

# SSH notes

SSH is a secure, comfortable way of logging into UNIX machines, remotely copying files, running remote X11 and TCP socket tunnels.

**Get rid of ftp/telnet/rlogin/rsh and use SSH now!**

***Note: This article is old, please refer to the newer [SSH-Part I](#) and [Part II](#).***

## Contents:

### SSH Overview:

#### Advantages

#### Disadvantages

### Implementations

- [SSH for UNIX & Linux](#) : [SSH1](#), [SSH2](#)
- [Free java SSH clients](#)
- Windows
  - [Free Windows SSH Clients](#)
  - [Commercial SSH for Windows](#): [F-Secure](#), [VanDyke](#)
- Others: [Mac Versions](#), [Palm Pilot](#), [VAX](#)

### Tweaking

- [SSH \(Secure Shell\) VPNs](#)
- [Compiling SSH](#)
- [Secure SSH configuration](#)
- [VNC via SSH](#)

### References

---

## SSH (Secure SHell)

Secure Shell (SSH) is authored by Tatu Yionen, Espoo, Finland and is a secure replacement for Telnet, rlogin, rcp, rsh and provides secured TCP tunnels. Optional compression of traffic is

provided and can also be used together with many Authentication schemes such as SecurID, Kerberos, S/KEY to provide a highly secure remote access point to UNIX servers. Efforts are underway to make SSH an official Internet Standard, see [www.ietf.org/html.charters/secsh-charter.html](http://www.ietf.org/html.charters/secsh-charter.html) .

It is very well designed, supports numerous encryption algorithms, is backward compatible with the Berkeley "r" commands and supports automatic encryption of X sessions.

The following is an extract from the SSH FAQ:

### What is ssh?

*Ssh (Secure Shell) is a program to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another. It provides strong authentication and secure communications over insecure channels. It is intended as a replacement for rlogin, rsh, and rcp. Additionally, ssh provides secure X connections and secure forwarding of arbitrary TCP connections.*

### Why should I use it?

- \* The traditional BSD 'r' - commands (rsh, rlogin, rcp) are vulnerable to different kinds of attacks. Somebody who has root access to machines on the network, or physical access to the wire, can gain unauthorised access to systems in a variety of ways. It is also possible for such a person to log all the traffic to and from your system, including passwords (which ssh never sends in the clear).*
- \* The X Window System also has a number of severe vulnerabilities. With ssh, you can create secure remote X sessions which are transparent to the user. As a side effect, using remote X clients with ssh is more convenient for users.*
- \* Users can continue to use old .rhosts and /etc/hosts.equiv files; changing over to ssh is mostly transparent for them. If a remote site does not support ssh, a fallback mechanism to rsh is included.*

### What kinds of attacks does ssh protect against?

- \* IP spoofing, where a remote host sends out packets which pretend to come from another, trusted host. Ssh even protects against a spoofer on the local network, who can pretend he is your router to the outside.*
- \* IP source routing, where a host can pretend that an IP packet comes from another, trusted host.*
- \* DNS spoofing, where an attacker forges name server records.*
- \* Interception of clear-text passwords and other data by intermediate hosts.*
- \* Manipulation of data by people in control of intermediate hosts.*
- \* Attacks based on listening to X authentication data and spoofed connection to the X11 server.*

*The above only holds if you actually use encryption. Ssh does have an option to use encryption of type "none" this is only for debugging purposes, and should not be used.*

### What kind of attacks does ssh **not** protect against?

*\* Ssh will not help you with anything that compromises your host's security in some other way. Once an attacker has gained root access to a machine, he can then subvert ssh, too.*

*\* If somebody malevolent has access to your home directory, then security is non-existent. This is very much the case if your home directory is exported via NFS.*

### How does it work?

*All communications are encrypted using IDEA or one of several other ciphers (three-key triple-DES, DES, RC4-128, TSS, Blowfish). Encryption keys are exchanged using RSA, and data used in the key exchange is destroyed every hour (keys are not saved anywhere). Every host has an RSA key which is used to authenticate the host. Encryption is used to protect against IP-spoofing;*

*public key authentication is used to protect against DNS and routing spoofing.*

*\* RSA keys are also used to authenticate hosts.*

### Advantages:

- The company Data Fellows have commercialised SSH into a range of products under the F-secure name. Secure VPN tunnels are also on offer. See [www.DataFellows.com](http://www.DataFellows.com) or [Europe.DataFellows.com](http://Europe.DataFellows.com) .
- Since this software was developed outside the U.S., it does not fall within U.S. Government export restrictions, unless resold by a U.S. vendor.
- Encryption protocols used: RSA public key, Triple DES, IDEA, Blowfish, ...
- Tunnelling of static TCP ports works well and can be automated.
- Three types of trust exist: *shosts*, *rhosts* compatible and RSA. Use *shosts* rather than *rhosts*. RSA authentication is stronger (using a private/public key system to identify peers), but bypasses the username/password authentication of UNIX.
- Supports strong authentication systems such as SecurID, S/Key and also Kerberos and TIS
- SOCKS5 proxy aware.

