



Secure RSA Authentication for Linux

Document v1.0 - Last edited 4/26/2008 by Chris Sherwood, SureTeq, Inc.

Let SureTeq provide you with FtOCC certified trixbox Pro and/or trixbox CE **hourly support!** Hourly rates and support contracts are available. For pricing information and contact details, please [contact us!](#)

I recently attended a class on trixbox Pro, and they mentioned that if you are going to be connecting remotely to the Linux CLI of your trixbox system (applies to pretty much any Linux box though), it is best to use secure RSA private/public key pairs for authentication if you have the SSH port (TCP 22) open to the public Internet. This is a GREAT idea, and strongly recommended, but just how do you do that? Here's how...

First, an explanation of what the heck I'm talking about. Typically when you connect via SSH to Linux, you are prompted for username and password...this is fine, as long as you have strong passwords, but is still susceptible to brute force password attempts. By using a secure key pair, you are explicitly telling Linux who can connect by ensuring that you **HAVE** to have the private key in order to connect. You can also then choose to have a password, or passphrase, in addition to the key pair (but you don't **NEED** to...it all depends on your desired level of security, or your level of paranoia).

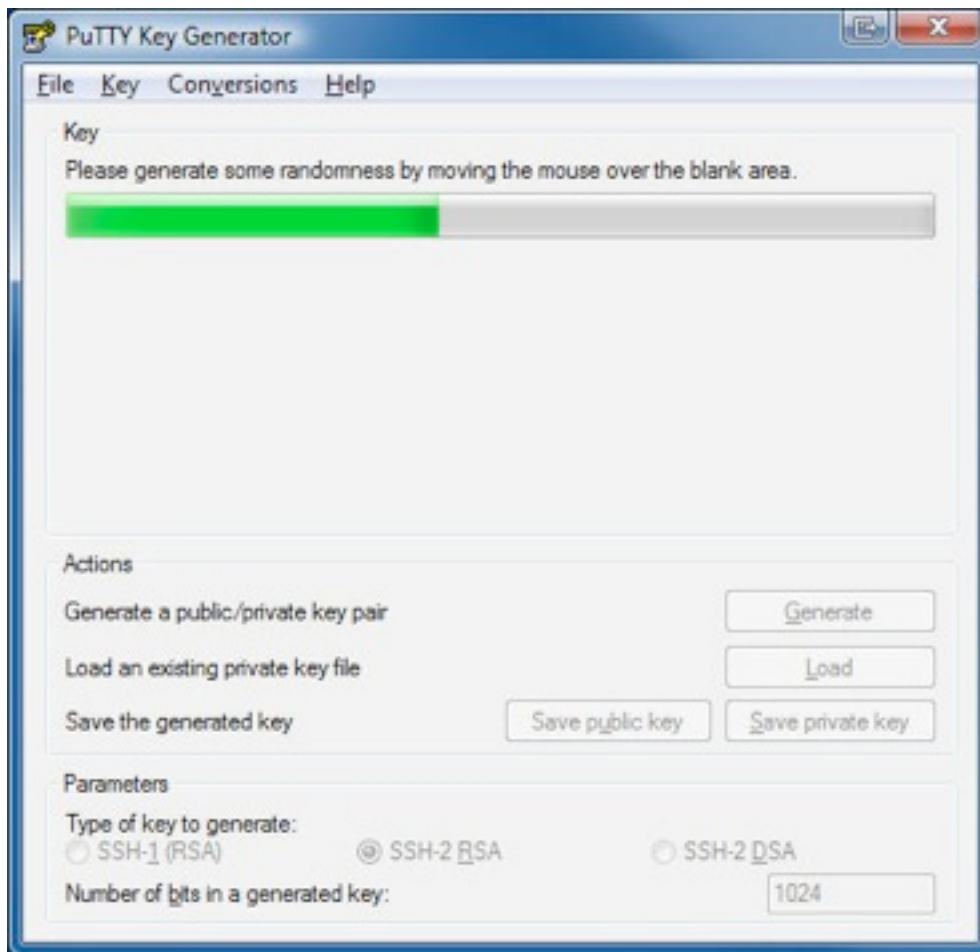
The key pair is generated on the client system. The private key stays with the client, and the public goes to the server (or servers) you wish to connect to. Let's start by generating the key pair.

