

# Securely Copying Files with SSH

author: Brian Rectanus <*Brian.Rectanus@vt.edu*>

May 2002



## Contents

<b>1</b>	<b>Overview</b>	<b>1</b>
<b>2</b>	<b>The Different Secure Shell Protocols and Servers</b>	<b>1</b>
<b>3</b>	<b>SSH Client for Windows</b>	<b>1</b>
3.1	General Usage/Setup . . . . .	2
3.2	Using SFTP to Transfer Files (recommended) . . . . .	2
3.3	Using FTP via SSH to Transfer Files . . . . .	3
<b>4</b>	<b>General scp Command Line Use</b>	<b>3</b>
4.1	Syntax . . . . .	3
4.2	Common Options . . . . .	4
4.2.1	Keeping the File Attributes and Timestamps (-p) . . . . .	4
4.2.2	Copying Entire Directories (-r) . . . . .	4
4.2.3	Using Compression For Text Files (-C) . . . . .	4
4.2.4	Using Batchmode (-B   -o 'Batchmode yes') . . . . .	5
<b>5</b>	<b>Automated Copying</b>	<b>5</b>
5.1	Key Based Authentication . . . . .	5
5.1.1	Generating Keys . . . . .	5
5.1.2	Authorizing Keys . . . . .	5
5.1.3	Converting Keys For Use With OpenSSH . . . . .	6
5.2	Step-By-Step Setup for SSH1 . . . . .	6
5.3	Step-By-Step Setup for SSH2 . . . . .	6
5.4	Step-By-Step Setup for OpenSSH . . . . .	8
<b>6</b>	<b>Large Scale Copying</b>	<b>8</b>
6.1	Entire Directory Structures . . . . .	8
6.2	Partial Directory Structures . . . . .	9
<b>7</b>	<b>Common Problems</b>	<b>9</b>
7.1	Setup Problems . . . . .	9
7.2	Execution Problems . . . . .	9
7.2.1	SCP Prompts and Hangs the Script . . . . .	9
7.2.2	SCP From Cron Fails . . . . .	9

## 1 Overview

Secure Copy (scp) allows files to be securely copied between machines. Automated copies are also possible with scp using RSA or DSA key authentication instead of password authentication. In the following text, **client** refers to the machine issuing the **scp** command and **server** refers to the machine that the **scp** command is directed to.

## 2 The Different Secure Shell Protocols and Servers

This document covers both the SSH1 and SSH2 protocols and three types of SSH servers. Each one varies in syntax and configuration.

- SSH1 Protocol

This SSH protocol is an older protocol, but is still used by many systems. There are many known problems with SSH1 that fall outside of this document's scope. SSH2 or OpenSSH are recommended if possible. An SSH1 client/server is installed on the E10000, but its use is discouraged. All SSH1 commands end in '1' on the E10000 (i.e. **scp** is the SSH2 client where as **scp1** is the SSH1 client). The configuration is roughly similar to the OpenSSH server, but differs drastically from the newer SSH2 protocol server

- SSH2 Protocol

This is the new SSH protocol. This protocol uses DSA encryption keys (and can also uses PGP keys, though it is not discussed here). The configuration files have changed drastically since SSH1. An SSH2 client/server is installed on the E10000 and is the preferred protocol.

- OpenSSH

BSD created OpenSSH, which is both SSH1 and SSH2 compliant. It's configuration files resemble SSH1 and are more consistent when using both the SSH1 and SSH2 protocols. However, encryption keys must be converted (a conversion program comes with the server) to be used with SSH1 and SSH2 servers.

## 3 SSH Client for Windows

This section briefly describes how to use the SSH Client for Windows available from [ssh.com](http://ssh.com).

### 3.1 General Usage/Setup

- Install SSH client from ssh.com (SSHSecureShellClient-x.x.x.exe)

- To add a profile

Profiles ▸ Add Profile...

Then, type a name

Profiles ▸ Edit Profiles...

Then, edit that profile

OR

- Use Quick Connect
- A box will come up briefly where you can type a profile name and click add
- If you miss the box, then

– Profile ▸ Add profile...