### Disadvantages:

- Port ranges & dynamic ports can't be forwarded.
- No "scp" on Windows versions.
- SSH1 daemon:
  - Cannot restrict what ports may or may not be forwarded, per user.
  - When a user is authenticated by password, the client's RSA identity is not verified (against `ssh_known_hosts`). The verification only takes place when `.[sr]hosts` trust is used.

# IMPLEMENTATIONS

## SSH for UNIX & Linux

SSH1 for UNIX is available as a free or commercial product (from DataFellows):

- The author has been running the free versions V1.2.13 - 1.2.27 on the following platforms for since late 1995: Solaris 2.4, 2.5, 2.6, 2.7, SunOS 4.1.3, OSF1.3, IRIX 5.3. Works very well on Solaris, with problems on IRIX and rdist problems on SunOS (use rdist 6.1.2 to fix). **Highly Recommended.**
- POP, SMTP, FTP authentication and other TCP socket sessions can be tunnelled, e.g. for SMTP:  

```
ssh -L 25:smtp.target.domain:25 target &
```
- A (public domain) Firewall proxy for SSH exists for the FWTK (not tested). Where?
- In addition, V1.2.17 (and later) works with SOCKS5 proxies (used since late 1996).
- SecurID authentication is also supported (used since 1996).
- *Rdist* 6.1.2 (public domain version) runs over SSH (on Solaris 2 & SunOS).
- *Fsh* is an addon to keep an SSH tunnel open to allow several commands to be remotely executed without reopening the tunnel each time [www.lysator.liu.se/fsh](http://www.lysator.liu.se/fsh). I've not yet tried this.

## SSH2

SSH2 is the new protocol version, submitted to the IETF for approval. It includes *sftp*, an SSH2 tunnelled ftp. SSH2 uses separate config files to SSH1 (e.g. /etc/ssh2/ssh2\_config), but can call SSH1 if a client requests SSH1 protocols and SSHD1 is available.

- SSH2 from DataFellows is a commercial product for UNIX or Windows. There is a free version for non-commercial use, but licensing is pretty restrictive.
- Efforts are underway to develop **LSH**, a free version of SSH2. [www.net.lut.ac.uk/psst/](http://www.net.lut.ac.uk/psst/)

## Java SSH clients (free/commercial)

- An extract from the [Readme](#):  
*MindTerm is an entirely FREE SSH client program written in 100% pure Java (non-certified). It can be run as a stand-alone program or as an applet in a webpage. It can be run with or without a GUI (stand-alone). It has other notable features setting it apart from some other clients, scp - file copying and a special ftp-tunnel which works with "ordinary"*

*ftpd's "behind" the sshd. Licensed under Gnu's General Public License (GPL).*

The current version is v1.12, see [www.mindbright.se/mindterm/](http://www.mindbright.se/mindterm/) . **Recommended - my SSH client of choice at the moment.** It has been tested as a standalone application (not an applet):

- Works! ... Stable, pretty, flexible terminal emulation, saves properties per server, can generate RSA keys, session can be logged to file, can be used as GUI or command line, X11 and port forwarding works. Brilliant!  
Some nifty extras: clone terminal, copy on select, capture to file.
- **It has scp** (file copy).
- It has **ftp tunnelling** (in PASV mode).  
Example ftp instructions:
  - 1) On MindTerm client: Go to menu Tunnels -> Basic... Enter a local port of your choice.. Select protocol ftp... Give host-name of ftp-server behind sshd... Click Add button
  - 2) On the FTP client (e.g. ws\_ftp): Define a new "site" with address localhost... go to "Site properties"... in "folder" advanced set "Remote Port:" to local port selected above... enable "Passive transfers"
- Problems: no serious ones, but...  
*scp* will only copy files to the connected server and does not understand target syntax like `john@server3:/mydir` or `~john` for home directories.  
There's no online help (but the *readme* is good).  
I've also had occasional blocked tunnels and had to restart.  
The encryption algorithm can be set to *none* (not at all desirable!).
- More: There is also the [mindtunnel](#) server, maybe it can be used for remote NT login? No changes since Dec.98, and it looks basic, but interesting..

