet and other computer-related infrastructure to further the goals of the anarchist struggle. We recognize that many, if not most, among us are hunted by state or corporate interests, and we attempt to help anarchists of every color and creed remain secure, safe, and free.

It is the belief of the Radical Technology Collective that technology is neither inherently good or bad, but fully neutral. Therefore, it is not only our responsibility to advance technology as far as humanly possible, but also to ensure it is only used for ethical means. We hold that the negative uses of any technology have the potential to be limited and outweighed by the positive uses. We hold that the access to information is a fundamental human right, and therefore, access to technology related to information, such as computers and the Internet, is a fundamental human right.

Contacting the RTC

You can find the RTC's contact information on our homepage, currently on LibrePlanet:

<http://groups.fsf.org/wiki/RTC>

Our homepage contains links to our materials, our OpenPGP public key, and our current contact information.

Anonymity

All RTC contributors remain anonymous with respect to their work with the RTC. The RTC believes that the source of an idea is irrelevant to its worth, and will never take the source of an idea into regard when attempting to judge the worth of an idea.

Installing Ubuntu

Why Ubuntu

Ubuntu is a GNU/Linux distribution that has become increasingly widespread as of late. Several OEM's have begun installing Ubuntu, most notably Dell. Ubuntu is secure by default, and is very easy to install fully encrypted.

While Ubuntu does include and install non-free software, this may be optimal for those who have no other hardware but that which requires non-free software. Also, since gNewSense is based on Ubuntu, it is possible to 'upgrade' to it (though this is recommended for advanced users only and not covered in this guide) from Ubuntu and be secure in freedom.

In short, Ubuntu makes it very easy to have a fully functional GNU/Linux system with the maximal degree of security.

Getting the Disks

The first step in any installation process is obtaining install media. If you are installing the RTC's Ubuntu remix, the best way to download the disk image is via one of the torrents. You can find them in the 'ubuntu' section of any RTC mirror.

If you cannot obtain the RTC Ubuntu remix, download a CD from ubuntu.com. <bold>Download the "alternate install" CD, <underline>NOT</underline> the "live CD". The "Live CD" will <underline>NOT WORK</underline>. <italics><underline>DO NOT ATTEMPT.</underline></italics>

Once you've downloaded the CD image, burn it to a disk. To do this, you may or may not need special software, depending on your operating system. There are far too many possible programs to enumerate how to burn the CD in all of them, but one thing you CANNOT do is merely copy the image file to the CD. That will not work. Look for an option that is similar to "Burn image to disk" or "Copy image to disk".

Initial Steps

After burning the disc successfully, leave it in your computer's CD drive and reboot. If you computer is configured properly, it should reboot into the CD. If not, restart your computer again and change the boot order in the BIOS (the initial environment for your computer). Typically, when your system is first powered on, it will display a key to press to access the settings -- like F12 or F1 or DEL, etc..

Once you are booted into the CD, choose the language for the installer, and select the menu item "Install Ubuntu".

The installer will then ask you what language you want to install the system in, and guide you through selecting a keyboard layout. This is all rather straightforward and shouldn't pose any difficulty.

After this, the installer will load components from the CD and attempt to autoconfigure the network. If you are on a laptop or another system with a wireless card, the autoconfiguration may fail -- this isn't an issue. If it does happen, select "Do not configure the network at this time".

The installer will then ask you for your hostname -- this is just a name for your computer, and you can safely pick just about anything. When you've picked a name, the installer will ask for your timezone.

Partitioning

Partitioning is the most difficult part of the installation, which isn't saying terribly much. However, it is also the most important.

ACHTUNG! WARNING! FOLLOWING THE INSTRUCTIONS IN THIS SECTION WILL DESTROY ALL DATA ON YOUR DISK! MAKE SURE YOU HAVE BACKUPS!!!

The GNU/Linux filesystem is monolithic and hierarchical. There aren't any drive letters (c:, d:, e:) like in DOS or its variant, Windows. Instead, all filesystems are merged into one single filesystem, which begins at the directory '/'.

The biggest upside of this is that unlike Windows, where your user files are tied to the same partition as the system files, on GNU/Linux, we can make '/home', the directory where user files are stored, a different filesystem than '/', the root directory (where we'll install the system to).

Before you begin, it's easiest to obliterate the old partition layout on the disk by making a new partition table on the disk. To do that, select the name of the disk and follow the instructions, selecting "yes" when told of the imminent doom of the device's data.

Now we can start making new partitions. To make a new partition, select the free space in the menu, and select "create a new partition". Then, when the installer asks, input the size and type of the partition. There's no hard and fast rule for these values -- they will vary from partition to partition.

Order is important for the next steps -- don't expect to be able to jump around unless you really know what you're doing.

The first partition we're going to make will house the '/boot' filesystem, which will house the kernel and other bootloader files. It shouldn't be more than 500MB in size, and should be a primary partition at the beginning of the available space.

By default, the installer will produce an ext3-format filesystem configured to be the '/' or root directory. Select "mount point", and change the value to '/boot - static files of the boot loader'. Then, move the cursor down to the "Bootable Flag" option and select the entry to toggle it to "on". When you've done that, verify the value for "Bootable Flag" is 'on', and then select "Done setting up the partition".

The next partition we're going to make will house encrypted Logical Volume Management system, which will in turn house the rest of our partitions. Make a new partition by again selecting "Create new partition" after selecting the free space on the main partitioning screen. Accept the default (or put in 'max') for the size, and select "logical" for the partition type.