I prefer to use SecureCRT for my terminal sessions, but most people use PuTTY (the free SSH client available at <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>). I'll show you how to generate the key pairs with both. The only difference I have found is that SecureCRT allows you to store the passphrase in addition

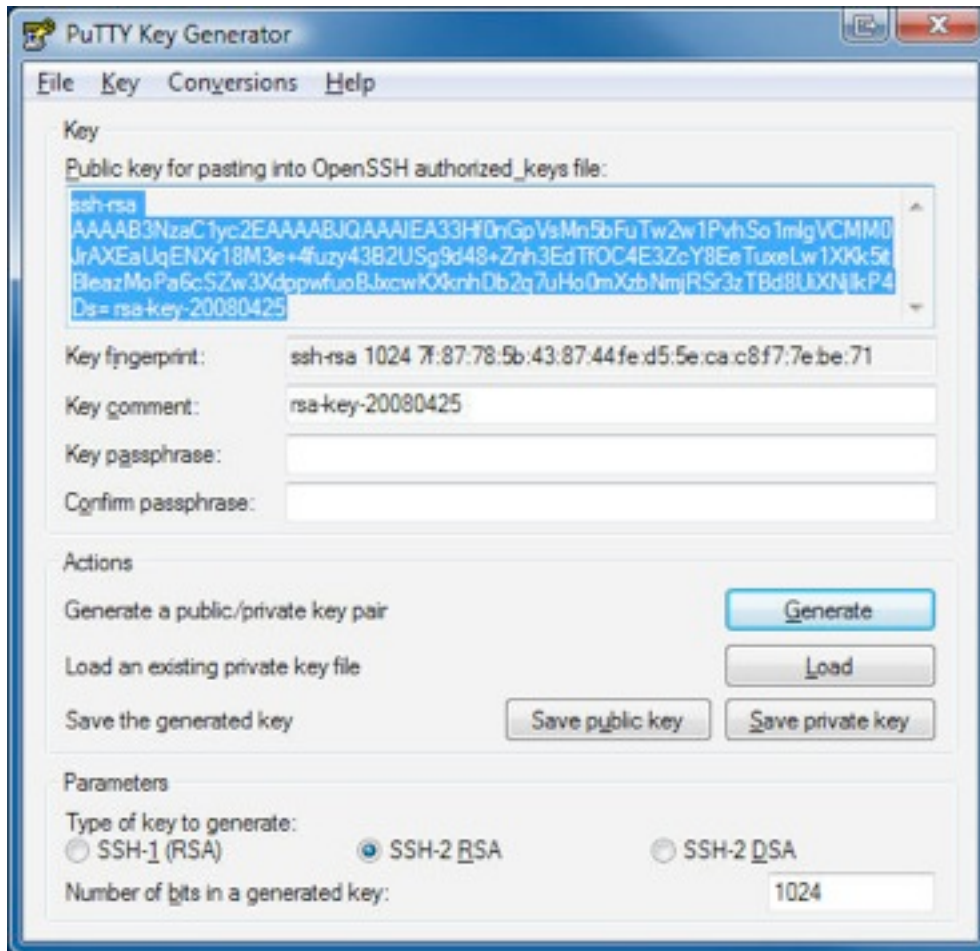
to the public/private key pair...I like this a lot because I'm lazy, and I don't want to have to type the passphrase every time I connect.

PuTTY Key Generation

First, you will have to download both the PuTTY client and PuTTYgen.exe, the PuTTY key generation program. We start with PuTTYgen.exe. Double click on it, and click 'Generate.' It will have you move your mouse around to generate some randomness for the key generation.



Once you have moved the mouse around enough, you will see your key, plus some new options.



You can change the key comment to whatever you want...I usually leave it default. The next two fields are for the passphrase. This can be a password, or a phrase...I'm not sure what the limit is on the phrase, but it's pretty long. It's recommended to use something that you're going to remember such as a song lyric. DO NOT use your root password as the passphrase for the best security.