\* Type a name

### 3.2 Using SFTP to Transfer Files (recommended)

- Execute the 'Secure File Transfer Client'
- Create a new profile entry if you wish, or use quick connect.

For faster transfers you can use the Blowfish or Twofish Encryption Algorithm.

For better security use 3DES (or <Default>).

- Just drag and drop files to copy
- To setup certain types of files to be converted from MSDOS format to UNIX format (ie get rid of ^M line endings), go to

Edit ▸ Settings ▸ Global Settings ▸ File Transfer ▸ Mode

Then select 'auto select' mode and add any filename extensions you want converted into the ASCII extension list.

### 3.3 Using FTP via SSH to Transfer Files

- Execute the 'Secure Shell Client'
- Setup a profile as you would normally, but also do the following:

- Go to the Tunneling tab
- Select the Outgoing tab
- Click Add...

Display Name: FTP

Type: FTP

Listen Port: 21

NOTE: You can use any free port you want, but you cannot use two profiles at the same time with the same port. You also cannot use port 21 if you are running an FTP server on your PC (uses port 21)

Allow Local connections only: checked

Destination Host: localhost

Destination port: 21

- Connect using this new profile and minimize the window.

NOTE: What happens is that your listen port is forwarded to the destination host's destination port. So, when you connect to localhost:port you are really connecting to desthost:destport via a secure tunnel.

- Bring up your favorite FTP client (SSH must still be running) and connect to:

Host: localhost

Port: <Listen port from above> (you do not need this if you used 21)

NOTE: The builtin windows command line FTP always uses port 21. Also you may need to turn on passive transfers in your FTP client (PASV) - see the FTP client's documentation for this.

- Log in as the user/password on the destination host.
- User your FTP client as you normally would.

## 4 General scp Command Line Use

### 4.1 Syntax

When copying file, you can copy from the client to a server or vice versa. In any case, the last file/directory listed is always the destination. Here is the common syntax. Where

`file1` and `file2` can be either file or directory specifications (either with full paths or paths relative to the user's home directory). If the user is not specified, then the user issuing the command is assumed. If the host is not specified, then the client host is assumed.

```
scp [options] [[user@]host1:]file1 [...] [[user@]host2:]file2
```

**Example:** Copy a file named `test_file` in the working directory to alex's home directory on the `batman.db.vt.edu` server. The `'.'` is a special notation for the current working directory, which is the user's home directory. The `'alex@'` would not be needed if `'alex'` was the user ID of the person issuing the command.

```
scp test_file alex@batman.db.vt.edu:.
```

**Example:** Copy a file named `test_file` from alex's home directory on the `batman.db.vt.edu` server to the current working directory on the local machine. Again, the `'.'` is a special syntax for the current working directory and `'alex@'` would not be needed if `'alex'` was the user ID of the person issuing the command.

```
scp alex@batman.db.vt.edu:test_file .
```

## 4.2 Common Options

### 4.2.1 Keeping the File Attributes and Timestamps (-p)

Using the `-p` option will preserve file attributes and timestamps of each file/directory copied. Some attributes, such as owner and group, may not be preserved if you do not have permissions to do so.

### 4.2.2 Copying Entire Directories (-r)

You can use the `-r` option to copy directories recursively. Note, however, that this will follow symbolically linked directories and copy linked files as real files (i.e. if you have a single file linked three times, then the file will be copied three times instead of being re-linked on the destination). If you wish links to be re-established, then see section 6.

**Example:** To copy a directory named `test_dir` on the `batman.db.vt.edu` server to the current working directory on the local machine, you would do this. Note that `test_dir` will be created (unless it already exists) on the local machine.

```
scp -r batman.db.vt.edu:test_dir .
```

### 4.2.3 Using Compression For Text Files (-C)

This option is not available when using the SSH2 servers. However, if you are using SSH1 and/or OpenSSH, you can specify the files to be compressed during the copy by using the `-C` option. The files are compressed on the source machine, then uncompressed on the

destination. This can significantly reduce file transfer times for large text file, but does not help (and can actually make transfer times longer) with already compressed or binary files.