At this point, you'll be back to the "Partition Settings" screen. Select "Use as", and then select "Physical volume for encryption". When you get back to the Partition Settings screen, it will have changed -- several new options will have appeared. Feel free to accept the defaults if you do not need a high level of security -- as of the time of this document's writing, the installer defaults to using AES as the encryption type, and a key length of 256 bits, which is perfectly fine for most use.

If you have highly confidential files, do not care about disk access times, or are paranoid, feel free to select "Encryption" and change it to Twofish or Serpent. Twofish is the best compromise between security and usability -- it is faster than Serpent, and slower, but slightly more robust, than AES. Serpent is not for the faint of heart or the impatient. Do NOT select blowfish as your encryption algorithm -- it is outdated and not as strong as the other options.

When you select "Done setting up the partition", an extra option will appear on the main partitioning screen at the very top -- "Configure Encrypted Volumes". Select it to continue partitioning.

Before you move on, the installer will prompt you to verify the changes committed to the disk. SELECTING YES HERE WILL IRREVOCABLY MODIFY YOUR DISK AND POSSIBLY DESTROY YOUR DATA. BE CERTAIN OF WHAT YOU ARE DOING.

The installer will then ask you for a passphrase. You will have to enter this passphrase every time you turn on your computer. It will protect your data from being read or modified by those without the passphrase. If you forget your passphrase, all the data on the system will be lost.

This passphrase must be secure, but also memorable. To be secure, a passphrase should be longer than 10 characters and contain uppercase and lowercase letters, numbers, and symbols (like &, or *). Pick a secure passphrase, enter it twice, and continue.

When you finish configuring the encrypted volume, it will appear in the listing of disks above the physical disk, with a single partition. Select that partition, and then select "Use as" and then "physical volume for LVM". Select "Done setting up the partition" to continue.

Now, another option will have appeared at the top of the main partitioning screen - "Configure the Logical Volume Manger". Select it.

We're now going to configure the LVM. It will contain the last three filesystems in our configuration. First, select the "Create volume group" entry, and pick a volume group name. The name does not matter and can be anything you want. Select the encrypted partition to be in the LVM, and then continue back to the main LVM screen.

We now need to create three volumes in our volume group.

The first one should be called "swap". For its size, put twice the amount of memory in your system. If you don't know the amount of memory in your system, two gigabytes should be fine.

The second should be called "root". For its size, put about 10% of the space left after subtracting the size of the two partitions we just created (/boot and swap). At minimum, put 10GB. At maximum, put 50GB.

The third should be called "home", and should have the rest of the available space.

When you have created all of these volumes, select "Finish" in the main LVM configuration screen to return to the main partitioning screen.

Your logical volumes will now appear as partitions above the encrypted partition.

First, select "swap". Initially, the value "Use as" will be set to "Do not use". Select it, change it to "swap area", and then select "Done setting up the partition."

Next, select "root". This time, however, select "Use as" and change the value to "EXT4 Journaling File System". After this, select "mount point" and change the value to "/ - the root file system". After that, select "Done setting up the partition."

Last, select "home". Select "use as" again and change the value to "EXT4 Journaling File System". This time, however, when you select "mount point", change the value to "/home - user home directories". After you've done that, select "Reserved Blocks", and change the value to 0. Then select "Done setting up the partition."

You're done partitioning! Scroll to the bottom of the main partitioning menu and select "Finish partitioning and write changes to disk." The installer will confirm one last time that you know what you're doing, and then proceed to format the disk and install the base system. You can sit back and watch it, marveling at the glory of free software.

The rest

Once the base system is installed, the installer will start setting up your user account. It'll ask for your real name, your username, and your password -- follow along with it. If you're not using the RTC's Ubuntu remix, you'll be asked if you want to encrypt your home folder. It's best to say yes here -- the crypto is fast, and this allows you to have a little extra security without any extra effort. On RTC's Ubuntu version, this is enabled by default.

Next, the installer will begin setting up the rest of the system. It'll begin by asking you for your http proxy information -- if you need to enter anything here, you'll know it. If you don't know, then just hit enter and wait for the system to install.

The next time you'll need to do anything with the system is when it asks you to reboot -- you're done at this point. Enjoy your new, free, secure operating system.

Adding and Removing Software in Ubuntu

On virtually every non-free operating system, finding and installing software is a laborious process. This is because non-free operating systems deny themselves the ability to share and collaborate with a community -- as such, instead of coming bundled with useful programs, non-free systems typically come bundled with the dreck of whatever corporation managed to pay the vendor. Free operating systems like Ubuntu, on the other hand, have no such failing.

Almost every GNU/Linux distribution has a framework for package management. A package is a bundle of anything that can be installed on a computer; graphic programs, command-line programs, system libraries, development libraries, documentation, art, themes, or anything the developers can cram into an archive.

There are three primary ways to "manage packages" -- that is, install, uninstall, reinstall, and configure packages -- on Ubuntu. The first is gnome-app-install, or "Add/Remove", the next is Synaptic, and the last, but by far the most powerful, are the command-line Apt tools. All of these programs are interfaces to the Advanced Packaging Tool, the main framework upon which Ubuntu, and its parent system, Debian, are based.