### Installation on Windows:

1. Get it from [www.mindbright.se/mindterm](http://www.mindbright.se/mindterm) and install a Java Runtime v1.1 from [Sun](#),
  2. Or, download my [mindterm\\_install.zip](#) (2MB) and extract to C:\Progra~1, then setup a shortcut on your desktop to c:\Progra~1\mindterm\mindterm.bat .
- This will work with English, French, German, Italian on Win95 and NT (hence the use of the short name). Java runtime v118 is included.

**Installation on UNIX:** If you need a GUI SSH client (but not a server - see next section for SSH server) on your UNIX box, look no further.

- An example startup script for Solaris is [mindterm.sh](#). Copy this along with [mindterm.zip](#) to /usr/local/bin
- Install Java (already installed on newer solaris, or get it from [Sun](#)).
- Make sure /usr/local/bin is in your path and then just type *mindterm.sh*.

- Cédric Gourio also produced an SSH java applet for his Diploma, see [www.cl.cam.ac.uk/~fapp2/software/java-ssh/](http://www.cl.cam.ac.uk/~fapp2/software/java-ssh/)

## Windows

I have to admit that my favourite SSH client for Windows is now MindTerm (see [above](#)) after 2 years using DataFellows F-Secure SSH.

## Free Windows SSH Clients

1. TTSSH (TeraTerm SSH)
  - TTSSH is an SSH add on (DLL) to TeraTerm Pro (a free terminal emulation package) .  
TTSSH [www.zip.com.au/~roca/ttssh.html](http://www.zip.com.au/~roca/ttssh.html)  
TeraTerm <http://hp.vector.co.jp/authors/VA002416/teraterm.html>
  - Version 1.4 of TTSSH (Dec'98) includes the following features: Compatible with SSH protocol version 1.5 Ciphers: 3DES, IDEA, Blowfish, DES, RC4 Server authentication using the ssh\_known\_hosts database (including the option of adding a server's key to the database) Authentication using password, RSA, rhosts and rhosts+RSA. Compression support. Connection forwarding, including full support for X connection forwarding.
  - Note that the older V1.2 did not have X11 or port forwarding.
  - **I've tested V1.4 and am impressed, but it doesn't have the ftp tunnelling of MindTerm.** TeraTerm is a nice terminal emulation too, offering a good GUI and features such as session logging and custom key mapping.  
Tip: set TERATERM\_EXTENSIONS=1 in your environment (so that Teraterm enables extensions and presents SSH by default) and edit [teraterm.ini](#) to change a few defaults.  
TBD: where is the client key generated (Keygen), so that the server can authenticate the client on the host level?
2. See Mindterm above.
3. Most of the following use the Cygnus Win32 libraries, you'll need *usertools.exe* from [sourceware.cygwin.com/cygwin/download.html](http://sourceware.cygwin.com/cygwin/download.html)
4. Raju Mathur/Gordon Chaffee wrote a Win32 patch for ssh-1.2.14, the binaries are in <ftp://ftp.cs.hut.fi/pub/ssh/contrib/>. There's also a ssh-1\_2\_22-Win32-Beta1.

Notes:

- You'll need to create a c:\ssh\etc directory to keep your ssh\_config, ssh\_host\_key, and ssh\_host\_key.pub in. You need to generate the keys on your Unix box with ssh-keygen (bug: ssh-keygen doesn't work on NT)
- Next, make sure your HOME environment variable is set. On NT, this can be done via the Control Panel->System.

- Create a <HOME>\.ssh directory (don't forget the dot). Copy (ftp) your identity and identity.pub files in this subdirectory (these 2 files were generated by ssh-keygen on the unix box)
  - Setting up ssh to work without requiring any passwords: You need a destination machine to trust the source machine. So .ssh/Identity.pub (public key) of the source machine needs to be appended to the list of keys in .ssh/authorized\_keys on the destination machine.
5. A version based on 1.2.20 that hasn't been further developed, is at <http://guardian.htu.tuwien.ac.at/therapy/ssh/>
  6. Cedomir Igaly offers a free 16 and 32 bit version without sources. It uses Peter Gutmann's cryptographic library on [Garbo](http://www.doc.ic.ac.uk/~ci2/ssh). See [www.doc.ic.ac.uk/~ci2/ssh](http://www.doc.ic.ac.uk/~ci2/ssh)
  7. Simon Tatham has developed PuTTY, a free Win32 Telnet/SSH Client. It is at beta v0.47 [www.chiark.greenend.org.uk/~sgtatham/putty.html](http://www.chiark.greenend.org.uk/~sgtatham/putty.html) .