#### 4.2.4 Using Batchmode (-B | -o 'Batchmode yes')

Batchmode may be necessary if you want to guarantee you will not be prompted (such as in a script) or you are using scp without a TTY (a cron job). This option differs between the different servers and protocols. For SSH1 and OpenSSH, use -o 'Batchmode yes'. For SSH2, you must use -B.

## 5 Automated Copying

### 5.1 Key Based Authentication

#### 5.1.1 Generating Keys

Key generation is accomplished through the `ssh-keygen` program.

##### SSH1:

```
man ssh-keygen1
```

##### SSH2:

```
man ssh-keygen
```

##### OpenSSH:

```
man ssh-keygen
```

#### 5.1.2 Authorizing Keys

**SSH1:** You must copy the client's public key to the server. This public key should be appended to the `$HOME/.ssh/authorized_keys` file. This file should contain authorized public keys, one per line. See section 5.2 for an example.

**SSH2:** SSH2 servers are much more complicated to set up. Basically it involves authorizing the client's private key on the client machine, copying the client's public key to the server, then authorizing the client's public key on the server. See section 5.3 for a more detailed explanation and an example.

**OpenSSH:** OpenSSH servers use the same mechanism as the SSH1 for both its SSH1 protocol and SSH2 protocol setup. The SSH2 protocol setup, however, appends a '2' to the configuration files. See the SSH1 example in section 5.2 for the SSH1 protocol and the OpenSSH example in section 5.4 for an example of setting up OpenSSH using the SSH2 protocol. Please note that connecting an OpenSSH client to an OpenSSH server requires no conversion, however, if you are connecting an OpenSSH client to either an SSH1 or SSH2 server, you will need to convert the public keys using the `ssh-keygen` program - see section 5.1.3 for details on this conversion process.

### 5.1.3 Converting Keys For Use With OpenSSH

See the man page for the OpenSSH version of `ssh-keygen` for further details on this conversion.

## 5.2 Step-By-Step Setup for SSH1

Quick, quick version:

1. Create the key on the client:

```
ssh-keygen1
scp1 $HOME/.ssh/identity.pub user@machine.dom.ain:.
```

2. Authorize the key on the server:

```
chmod o+x $HOME
mkdir $HOME/.ssh
chmod 755 $HOME/.ssh
cat $HOME/identity.pub >> $HOME/.ssh/authorized_keys
chmod 644 $HOME/.ssh/authorized_keys
rm $HOME/identity.pub
```

## 5.3 Step-By-Step Setup for SSH2

1. Create Your Key Files On The Client

Generate a key with this command (on the client machine):

```
ssh-keygen
```

Don't type a passphrase, just press enter. Take note of the public key file name (normally: `$HOME/.ssh2/id_dsa_1024_a.pub`).

## 2. Copy Your Public Key On The Client To The Server

First you must login to the server and make sure there is a directory named `$HOME/.ssh2`. If not, then you can create one with:

```
mkdir $HOME/.ssh2
chmod 700 $HOME/.ssh2
```

Then, you must copy the public key file from the client to the server. I like to use a standard format for the filename, but it is not necessary. This format is as follows – replacing 'user' and 'host' with the real username and hostname from the client:

```
user.host.ssh-dss.pub
```

Take note of what this filename will be. You can produce this filename format with the following command on the client (note the quotes are actually back ticks – usually left of the '1' key on PC keyboards):

```
echo $USER.`hostname`.ssh-dss.pub
```

You can copy the public key file on the client with the following command (again, the quotes are actually back ticks – usually left of the '1' key on PC keyboards):

```
scp public_key_filename user@host:~/.ssh2/$USER.`hostname`.ssh-dss.pub
```

## 1. Authorizing The Key On The Client Side

On the client, you must add a line like the following to your `$HOME/.ssh2/identification` file. If the identification file does not exist, then it must be created. You only have to do this once for every key you generate. The public key that you copied in step 2 must correspond to this private key. The line should be as follows replacing 'private\_key\_filename' with the filename of your private key (normally: `id_dsa_1024_a`). The filename should not include the path.

```
IdKey private_key_filename
```