I'm gonna set my passphrase to 'that's the way uh huh uh huh I like it uh huh uh huh'. (without the quotes). Once you have typed it in twice, make sure that your settings at the bottom are set to 'SSH-2 RSA' and 1024 bits in a generated key. Linux (or OpenSSH specifically) will take either RSA or DSA, but I just leave it default. I'm sure there is a crazy technical difference between the two, but I don't know if I even want to know what it is...for our purposes, it doesn't matter.

Time to save your public and private keys. First, click the 'Save public key' button and save your public key on your computer somewhere. I always append .pub to the end of my public keys so I know which one is which.

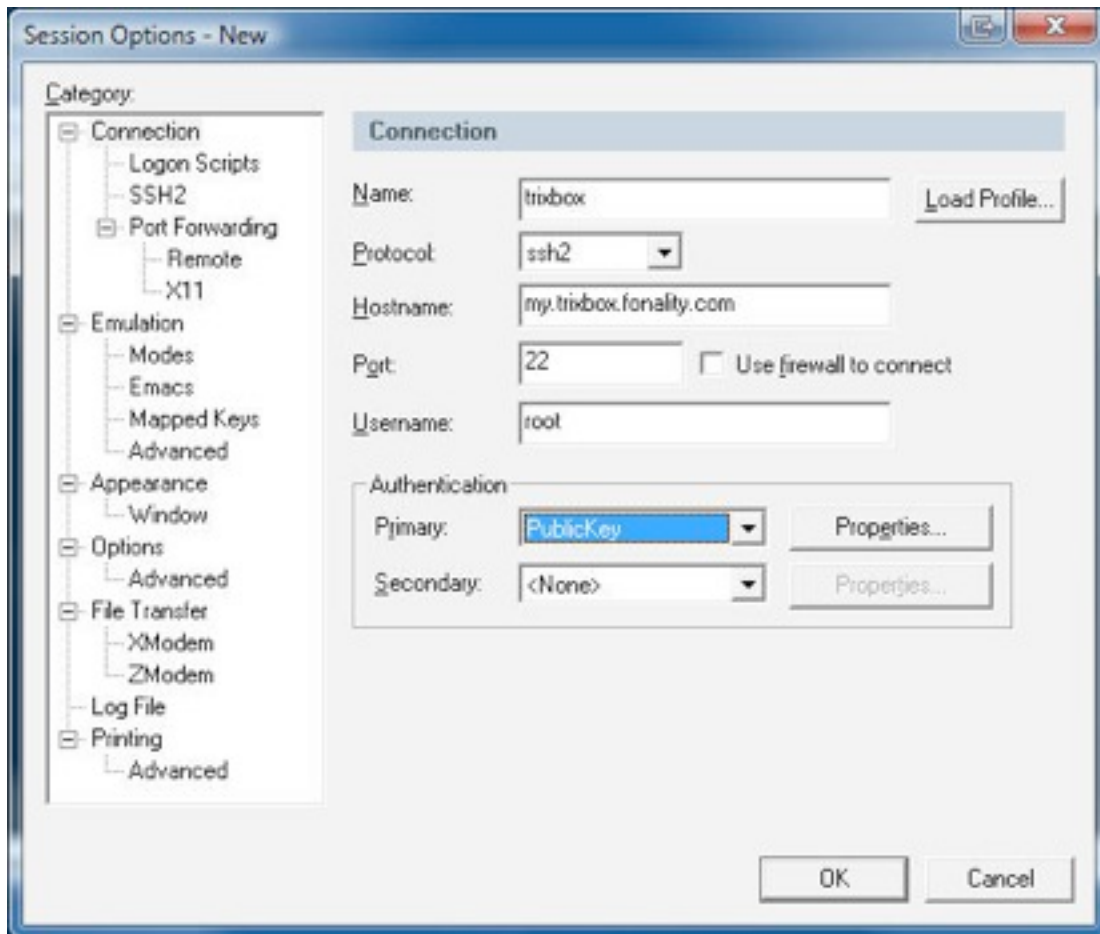
Next, click 'Save private key' and save your private key...these can be saved in the same location since PuTTYgen adds the .ppk extension automatically.