To open gnome-app-install, click on Applications in the main Ubuntu menu, then click on "Add/Remove." If this is your first time using gnome-app-install, click on the drop-down menu in the upper left of the screen and select "All Available Applications" to make all possible programs visible.

gnome-app-install is a manager for graphic programs only -- the categories on the left-hand side match where they'll be in the Gnome menu (under Applications). Installing and removing is simple -- just check the box or uncheck the box, as applicable, and click "Apply Changes" in the lower right to have gnome-app-install add or remove the programs you've selected or deselected.

Before it does that, though, gnome-app-install will ask you for your password. This is because adding software to the system is a privileged operation only available to the system administrator. By giving it your password, you're authenticating as an administrative user.

However, there are many more packages available to install than just the ones in gnome-app-install. To see the entire package management system in a graphic program, use Synaptic. To start it, click System, then Administration, then Synaptic Package Manager, and enter your password when it asks.

Synaptic looks like gnome-app-install somewhat, but is much more advanced. Instead of seeing menu categories on the left, you'll see the "sections" packages can be in. There are quite a lot of sections, and you might be better off just ignoring them and searching for what you want with the aptly named "Search" button.

Additionally, instead of just installing or removing, in Synaptic, you have access to the full array of Apt actions. These include reinstalling and "purging", or what's marked as "Remove Completely" in Synaptic. This will remove a package and all configuration files or documentation related to it.

While Synaptic is powerful for a graphic program, it can't compare with the command line. The command line is by far the most powerful interface to a GNU/Linux system -- instead of being limited to what you can poke out through menus, you're able to tell the system exactly what you want. The command line tools to interface with Apt are 'apt-get', which handles adding and removing, and 'apt-cache', which handles searching and displaying data about packages. This article will only mention a small subset of these programs' options -- to see all of them, read their manual page by running "man <program name>" in terminal, where "<program name>" is replaced with the name of the program, "apt-cache" or "apt-get".

To search for software on the command line, use the "search" argument to apt-cache:

```
apt-cache search firefox
```

and to show the description and other information for an individual package, use the "show" argument:

```
apt-cache show gnupg
```

To install a package, use the "install" argument to apt-get:

```
sudo apt-get install anarchism
```

'sudo' is a command to run other commands as the system administrator, which is required since we're adding software to the system. Give it your password to continue (by the way, that *is* a real package in Ubuntu).

To remove packages, you guessed it -- tell apt-get to remove them.

```
sudo apt-get remove tomboy
```

and to remove a package completely, with configuration files, use the --purge flag:

```
sudo apt-get remove --purge mono-common
```

(Note: If you don't want to remove those packages, don't run the above commands. If you don't know what packages they are, you probably don't want to bother removing them.)

The strongest aspect of the free software movement is the availability of community -- by not squashing community like non-free software does, free software allows communal sharing of software, leading to the elegant system described above. For almost any possible need you may have, there is software in the Ubuntu package system that can meet that need. That's the power of freedom.

Generating an OpenPGP Key with GnuPG

Note on terminology: OpenPGP is a standard, defining how programs interact with each other with regard to encrypted data. PGP is a program that uses that standard, as is GnuPG. As such, we are generating an OpenPGP key with GnuPG, but you might hear the term "PGP key" or "GPG key". They all mean the same thing.

We're going to generate a pair of 4096-bit RSA keys. One of those keys will be limited to signing, and will not expire. This is the key you can have your friends sign to verify its authenticity. The other will be limited to encryption, and will expire in a year (or less). This protects us in case of compromise -- an attacker will only be able to use that advantage until the encryption key expires, at which point they will have to compromise your key again if they wish to read your communications.

While most GNU/Linux systems come with some sort of graphic manager for GnuPG, the command line is universal. As such, the following guide expects you to be using a terminal emulator of your choice -- you can find the default GNOME terminal emulator on Ubuntu in Applications>Accessories>Terminal. Once you have that started, continue.

Section One: Generating your Primary Key

Start up GnuPG with the following command line:

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

The option '--gen-key' means, obviously, generate key. The '--expert' option allows us to use aspects of the generation process GnuPG thinks only "experts" will be able to handle. Congratulations, you are now an expert.

You will probably be faced with a screen like this:

```
 gpg (GnuPG) 1.4.9; Copyright (C) 2008 Free Software Foundation, Inc.
 This is free software: you are free to change and redistribute it.
 There is NO WARRANTY, to the extent permitted by law.

 Please select what kind of key you want:
    (1) DSA and Elgamal (default)
    (2) DSA (sign only)
    (3) DSA (set your own capabilities)
    (5) RSA (sign only)
    (7) RSA (set your own capabilities)
Your selection?
```

Choose the option (7), by typing _7_ and pressing enter. Any time GnuPG asks you to make a selection, you do so by typing the number of the selection you want, and pressing enter.

Now, GnuPG will show you this screen:

```
Possible actions for a RSA key: Sign Certify Encrypt Authenticate
Current allowed actions: Sign Certify Encrypt

   (S) Toggle the sign capability
   (E) Toggle the encrypt capability
   (A) Toggle the authenticate capability
   (Q) Finished
```

We want this key to be able to sign, but not to encrypt. The 'authenticate' capability is unrelated to this guide, but you can enable it if you want and play with it later. To disable to encrypt capability, press 'E' and then enter. When you're finished, the list of 'Current allowed actions' MUST included 'Sign'. To go on, press Q, and then enter.