## Commercial SSH for Windows

### F-Secure SSH for Windows (16 and 32 bit)

Datafellows produce a commercial implementation of SSH. Recommended. This PC version allows encrypted "Telnet" sessions and secure TCP socket connections (including X11) with UNIX servers running SSH. See also [www.datafellows.com/f-secure](http://www.datafellows.com/f-secure).

#### Advantages:

- Well integrated into the Windows environment. Easy to use. Stable.
- Strong encryption: the DES, 3DES and Blowfish algorithms may be selected. This offers military grade encryption strength (assuming the algorithms have been correctly implemented!).
- Can forward encrypted TCP sockets for created simple VPN tunnels. Works with SMTP, POP3, telnet etc. (where known static ports are used).
- Although only a 16 bit version is currently available (V1.1), it works fine with Win95 & NT4.

#### Problems:

- There is no "secure file copy" feature, as found on UNIX clients. This is a great pity as it would be very useful.
- Make sure you use v1.1 rather than v1.0, which is a bit unstable.

## VanDyke SSH for Win32 (U.S. users only)

[VanDyke](#) offer a 32bit client that is user friendly, but obviously is U.S. export restricted. It a worrying option: it allows users to save password to allow "easier" login. See [www.vandyke.com/products/securecrt/index.html](http://www.vandyke.com/products/securecrt/index.html)

## Others

### Mac Versions

- [NiftyTelnet SSH](#) is a free ssh client for MacOS. It is an enhanced version of [NiftyTelnet](#).
- Datafellows also produce a commercial Mac client: [www.datafellows.com/f-secure](http://www.datafellows.com/f-secure).

### Palm Pilot

- Top Gun ssh 1.2 at [www.isaac.cs.berkeley.edu/pilot/](http://www.isaac.cs.berkeley.edu/pilot/) or [ftp.zedz.net/pub/crypto](ftp://zedz.net/pub/crypto) outside the US.

### VAX

- Yes, there is a version of SSH that runs on VAX, I've seen it, but am still waiting for the details..
- 

# TWEAKING

## SSH (Secure Shell) VPNs

1. SSH1:
  - SSH1 can easily forward single sockets (POP3, SMTP, etc..) out of the box, with either a PC or UNIX client. The problem is tunnelling an entire TCP/IP session, especially with applications using dynamic ports.
  - If PPP is redirected over an SSH socket, SSH can be used as a general purpose VPN. A practical example using Linux as a gateway on both sides is described the O'Reilly book "[Virtual Private networks, 2nd Edition](#)", [chapter7](#) ([local copy](#)) and more information is at <http://sunsite.unc.edu/LDP/HOWTO/mini/VPN.html> ([local copy](#)). You'll need Linux on both sides and it's a bit tricky. (dated 1997).

- Try also <http://sites.inka.de/sites/bigred/sw/ssh-ppp-new.txt> (dated 1996)
2. The new SSH2 is more suited to general VPNs. TBD.

## Compiling & Installing SSH

**Compiling** SSH1 for UNIX, is straight forward. Assuming we want the standard options and wish to install in /usr/bin and /usr/sbin (the default is /usr/local/bin & sbin):

```
gzcat ssh-1.2.26.tar.gz | tar xf -
cd ssh-1.2.27
./configure --prefix=/usr --without-none --without-rsh
make
make install
```

The SSH daemon will have to be added to one of the system startup files, is is not done by "make install". An example for Solaris to be copied to /etc/rc2.d is [S10sshd](#).