*** NOTE: VERY IMPORTANT!! Back up your public/private key pair somewhere where you won't lose it. If you have your Linux system set to require keys, and you lose it, you're outta luck.

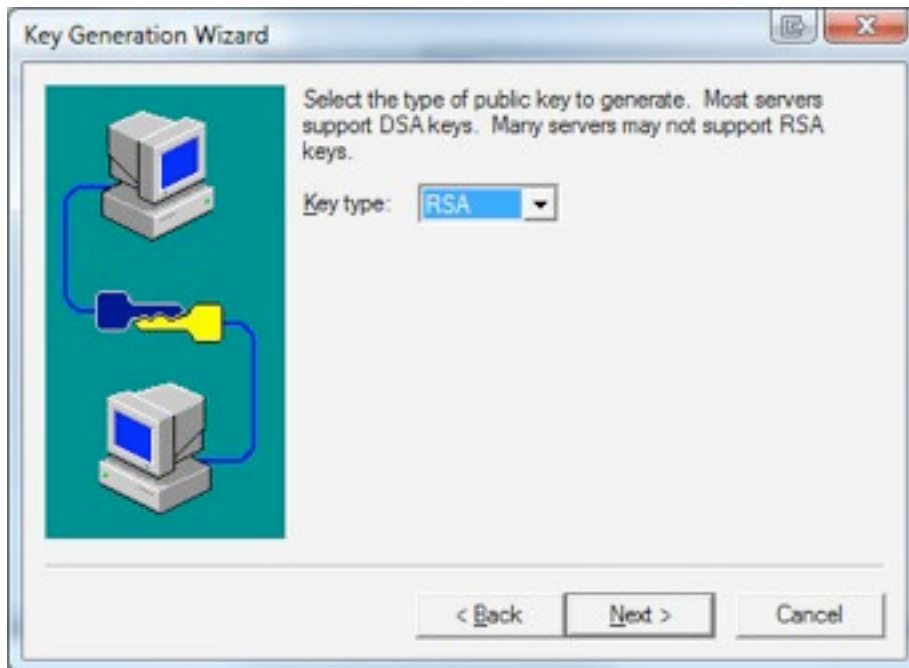
Ok...so now we have our public and private key pair...what the heck do we do with it? Skip down to 'Installing the key pair.'

SecureCRT Key Generation

To create a public/private key pair in SecureCRT, open up SecureCRT and start a new session, or open the options for an existing session. Under 'Authentication,' change 'Primary' to 'PublicKey.' Then click 'Properties.'



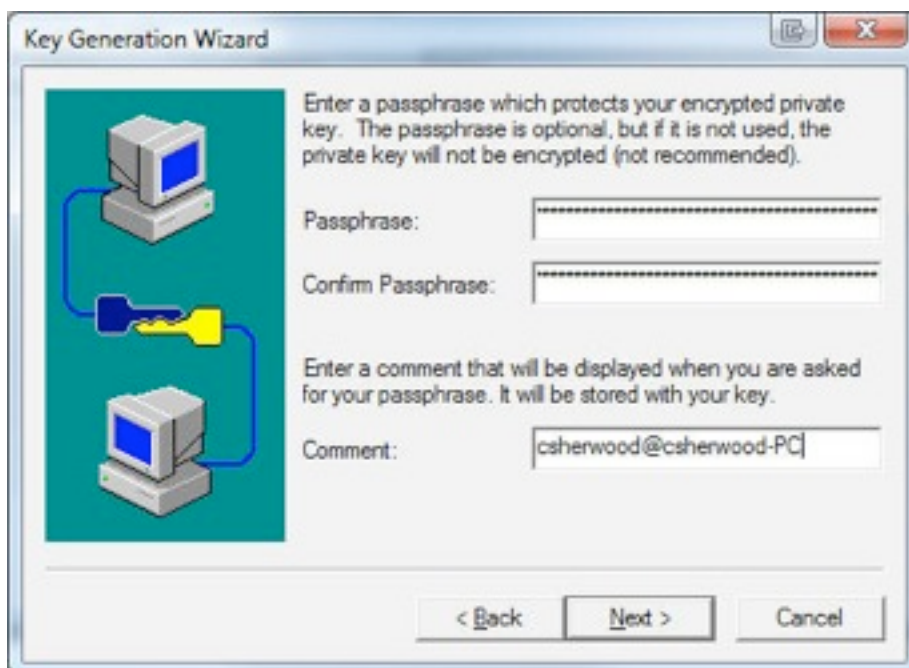
In the Properties page, click 'Create Identity File...!' and the key generation wizard starts. Click 'Next' to get started.