Now we're going to choose the length of our key.

```
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
```

The number inside the parenthesis is the default value -- if you press enter without typing anything, it will select that. However, we want our key to be longer -- set it to the maximum of 4096. The longer your key, the longer it will take to generate, encrypt, and decrypt things with it, but it will also be more secure.

The next step is setting the expiration date of our key.

```
Please specify how long the key should be valid.
         0 = key does not expire
       <n>  = key expires in n days
       <n>w = key expires in n weeks
       <n>m = key expires in n months
       <n>y = key expires in n years
 Key is valid for? (0)
```

Since this is our signing key, we want it to be valid forever. This is fine, since there are other ways to render a key invalid other than letting it expire. If you do set a different expiration date, do so. If not, just press enter to accept the default of a non-expiring key. You will be prompted if your selection is correct -- verify if it is and if so, type 'y' and press enter.

At this point, GnuPG will ask for your user ID information. The user ID is a set of metadata attached to the key -- data not relevant to the key, but rather describing it. This metadata will include your "Real Name", your email address, and a "comment". All of it can be changed later, but it is important to put metadata in that will be relevant. Many people will not regularly update the keys they store, and thus might use a key with out-of-date metadata, so you shouldn't rely on key metadata to communicate important information to people.

When you have entered all of your metadata, GnuPG will confirm it with you as such:

```
    You selected this USER-ID:
        "test key (key as of mm/dd/yyyy) <test@example.com>"

    Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit?
```

After you confirm the information you entered, GnuPG will ask you for a passphrase. This is used to protect your key (which is an actual piece of data in a file) as it resides on your disk. Choose your password according to the threat model you face -- make sure it is always secure, in case you have to move it elsewhere, but if your entire system is encrypted, you might have more basis for choosing a weaker passphrase. The passphrase on your key can be changed at any time after generation. Make sure it is at least 10 characters, and includes uppercase and lowercase characters, numbers, and symbols.

Once you give GnuPG a passphrase, it will generate your key. This operation requires a large amount of entropy -- you can help your system collect entropy by opening up a text editor and typing randomly, performing operations that utilize the hard disk, or running any other programs you have installed that collect entropy. If you do "type randomly", make sure it is truly random, and not in any discernible pattern (hint: asdf;lkjqwerpoiu is an example of non-random text). This step could take a long time (4096 is a large number of bits!) so be patient.

When the operation completes, you'll see a message similar to the following:

```
      gpg: key A6ADB947 marked as ultimately trusted
      public and secret key created and signed.

      gpg: checking the trustdb
      gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
      gpg: depth: 0  valid:   2  signed:   2  trust: 0-, 0q, 0n, 0m, 0f, 2u
      gpg: depth: 1  valid:   2  signed:   0  trust: 1-, 0q, 0n, 1m, 0f, 0u
      pub   4096R/A6ADB947 2009-07-02
            Key fingerprint = 81B9 09F1 510C 78BA 493E  AEB7 A565 B443 A6AD B947
      uid                  test key (key as of mm/dd/yyyy) <test@example.com>

     Note that this key cannot be used for encryption.  You may want to use
     the command "--edit-key" to generate a subkey for this purpose.
```

The parts of that you need to worry about are the key ID number (in this case, A6ADB947), the fingerprint, and the user ID or UID. The ID number is relatively arbitrary, but keyservers use it to identify your key. The fingerprint is what it sounds like -- just a method of identifying a particular key. You can use it to let people know they've downloaded the proper key from a keyserver, for example.

Section Two: Generating your Subkey

As GnuPG so nicely pointed out, our key can't be used for encryption. To fix that, let's edit our key and add an encryption subkey. To start, run the following command line:

```
       gpg --edit-key --expert <key ID>
```

The '<key ID>' in the above line can be replaced with anything that GnuPG can use to uniquely identify your key out of all the others you have. If you have only one key with the email address you used earlier, you can replace '<key ID>' with that email address. You can also use the hex ID. For the test key just generated, the following lines are acceptable:

```
      gpg --edit-key --expert test@example.com # uses email
      gpg --edit-key --expert "test key"       # uses "real name"
      gpg --edit-key --expert A6ADB947         # uses hex ID
```

You will be greeted by a screen like this:

```
gpg (GnuPG) 1.4.9; Copyright (C) 2008 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Secret key is available.

pub  4096R/A6ADB947  created: 2009-07-02  expires: never       usage: SCA
                     trust: ultimate      validity: ultimate
[ultimate] (1). test key (key as of mm/dd/yyyy) <test@example.com>

Command>
```

The command we will use first is the "addkey" command. It will generate a new key similar to how we generated the main key, but this time we will generate a "subkey" that will be tied to the primary key.

The first thing GnuPG will ask you for after you type 'addkey' and press enter is the passphrase for your key. You will need to supply that before going further.

After doing that, we'll see a familiar screen:

```
Please select what kind of key you want:
   (2) DSA (sign only)
   (3) DSA (set your own capabilities)
   (4) Elgamal (encrypt only)
   (5) RSA (sign only)
   (6) RSA (encrypt only)
   (7) RSA (set your own capabilities)
Your selection? 7
```

We again want to select '7', which will take us to another familiar screen:

```
Possible actions for a RSA key: Sign Encrypt Authenticate
      Current allowed actions: Sign Encrypt

         (S) Toggle the sign capability
         (E) Toggle the encrypt capability
         (A) Toggle the authenticate capability
         (Q) Finished

      Your selection?
```

This time, however, we want to make a key only useful for encryption. Disable signing and finish.