Compilation options that might be useful:

```
--without-none never allow clear text (unencrypted) communication in the case
where one of the servers has no key.
--without-rsh never allow rshell rhosts as an option when a server has no key.
```

```
IRIX 5:  ./configure --prefix=/usr/bsd --sbindir=/usr/bsd --bindir=/usr/bsd
SunOS   ./configure --prefix=/usr/local --sbindir=/usr/local/bin
Socks5 proxying support: -with-socks5=/usr/local/lib
Solaris2 with SecurID:
./configure --prefix=/usr --with-securid=../ace
vi Makefile, add -lsocket to the line: LIBS = ../ace/sdclient.a -lnsl -L/usr/local/lib
vi sshd.h, and add the line: #include <ulimit.h>
make install
vi /etc/securid.users (lists all the users with cards, one per line).
```

## Installing on a number of machines

To make life easier, compile on (say) one solaris machine as above, then create a tar file of the binaries (in the C-Shell):

```
tar cvf ssh_bin.tar /usr/bin/{ssh,ssh1,scp,scp1,slogin,slogin1,ssh-keygen1,ssh-
keygen,ssh-agent1,ssh-agent,ssh-add1,ssh-add,ssh-askpass1,ssh-askpass,make-
```

```
ssh-known-hosts1,make-ssh-known-hosts}
tar uvf ssh_bin.tar /etc/{sshd_config,ssh_config}
tar uvf ssh_bin.tar /etc/rc2.d/{S10sshd,K10sshd} /etc/init.d/sshd
tar uvf ssh_bin.tar /usr/sbin/{sshd,sshd1}
tar uvf ssh_bin.tar /usr/man/man1/{ssh-keygen.1,ssh-agent.1,ssh-add.1,ssh.1,
ssh1.1,slogin.1,slogin1.1,scp.1,scp1.1,make-ssh-known-hosts.1} /usr/man/man8/
{sshd.8,sshd1.8}
compress ssh_bin.tar
```

Copy ssh\_bin.tar.Z to the new target system, backup any existing config files, extract in root, "rehash" (if using csh) and then generate a host key:

```
ssh-keygen -b 1024 -f /etc/ssh_host_key -N "";
```

Add the ssh service, by adding the following to /etc/services:

```
ssh 22/tcp      # Secure Shell
```

Start the ssh daemon:

```
sh /etc/rc2.d/S10sshd start
```

## SSH configuration

- Configuration files: The server has a configuration file /etc/sshd\_config, the client reads a configuration file /etc/ssh\_config, which gives site-wide defaults for various options. Options in this file can be overridden by per-user configuration files (in ~user/.ssh directory).
- Server: Configure the ssh daemon so that access is restricted to named hosts with known public keys (/etc/ssh\_known\_hosts) and rhosts authentication is disabled. See commented example [/etc/sshd\\_config](#). In particular, look at options such as:
  - The StrictHostKeyChecking option can be used to prevent logins to machines whose host key is not known or has changed. If this flag is set to "yes", ssh will never automatically add host keys to the /etc/ssh\_known\_host or \$HOME/.ssh/known\_hosts file, and refuses to connect hosts whose host key has changed. This provides maximum protection against trojan horse attacks. For many Administrator situations setting this flag to "ask" to prompt the user about whether to add the key to the known list of hosts is ideal.
  - RhostsRSAAuthentication when set to yes, allows ~/.shosts to define trust relationships.
 

May be set to "yes", "nopwd", or "no". The "nopwd" value disables password-authenticated root logins, unless there is an .shosta llowing access without a password.

Root login with RSA authentication when the "command" option has been specified will be allowed regardless of the value of this setting (which may be useful for taking remote backups even if root login is normally not allowed.

- An empty config file can be placed in the users home directory owned by root and writable only by root. This will force the system wide settings for all users.
- Use .shosts rather than .rhosts, if remote admin is needed. If /.shosts is used, make sure that permissions are 600 or 400.
- Client:
  - Configure the system wide defaults for the SSH client. See commented example [/etc/ssh\\_config](#)
- Use at least v1.2.26 to ensure forward compatibility with SSH2.
- Rdistd for remote file sync. Use rdist only over SSH.
- An install tool for Linux RPMs is available at <ftp://ftp.yellowdoglinux.com/pub/yellowdog/install-ssh/>
- Using SSH with RPC/keyserv/NFS/NIS+ : see [fy.chalmers.se/~appro/ssh\\_beyond.html](http://fy.chalmers.se/~appro/ssh_beyond.html)
- Nokia firewalls (based on Firewall-1 and a hardened FreeBSD) have an SSH option - use it and disable telnet/ftp!
- Client SUID root, server run as root: SSH package installs two programs that need special privileges. Ssh is the client program, and it is by default installed as suid root, because it needs to create a privileged port in order to use .rhosts files for authentication. If it is not installed as suid root, it will still be usable, but .rhosts authentication will not be available. Also, the private host key file is readable by root only. Sshd is the daemon that listens for connections. It should preferably be run as root, because it is by normally listening on a privileged port (22), and it needs to be able to do setuid(), update utmp, chown ptys etc. when a user logs in. If it is not run as root, explicit "-p port" option must be given to specify an alternate port (same port must also be specified for clients), "-h host\_key\_file\_path" must be given to specify an alternate host key file, and it cannot be used to log in as any other user than the user running it (because it cannot call setuid()). Also, if your system uses shadow passwords, password authentication will not work when running as someone else than root. Both the server and the client have been carefully screened for possible security problems, and are believed to be secure. However, there can be no guarantee.