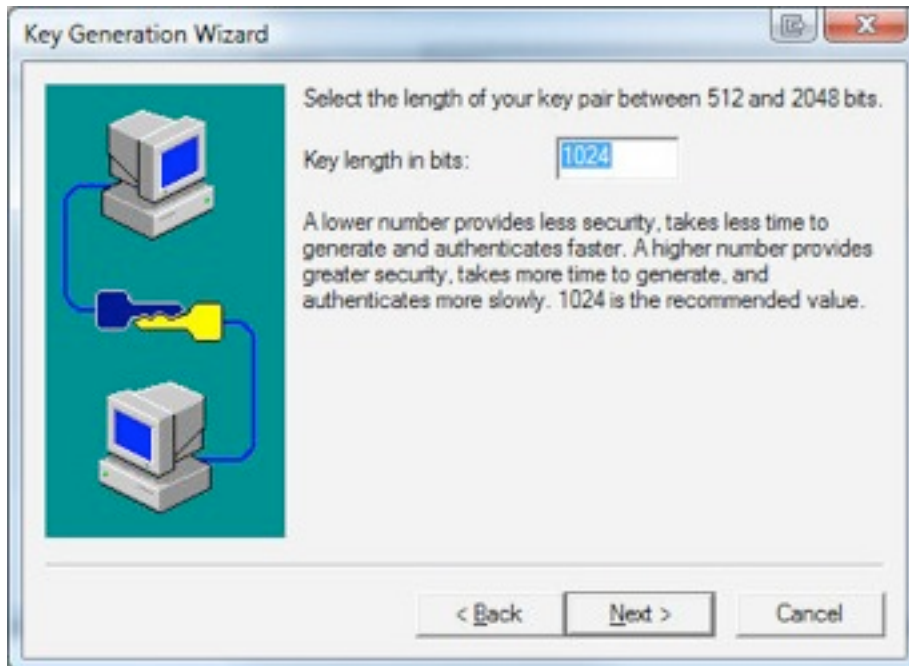
Select RSA as the Key Type and click 'Next.'

Next, you are prompted to enter a passphrase twice. This can be a password, or a phrase...I'm not sure what the limit is on the phrase, but it's pretty long. It's recommended to use something that you're going to remember such as a song lyric. DO NOT use your root password as the passphrase for the best security.

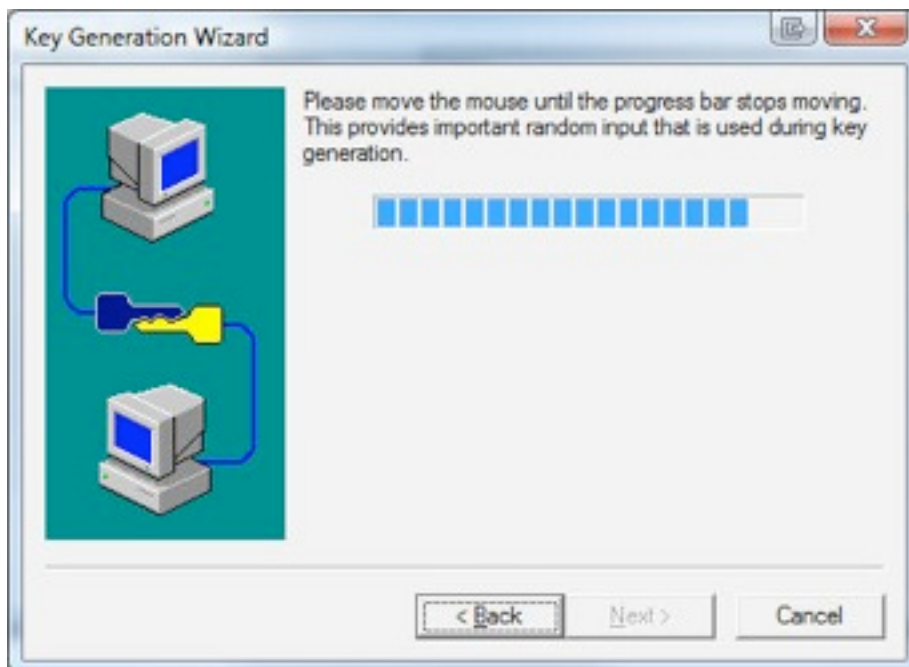
I'm gonna set my passphrase to 'that's the way uh huh uh huh I like it uh huh uh huh'. (without the quotes). Type it twice and then type whatever comment you would like (this doesn't really matter, so I leave mine default). Click 'Next.'