This will take us to a completely new and unheard-of place where we will gallop with Crypto Unicorns in fields of ciphertext. Just kidding, it'll take us to another screen we've been to before.

```
     RSA keys may be between 1024 and 4096 bits long.
     What keysize do you want? (2048)
```

We want to put 4096 as the length of this key, too. After setting the length, we'll move on to specifying the key lifetime --

```
      Please specify how long the key should be valid.
               0 = key does not expire
            <n>  = key expires in n days
            <n>w = key expires in n weeks
            <n>m = key expires in n months
            <n>y = key expires in n years
      Key is valid for? (0) 1y
```

As you can see, we've set our subkey to expire in one year, at which point we'll need to generate a new one. Depending on your needs, set the expiration date for less. GnuPG will confirm the expiration date with you when you choose.

After this, we're done setting up the key, and GnuPG will ask us one last time if we want to add this subkey. Since we do, we'll say yes, and GnuPG will generate another key. It needs just as much entropy as it did the first time, so type randomly, use the disks, fire up the webcam and make random faces at it, etc.

When GnuPG is done generating your subkey, it'll take you back to the main screen:

```
pub  4096R/A6ADB947  created: 2009-07-02  expires: never       usage: SCA
                     trust: ultimate      validity: ultimate
sub  4096R/214C8A86  created: 2009-07-02  expires: 2010-07-02  usage: E
[ultimate] (1). test key (key as of mm/dd/yyyy) <test@example.com>

Command>
```

Except this time, it'll show your new subkey in the list.

Section Three: Hardening your Key

Note: If you use RTC's Ubuntu remix, you will have this preference string by default, and can skip this section.

Before we're done editing our key, we have one more thing to do. Type the command 'showpref' to see your key's current preferences of encryption and hashing algorithms:

```
Command> showpref
[ultimate] (1). test key (key as of mm/dd/yyyy) <test@example.com>
     Cipher: AES256, AES192, AES, CAST5, 3DES
     Digest: SHA1, SHA256, RIPEMD160
     Compression: ZLIB, BZIP2, ZIP, Uncompressed
     Features: MDC, Keyserver no-modify

Command>
```

While these settings are adequate, we'd like to make them a little more secure. Sadly, this is a rather arcane process, but it only takes one command:

```
Command> setpref S10 S9 S8 S7 S4 S3 S2 H10 H9 H8 H11 H3 Z3 Z2 Z1
Set preference list to:
     Cipher: TWOFISH, AES256, AES192, AES, BLOWFISH, CAST5, 3DES
     Digest: SHA512, SHA384, SHA256, SHA224, RIPEMD160, SHA1
     Compression: BZIP2, ZLIB, ZIP, Uncompressed
     Features: MDC, Keyserver no-modify
Really update the preferences? (y/N) y
```

The lists of algorithms above are in order from most preferred to least preferred. People encrypting messages to us will use algorithms we prefer if they are able to.

Section Four: Sharing your Key

After you're done adding subkeys or hardening your preferences, save and exit from gpg's key editor (this is one command):

```
Command> save
```

Now we have a fully functional public key, and we can send it off to one of the GnuPG keyservers. We will need to use our hex key ID, not our user ID, to identify our key:

```
    gpg --send-keys <Key ID>
```

In our case, the command line will be as follows:

```
   gpg --send-keys A6ADB947
```

This will tell GnuPG to upload our key to one of the many OpenPGP keyservers. All of the major keyservers synchronize with each other on a regular basis, so uploading your key to one of them will soon share it with all of them.

Now you have a fully operational key that others can download from a keyserver and use to send encrypted messages to you, or to verify that you are indeed the source of a given message.

Installing Anonymity Technology

All of the following programs are installed by default on Radical Ubuntu. This section outlines how to install them on a normal Ubuntu system as part of a radicalization process.

Tor

Tor is a low-latency anonymity network designed for anonymous internet use. On the RTC Ubuntu remix, Tor is used for anonymous web browsing and instant messaging.

To install Tor, first add the two repository lines listed below to your Apt configuration. This can by done by putting them in the bottom of the /etc/apt/sources.list file, putting them in a separate file in /etc/apt/sources.list.d/, or by using System>Administration>Software Sources.

```
  deb     http://mirror.noreply.org/pub/tor <DISTRIBUTION> main
  deb-src http://mirror.noreply.org/pub/tor <DISTRIBUTION> main
```

...where <DISTRIBUTION> is replaced with your version of Ubuntu -- probably 'jaunty'. To demonstrate, you could add the repository to a file in /etc/apt/sources.list.d/ by running the following command line:

```
echo '  deb     http://mirror.noreply.org/pub/tor <DISTRIBUTION> main
  deb-src http://mirror.noreply.org/pub/tor <DISTRIBUTION> main' | sudo tee
/etc/apt/sources.list.d/tor.list
```

Next, add the repository's public key to apt-keyring. This can be done with the following command line.

```
gpg --keyserver keys.gnupg.net --recv 94C09C7F && gpg --export 94C09C7F | sudo apt-
key add -
```

Refresh the package list, and then install the 'tor' package. On the command line, this would be:

```
sudo apt-get update && sudo apt-get install tor
```

i2p

i2p is another anonymous Internet overlay much like Tor. However, while Tor seeks to serve best as an outproxy network, i2p focuses more on its inproxy capabilities, creating a self-contained anonymous network.

i2p is best installed with the RTC i2p debian package. Download it from the anonnet section on an RTC mirror, and install it by double-clicking on the downloaded package and using GDebI.