The following command will put this line into the identification file and create it if necessary:

```
echo 'IdKey filename' >> $HOME/.ssh2/identification
```

Now, you must make sure that the permissions on the identification file are correct:

```
chmod 600 $HOME/.ssh2/identification
```

## 2. Authorizing The Key On The Server Side

Login to the server and change directories into the `$HOME/.ssh2` directory:

```
cd $HOME/.ssh2
```

Change permissions on the file you copied with the following command – be sure to change 'filename' to the name in step 2.

```
chmod 600 filename
```

Now, you must authorize the new public key file so it can be used. To do this you must add the following line to a file named `$HOME/.ssh2/authorization`. If the authorization file does not exist, then it must be created. The line should be as follows, again, replacing 'filename' with the name in step 2. The filename should not include the path.

Key *filename*

The following command will put this line into the authorization file and create it if necessary:

```
echo 'Key filename' >> $HOME/.ssh2/authorization
```

Now, you must make sure that the permissions on the authorization file are correct:

```
chmod 600 $HOME/.ssh2/authorization
```

### 3. Test The Setup

On the client – replacing 'user' and 'host' with your username and hostname of the server:

```
echo 'It Worked' > test_file
scp test_file user@host:.
```

You should have seen output resembling this after executing the above scp command:

```
test_file | 10B | 0.0 kB/s | TOC: 00:00:01 | 100%
```

If the test was successful, you should not have been prompted for a password. If you were, then verify you completed the above steps correctly.

If all steps were completed successfully, then the server may have key authentication turned off. Key authentication on the server is required for automation. To determine if this is the cause of failure try using verbose or debugging mode to display information on the authentication process:

```
scp -v test_file user@host:.
```

or

```
scp -D 2 test_file user@host:.
```

## 5.4 Step-By-Step Setup for OpenSSH

## 6 Large Scale Copying

### 6.1 Entire Directory Structures

One of the problems with using `scp` with the `-r` option is that links and other special files are not copied correctly. You can get around this problem by using `ssh` as a pipe between

two `tar` commands. Basically, the idea is to use `tar` to create a stream of bytes to standard output, send that output to the server through `ssh`, then `untar` the stream of bytes on the server.

- To create a stream of bytes on the client:

```
tar cf - path/dir
```

- To create an `ssh` connection to the server, that will catch the output and exit:

```
ssh machine tar xf - -C path
```

- The full command would then be:

```
tar cf - path/dir | ssh machine tar xf - -C path
```

## 6.2 Partial Directory Structures

If you don't want to copy all of the files under a directory, then you can use the `find` command to create a list of files you do want. This list can be supplied instead of the `path/dir` parameter in the above example.

- Here is an example of how you could use the `find` command within the above example:

```
tar cf - 'find . -name '*.c'' | ssh machine tar xf - -C path
```

## 7 Common Problems

### 7.1 Setup Problems

### 7.2 Execution Problems

#### 7.2.1 SCP Prompts and Hangs the Script

- Make sure SSH is not requesting information. SSH will normally ask you if you wish to accept the server's public key the first time you attempt to connect.
- Use batch mode by adding the `-B` option (for SSH2) or `-o 'Batchmode yes'` option.

#### 7.2.2 SCP From Cron Fails

- Use batch mode by adding the `-B` option (for SSH2) or `-o 'Batchmode yes'` option.
- On many systems cron will only have the default execution path (`PATH` environment variable) defined. This default is typically `PATH=/usr/bin:/bin`. Make sure you use the full path to the `scp/ssh` command instead of relying on the `PATH` environment variable. Many times, you will have to explicitly set the `PATH` so that `scp` can find `ssh`.

```
PATH=$PATH:/usr/local/bin; export PATH
```

- Check your email for cron errors - cron will typically email any output back to the userid running the job. You can change the output's email using the MAILTO environment variable.

```
MAILTO="youremail@your.domain"
```