Next, it will ask you for your desired key length in bits. Take the default '1024' and click 'Next.'



Now it's time to generate some randomness for the key file by moving your mouse around. Once the bar has filled up, your key pair will be generated.



Once it has been generated, click 'Next.' You will now be prompted for the location and filename for where to save your public and private key. Note where they are being saved, and click 'Finish.' When asked whether or not you want to upload the following

key, say NO! We're going to do this manually since the key is not yet formatted properly for Linux.

Installing the key pair on the Linux server

Ok, so you should now have a public and private key on your system. The private key is going to stay with your system, and the public key is going to be given to whichever servers you would like to connect to.

First, get your public key uploaded to the Linux server in any way you are comfortable with...FTP works...I prefer to use WinSCP. One item to note while I'm talking about WinSCP is that once you have your keys, WinSCP can also utilize them. Upload the key to ~/.ssh. This is root's home directory, in the .ssh subfolder.

*** Note: If you are using trixbox, you already have this directory. If it doesn't exist, you can create it by changing to the root home directory (cd ~), and then manually adding the directory (mkdir .ssh).

We need to get your key into the 'authorized_keys' file, but first it needs to be formatted properly. If you do 'cat <public key filename>', you should see something like this:

```
---- BEGIN SSH2 PUBLIC KEY ----
Comment: "rsa-key-20080425"
AAAAB3NzaC1yc2EAAAABJQAAAIEA33Hf0nGpVsMn5bFuTw2w1PvhSo1mlgVCMM0J
rAXEaUqENXr18M3e+4fuzzy43B2USg9d48+Znh3EdTfOC4E3ZcY8EeTuxeLw1XKk5
itBleazMoPa6cSZw3XdppwfuobJxcwKXknhDb2q7uHo0mXzbNmjRSr3zTBd8UiXN
jIkP4Ds=
---- END SSH2 PUBLIC KEY ----
```

Basically, you will need to delete the first two lines, delete the last line, and get JUST the key on a single line with 'ssh-rsa' in front of it so that it looks like this (font is a little small, but it's 'ssh-rsa' (space) (public key)):

```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAIEA33Hf0nGpVsMn5bFuTw2w1PvhSo1mlgVCMM0JrAXEaUqENXr18M3e
+4fuzzy43B2USg9d48+Znh3EdTfOC4E3ZcY8EeTuxeLw1XKk5itBleazMoPa6cSZw3XdppwfuobJxcwKXknhDb2q7uHo0mXzbNmjRSr3zTBd8UiXNjIkP4Ds=
```

Open the file with 'nano <public key filename>' and you will be able to edit it properly. CTRL+X to save followed by 'Y'.

Now that your RSA key is properly formatted, you can add it to the 'authorized_keys' file. This will tell Linux that it is allowed to authenticate the private side of the public key in the file. To do so, do the following:

```
cat <public key filename> >> authorized_keys
```