If you cannot download the RTC i2p package, download the i2p grapic installer from the i2p website. Currently, this is <http://www.i2p2.de/download>. The downsides of this method are that you will not have i2p started automatically and may not be ablef to easily remove it.

Freenet

Freenet is an anonymous, censorship-resistant peer to peer network. Files stored on Freenet cannot be manually deleted by anyone -- they will only be gradually removed after years of disuse.

Freenet is best installed with the RTC Freenet debian package. Download it from the anonnet section on an RTC mirror, and install it by double-clicking on the downloaded package and using GDebI.

If you cannot download the RTC Freenet package, download the Freenet graphic installer following the directions on the Freenet website. Currently, this is <http://freenetproject.org/download.html>. The downside of this method is that you will not have Freenet started automatically and may not be able to remove it easily.

GNUnet

GNUnet is, like Freenet, an anonymous peer-to-peer filesharing network. It is not as censorship-resistant as Freenet, but it is faster and more resilient to blocking.

To install GNUnet, first add the two repository lines listed below to your Apt configuration. This can by done by putting them in the bottom of the /etc/apt/sources.list file, putting them in a separate file in /etc/apt/sources.list.d/, or by using System>Administration>Software Sources.

```
  deb     http://ppa.launchpad.net/teamgnunet/ubuntu <DISTRIBUTION> main
  deb-src http://ppa.launchpad.net/teamgnunet/ubuntu <DISTRIBUTION> main
```

...where <DISTRIBUTION> is replaced with your version of Ubuntu -- probably 'jaunty'. To demonstrate, you could add the repository to a file in /etc/apt/sources.list.d/ by running the following command line:

```
echo '  deb     http://ppa.launchpad.net/teamgnunet/ubuntu <DISTRIBUTION> main
  deb-src http://ppa.launchpad.net/teamgnunet/ubuntu <DISTRIBUTION> main' | sudo tee
/etc/apt/sources.list.d/tor.list
```

Configuring Firefox

Today, a web browser is one of the most vital components of an operating system. Sadly, it is also typically one of the most vulnerable -- at least, by default. Through some configuration, we can create a browsing environment that is entirely locked down.

Note for RTC Ubuntu users

If you use the RTC's Ubuntu remix, you can set up your anonymous Firefox profile using the program 'setup-anon-firefox', from the 'rtc-utils' package. You should still read the section on good browsing habits at the end of this article, however.

Getting started

The first thing we'll need to do is make separate Firefox profiles for us to use -- one will be for non-anonymous browsing, and one will be for secure, anonymous browsing. The main difference between the two is that the anonymous profile will store as little as possible data on the disk.

To being, open up a terminal and run the command:

```
'firefox -Profilemanager'
```

This will open up a window where you can add a new profile. For the purposes of this document, we'll assume you named it "anonprof", but any name will do.

When you've added the new profile, go to Firefox's entry in the main menu, right click it, and select "Add this item to desktop" to copy it to your desktop. Once it's there, copy it again.

Select the first copy, right-click it, and select "properties". Select the field "Command" and change it from

```
firefox %u
```

to

```
firefox -P default -no-remote %u
```

The -P option is used to specify the profile used -- in this case, the default profile. The -no-remote flag is used to allow this firefox to launch even if you have another instance of firefox running.

Close the launcher, select the other one on your desktop, and enter the "Properties" window again. This time, change the "Command" field from

```
firefox %u
```

to

```
firefox -P anonprof -no-remote
```

When you've done that, click on the icon in the properties menu. You'll open the icon selection screen. Pick another icon to identify this firefox as clearly different from the first.

If you want, you can drag the new icons to the panel, or keep them on your desktop.

Configuring Firefox

Now that you have the launcher set up for your new profile, start Firefox, and select Edit>Preferences to begin configuring it be secure.

On the "Main" tab, select "Show a blank page" in the drop-down menu next to "When Firefox starts", and set the text in "Home Page" to read

```
about:blank
```

This will keep sites from gaining information about you based on your homepage settings.

Next, check the "Always ask me where to save files" button. This will prevent you from accidentally downloading a file to somewhere it shouldn't go.

In the "Content" tab, uncheck the "Enable Java" box, and click the "Advanced" button next to "Enable Javascript". Once there, uncheck all items except "Disable or replace context menus". If you feel so inclined, you can disable Javascript entirely, which will save you a great deal of trouble and make you a good deal more secure.

Next, go to the "Privacy" tab. Uncheck all items under the "History" and "Cookies" categories. Under the "Private data" category, check "Always clear my private data when I close Firefox" (if you want, set it to ask you when it does so). Click the "Settings" button, and check the box next to everything except "Saved Passwords".

Now click the "Security" tab. Uncheck the options "Tell me if the site I'm visiting is a suspected attack site" -- while this may be counter-intuitive, every time you go to a site with this option enabled, Firefox queries Google for a database of known malicious sites -- indirectly giving your browsing history to Google.

Under "Passwords", check "Remember passwords for sites", and "Use a master password". Set the master password now, and make sure it is secure. It should be longer than 10 characters and use uppercase and lowercase characters, numbers, and symbols.