## VNC via SSH

[Virtual Network Computing](#) is a "remote control" program that allows you to see and use the desktop of another machine (NT, Win95, UNIX) over the network. It could also be used for teleworking over insecure networks such as the internet. Or SSH can be used to encrypt the VNC communications and hence increase it's security.

- Install an NT VNC server on the intranet, which is part of the usual NT domain and has accounts for those who need to logon. [It could also be UNIX, I'm just using NT as an example].  
Secure this machine by choosing a strong VNC password, enabling exclusive VNC access, by enabling a screen saver with password after 5 minutes of inactivity and by not putting it in a public area. Regularly monitor the event log.

- Install an Intranet UNIX SSH server.
- Install a SSH internet gateway, which allows SSH connections from the Internet. Successful logins are presented directly to the Intranet SSH server.  
Security: Put this host in the DMZ between two secure firewall filters. Harden. Disable all services except SSH. Allow no protocols except SSH from the Internet. Configure SSH to refuse root logins and disable trust mechanisms and RSA authentication. Configure all user accounts to use a strong authentication mechanism such as SecurID. Configure user shell to automatically login use to the Intranet UNIX server (don't allow local logins to this server). Be paranoid, monitor logs carefully and run integrity checks (such as tripwire) frequently.
- Install and SSH client such as *mindterm* on the VNC client (somewhere on the Internet).
  - SSH Client configuration: Connect to SSH gateway and authenticate with SecurID (or whatever). Login to UNIX Intranet server. Setup tunnel from local port 5902 to *NT\_VNC\_server* port 5900.
  - VNC Client: Start local VNC client, connect to localhost:2, enter the VNC password and voila, the desktop should appear. Now login to NT as usual.
  - Security: I suggest you use you own machine (not one in an Internet Cafe), with an up-to-date Virus checker, File shares disabled, and a personal firewall such as [BlackICE](#) installed (and set protection level to Paraooid, no file sharing). This machine should be physically protected and possibly fitted with encrypted disks (e. g. PGPdisk).

## PC Anywhere via SSH

PCAnywhere is another remote control program like VNC above, except it's limited to PCs. the newer versions use port 5631 and 5632, but forwarding these over SSH and then asking PCAny to connect to localhost doesn't work.

TBD: find out how to get PCAnywhere going over SSH.

---

## References

- [www.OpenSSH.com](http://www.OpenSSH.com) appeared recently.
- The main FTP site: <ftp://ftp.cs.hut.fi/pub/ssh/> or one if it's mirrors such as <ftp://ftp.cert.dfn.de/pub/tools/net/ssh/>
- The SSH2 home page is [www.ssh.fi](http://www.ssh.fi) (SSH2 and SSH communications products)
- *Getting Started with SSH* by Kimmo Suominen [www.tac.nyc.ny.us/~kim/ssh/](http://www.tac.nyc.ny.us/~kim/ssh/)
- SSH FAQ [www.employees.org/~satch/ssh/faq](http://www.employees.org/~satch/ssh/faq)
- [www.datafellows.com/gallery](http://www.datafellows.com/gallery)

- [www.ietf.org/html.charters/secsh-charter.html](http://www.ietf.org/html.charters/secsh-charter.html)
  - OpenBSD's OpenSSH [www.openbsd.org/openssh/](http://www.openbsd.org/openssh/)
  - Search for SSH pages on the net: [www.links2go.com/topic/SSH](http://www.links2go.com/topic/SSH)
  - Andy Polyakov's SSH notes: [fy.chalmers.se/~appro/ssh\\_beyond.html](http://fy.chalmers.se/~appro/ssh_beyond.html)
  - SANS Universal SSH project [www.sans.org/newlook/resources/ssh.htm](http://www.sans.org/newlook/resources/ssh.htm)
  - [Shane's SSH notes](#)
- 

Back to the [IT Security Cookbook](#)