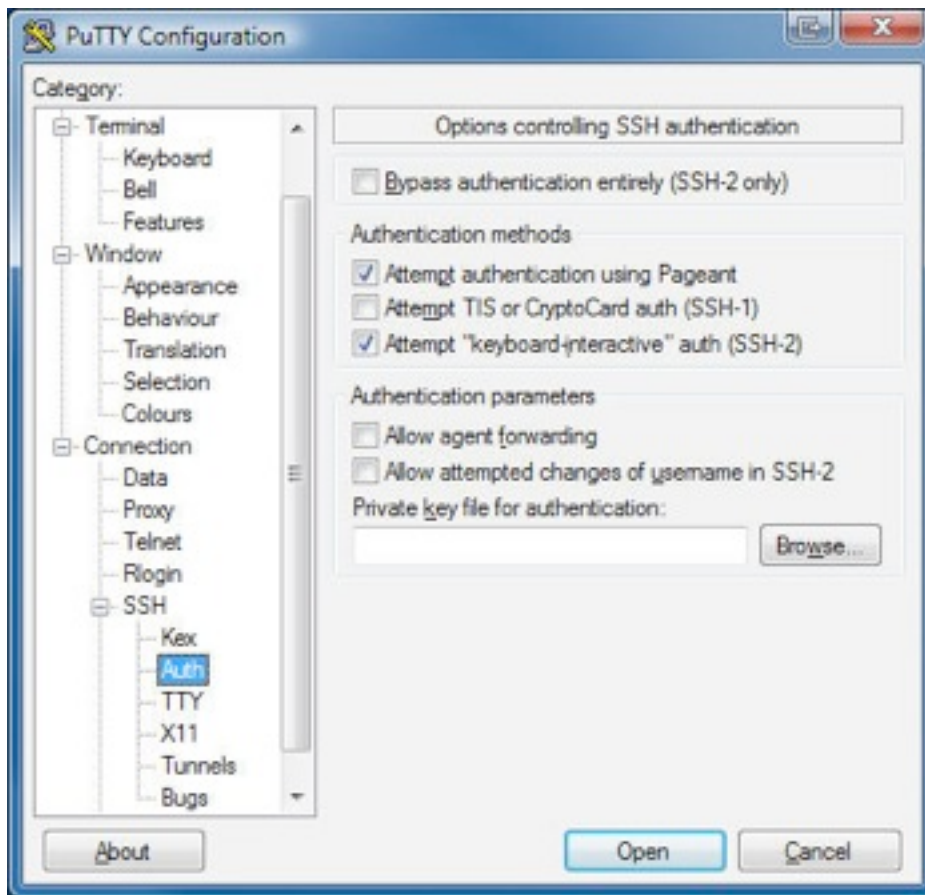
So if your public key is named 'public.pub' your command would look like this:

```
cat public.pub >> authorized_keys
```

If you now 'cat authorized_keys,' you should see your new key at the end of the file. Each key in the file should be on it's own line. Next, it's time to test out connectivity...let's start with PuTTY...if you're using SecureCRT instead, you can skip down to the SecureCRT section.