Now go to the "Advanced" tab, and then the "Network" subtab. Set the "offline storage" number to 0, and check "Tell me when a website asks to store data for offline use". Next, click on the "Update" tab, and make sure everything is unchecked.

We're now finished configuring Firefox via the preferences menu, but our system is not yet ready for anonymous browsing. We now have addons to install and configure.

Installing add-ons

The following Ubuntu packages need to be installed:

```
mozilla-noscript
torbutton-extension
adblock-plus
```

Noscript is an addon which blocks all forms of client-side scripting languages on a per-domain basis. Since client-side executing code can potentially bypass browser privacy settings, it is imperative that you only enable script execution on sites you trust and have a secure (https) connection to.

Torbutton is an addon from the Tor Project which allows you to toggle between Tor and Non-Tor browsing. For our purposes, we should always leave it in Tor mode. Torbutton also performs other things such as modifying your user-agent and sanitizing requests to strengthen anonymity.

Adblock blocks ads. This is a good thing both because ads take up bandwidth and because they quite often include tracking code. Adblock is also very fine-grained, so you can block individual files on a site -- this can come in handy for more than just ads.

These addons come with sensible defaults and require very little configuration. If you wish, you can look through the Torbutton addons and see if you find anything you wish to enable. The one thing we should do with addons once they are installed is enable Tor with Torbutton.

Disabling plugins

Plugins are the bane of anonymous browsing, as typically, they do not respect browsers' privacy settings. We will take a "scorched-earth" approach to dealing with them.

In Firefox, select "Tools", then "Add-ons" from the menu. Click the "Plugins" tab and disable everything except the printing plugin and the "Default Plugin".

heading: Safe Browsing Habits

You now have a fully anonymized Firefox profile. Remember, however, that a secure browser is only as secure as its user. Don't expect to be anonymous if you download addons without thinking, or enable javascript on arbitrary sites.

You should only enable Javascript with Noscript on sites that you have a secure connection to (https).

A major problem with regards to privacy on the Web is client-side executing code. This includes things like Java, Javascript, and Flash. The problem with these scripting languages is that they execute in the browser, or worse, in external plugins. The fact that they execute on the user's computer means they can search for identifying information or connect back to a hostile server. This is why we've disabled Flash and Java, and severely crippled Javascript.

The most pressing threat to anonymity is you. Always remember that loose lips sink ships, and loose fingers on a keyboard bypass the most paranoid of browser configurations. That doesn't rhyme, but it's probably more true than the first.

Your anonymous browser is your gateway to free information. You can be reasonably assured of your safety and anonymity if you stay within its bounds, but always keep your security as a top priority.

Configuring Mozilla Thunderbird

Having GnuPG installed and a key generated is of little use unless a properly configured email client or other client programs make use of them. Mozilla Thunderbird is a popular, cross-platform, free software mail client, and with the Enigmail plugin, it possesses powerful GnuPG capabilities.

Installing Thunderbird

If you are using the RTC's Ubuntu remix, Thunderbird will be installed by default, as will enigmail. If you're using stock Ubuntu, install the following packages:

```
thunderbird
enigmail
```

Configuring Thunderbird

Before you being using Thunderbird for encrypted mail, you first have to configure both Thunderbird to fetch your mail, and Engimail to use your keys.

Email configuration is somewhat idiosyncratic and varies based on your email host. Refer to your email provider's documentation on how to configure a mail client. If your email provider does not provide access via the POP or IMAP protocols, you will need to create an account on a provider that does. Both GMail and riseup.net are compatible with Thunderbird, and provide documentation.

To configure Enigmail, select "Edit", then "Account Settings" from the Thunderbird menu. From the list of sub-entries under your email account, select "OpenPGP Security".

First, make sure the check-box labeled "Enable OpenPGP support (Enigmail) for this identity". Select "Use specific OpenPGP ID" from the menu below -- click the "Select Key" button and choose your key. Check all the boxes necessary to sign non-encrypted and encrypted messages by default, and check the "Always use PGP/MIME" box.

Now click the "Advanced" button to configure Enigmail further. Check the "Display Expert Settings" box, then go the "Advanced" tab. Check all the boxes shown, except the last (if you do not use IMAP). Next, go to the "Key Selection" tab, and select "By rules and email addresses" for the query "How should we choose the keys". Next, go to the "Sending" tab, and check all the visible boxes.

At this point, Engimail should be successfully configured. Test it out by sending an encrypted, signed message to a friend.

Configuring Pidgin for anonymous and secure messaging

Pidgin is a full-featured, cross-platform, and most importantly, free software, IM and chat client. It supports just about every protocol you could care to name -- AIM, MSN, YIM, Jabber, IRC, ICQ, SILC, and more. Pidgin is installed by default on Ubuntu (at least for now), and it's simple to configure for security.

Software to install

The first thing you'll need to have installed is Tor, since it's the anonymity network we'll be using to anonymize Pidgin. Refer to <http://www.torproject.org> or other RTC documentation for more information on installing Tor, if it isn't already installed.Pidgin has a very mature and flexible plugin framework, allowing developers to write plugins to handle any additional functionality not present in Pidgin. The two most relevant plugins are the IRC More plugin, which adds additional options for Pidgin accounts using the IRC protocol, and Off-the-record messaging, a plugin that provides strong encryption and buddy verification for Pidgin.