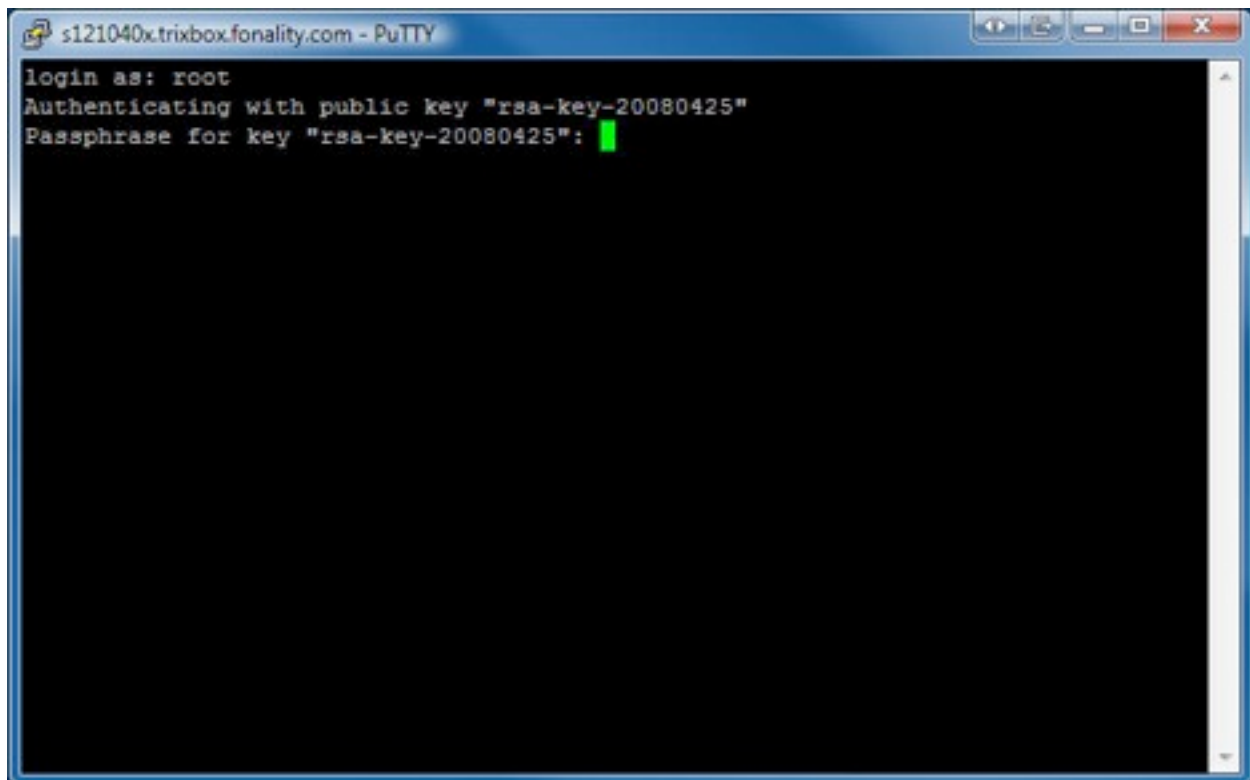
Connecting via PuTTY using your key pair

Open PuTTY. Let's assume that you're starting up a new session. Enter in the hostname, and then click 'SSH --> Auth' in the right hand 'Category:' section.



Click 'Browse' and find your private key (it should end in .ppk if you created it with PuTTYgen. Leave all other settings default and click back to 'Session' in the 'Category:' section. At this point, you should type in a name for your new session below 'Saved Sessions,' and then click the 'Save' button so that you don't always have to browse for your key.

Click 'Open.' You will be prompted with 'Login as:' Enter in your Linux username...typically, this is going to be for the root user. You will then be prompted for your passphrase.



Type your passphrase and you're connected! (NOTE: If you opted to not have a passphrase, it won't ask...you are simply connected after you type the username.)

Connecting via SecureCRT using your key pair

When you are using SecureCRT, start a new session, or edit an existing session. Put in all pertinent hostname and username information and change Protocol to SSH2. Under 'Authentication,' select 'PublicKey' as the primary method of authentication and click 'Properties.' If this is the same session you were already working with above, your identity file should already be filled in...if not, just click the browse button '...' and find your private key. Click OK.

Click 'OK' again on the session options, and then 'Connect' to connect to your new session. A window will pop up and you will be prompted for your passphrase. Type it in, and you're connected. With SecureCRT, you should only be prompted for your passphrase the first time you connect. That's it! Now onto the last step in securing your server.

Locking down OpenSSH to ONLY accept key authentication

The final step in securing your Linux box is to turn off password authentication and enable private key authentication only. Before we get to that though, let's lock down our public keys so that only the root user has access to them.

```
cd ~/.ssh
chmod 700 ~/.ssh
chmod go-rwx ~/.ssh/*
```

Now, let's configure OpenSSH. There are really only two settings we're concerned with...turning off password authentication, and turning on private key authentication.

```
cd /etc/ssh  
nano sshd_config
```

Find the uncommented line that says:

```
PasswordAuthentication yes
```

and change it to:

```
PasswordAuthentication no
```

Next, uncomment the following and set them to yes:

```
RSAAuthentication yes  
PubkeyAuthentication yes  
AuthorizedKeysFile .ssh/authorized_keys
```

CTRL+X to exit and Y to save.

Now, restart SSH with:

```
service sshd restart
```

That's it! You can now try it out by trying to connect via SSH 2 with password authentication...you will be denied and told that the server doesn't accept that form of authentication. When you try it with your key, you'll get right in. Nice. Now, you should have no qualms about keeping your SSH port exposed to the big bad Internet.