On RTC's Ubuntu remix, these two plugins are installed by default. On standard Ubuntu installs, however, you will have to install them yourself. They are in the packages 'pidgin-plugin-pack' and 'pidgin-otr' respectively.

After the plugins are installed, you'll have to enable them in the Plugins menu in the buddy list. To get there, select "Tools" from the buddy list window, then select "Plugins". To enable the plugins, check the box next to them. Neither of them require any extra configuration.

Configuring Pidgin for security

Pidgin doesn't have much in the way of global configuration, but there are a few switches we'll need to flip to get the maximum security out of our IM client. To start, go to the Preferences entry in the Buddy List, under Tools.

First, go to the "Conversations" tab, and uncheck "Show formatting on incoming messages" and "Enable buddy icon animation". While not strictly helping our anonymity, rendering bugs are a common flaw that can be exploited, and disabling these parts of Pidgin will decrease our likelihood of being affected.

Next, go to the "Network" tab. Uncheck "Autodetect IP Address", and then set the value of "Public IP" to "0.0.0.0". This will prevent Pidgin from advertising your real IP to the world. Also uncheck "Enable automatic router port forwarding".

You'll notice that there is a neutered "Configure Proxy" button. This is because on Ubuntu, Pidgin gets its proxy configuration from GNOME. We will have to set proxies on each individual account to work around this -- this is fundamentally a bug in Pidgin, but it will be eventually fixed.

Next, head to the "Logging" tab, and uncheck everything. We don't want any of our conversations or status changes to be written to the disk, just in case our encryption is compromised or (if we're really horrible people) we don't have any in the first place.

Last, go to the "Status / Idle" tab. Select "Never" for "Report idle time", and uncheck "change status when idle". This will prevent your IM client from divulging when you are away from your computer, so that your actions cannot be correlated with the actions of your instant messages (and therefore you cannot be linked as easily).

The most important parts of securing pidgin couldn't be covered here, because Pidgin doesn't provide those options in the global configuration. Instead, we will have to activate them on a per-account basis.

Creating an account

Since Pidgin derives most of its global configuration from GNOME, it is depressingly hard to set sane defaults. This may be fixed in the future, but for the time being, most of the locking-down process occurs inside the account creation system.

To add an account to Pidgin, select "Manage Accounts" from the "Accounts" menu in the buddy list, then press the "Add" button on the window that appears.

Pidgin supports a wide array of protocols, each with their own idiosyncratic settings, some of which can be used to make them more secure. However, covering them all in sufficient detail would be an incredibly lengthy process, so this guide will remain protocol-agnostic in nature.

The first thing you should do on the account creation screen click the "Advanced" tab. There will be a number of options depending on the protocol, but at the bottom, there will be the proxy information. Select "Socks v5" from the drop-down menu, set the value of the "Host" field to "localhost", and set the value of the "Port" field to "9050". This will configure that account to use Tor as its proxy, anonymizing the source of your communications. Nothing else is required to get Pidgin to use Tor for an account -- however, this step must be repeated for each account.

While you are using Tor, the operator of the exit node (the Tor server at which you exit the Tor network) will be able to read your network traffic. While most operators are trustworthy individuals, security should never rely on trust. Encrypt all of your conversations with OTR, and if possible, use SSL or TLS on your account. The switch for SSL or TLS will likely be under the

"Advanced" tab in the account editing window. Be warned: For decentralized services like IRC or XMPP, the server may not support SSL or TLS.

For most protocols, this will be enough to reasonably secure the account for now. However, on IRC, the IRC More plugin allows us to further harden the settings to prevent data leakage. On the "Advanced" page, change the values "Username", "Real name", "Auth Name", and "CTCP Version Reply" to garbage values, or "empty" strings, or something similar. The idea is to reduce the information about your system that is leaked to the end party. By default, these values are taken from your system environment and thus can be used to reduce the anonymity set.

OTR verfication

Off-the-record functions very much like PGP encryption -- both are based on a public-key (asymmetrical) cryptosystem, and both need users to verify that public keys belong to their purported owner before trusting them.

To verify an OTR key, first establish an OTR session with a buddy with an OTR-implementing client (these include Pidgin, Adium, or Trillian). Pidgin will tell you that you've established an "unverified" conversation. To verify the conversation, right-click the OTR icon in the lower right of the screen (which should read "unverified conversation") and select "Authenticate buddy".

OTR in Pidgin allows three modes of authentication -- question and answer, shared secret, and manual verification. Question and answer and shared secret are exactly what they sound like, and are the "simplest". However, they also have the most potential for security holes. If you communicate the shared secret or answer to the secret question with your friend over the phone, and the phone is wiretapped, or in person, with an eavesdropper nearby, a hostile adversary could give the correct answer or secret and fool you into accepting their key.

The best way to be sure that a buddy is authentic is by selecting the "Manual fingerprint verification" option. This will show you your fingerprint, and the fingerprint that your buddy is sending you.

Write down both fingerprints, and have your buddy do the same for theirs. Even if you are observed while comparing them, generating a key with the identical fingerprint would be nearly impossible for a hostile party to do. As such, you will be able to fully verify the authenticity of the key, and therefore, the security of the conversation.

Free Software Alternatives

AIM/MSN/Whatever → Pidgin

Photoshop → The GIMP

Illustrator → Inkscape

InDesign → Scribus

MS Word → OpenOffice.org

IE/Opera → Firefox

Outlook → Evolution or Thunderbird

Windows or Mac → GNU/Linux