

# 1. About Secure Shell

This section should answer general questions about Secure Shell and what it does and doesn't do. [Click here](#) for the contents of this section.

## 1.1. What is Secure Shell?

To paraphrase the README file:

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 unsecure channels. It is intended as a replacement for telnet, rlogin, rsh, and rcp. For SSH2, there is a replacement for FTP: sftp.

Additionally, Secure Shell provides secure X connections and secure forwarding of arbitrary TCP connections. You can also use Secure Shell as a tool for things like [rsync](#) and secure network backups.

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 unauthorized 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.

There are two versions of Secure Shell available: SSH1 and SSH2. This FAQ does its best to distinguish when the situation calls for the difference between the two.

## 1.2 How widespread is its use?

The most current figures available are over 2 million Secure Shell users in over 60 countries. This is not an accurate amount, but an estimate. It also does not necessarily include the different implementations of Secure Shell for different operating systems.

Note that this includes both SSH1 and SSH2 implementations.

## 1.3 What protocols does Secure Shell use?

*It should be noted that the SSH1 and SSH2 protocols are in fact different and not compatible with each other.*

For the SSH1 protocol, you can find this information in an old IETF draft [available here](#). It is also available with the latest source distribution for SSH1 at <ftp.ssh.com/pub/ssh/ssh-1.2.27.tar.gz>.

For the SSH2 protocol, you can find this information in the SSH2 IETF drafts:

- <http://www.ietf.org/ids.by.wg/secsh.html>

The fifth IETF draft for Secure Shell, Generic Message Exchange Authentication For Secure Shell is no longer available and expired after 6 months.

## 1.4 What encryption algorithms does Secure Shell use?

Secure Shell uses the following ciphers for encryption:

Cipher	SSH1	SSH2
DES	yes	no
3DES	yes	yes
IDEA	yes	no
Blowfish	yes	yes
Twofish	no	yes
Arcfour	no	yes
Cast128-cbc	no	yes

Secure Shell uses the following ciphers for authentication:

Cipher	SSH1	SSH2
RSA	yes	no
DSA	no	yes

Ciphers may be added or deleted later depending on implementations.

## 1.5 How does Secure Shell authenticate?

Secure Shell authenticates using one or more of the following:

- Password (the /etc/passwd or /etc/shadow in UNIX)
- User public key (RSA or DSA, depending on the release)
- Kerberos (for SSH1)
- Hostbased (.rhosts or /etc/hosts.equiv in SSH1 or public key in SSH2)

Since there is quite a big demand for it, there are some patches available for various forms of authentication. It is up to the authors to make those available. If you wish to have a particular type of authentication in Secure Shell, please submit a [feature request](#) to the SSH Secure Shell team. For OpenSSH features, [please contact the OpenSSH team](#).

## 1.6 What does Secure Shell protect against?

Secure Shell protects against (again, from the README):

- 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 cleartext 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

In other words, ssh never trusts the net; somebody hostile who has taken over the network can only force ssh to disconnect, but cannot decrypt or play back the traffic, or hijack the connection.

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

## 1.7 What doesn't Secure Shell protect against?

Secure Shell 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 nonexistent. This is very much the case if your home directory is exported via NFS.

## 1.8 What is the difference between SSH1 and SSH2?

The difference between SSH1 and SSH2 is they are two entirely different protocols. SSH1 and SSH2 encrypt at different parts of the packets, and SSH1 uses server and host keys to authenticate systems where SSH2 only uses host keys. SSH2 is a complete rewrite of the protocol, and it does not use the same networking implementation that SSH1 does. Also, SSH2 is more secure.

Because of the different protocol implementation, they are not compatible.

In a nutshell, SSH2 is a rewrite of the SSH1 protocol, with improvements to security, performance, and portability.

## 1.9 Who maintains Secure Shell?

SSH Communications Security, is the developer of Secure Shell (secsh) protocol and maintains the releases of SSH1 and SSH2. The [IETF](#) maintains the Secure Shell standards, which is vendor-neutral. The standards are currently in draft form; once there are two independent implementations available, then they can be submitted as an RFC.

There are currently several implementors, both freeware and commercial, of Secure Shell.

## 1.10 Can I run Secure Shell legally?

Most likely. It depends on your country's laws for cryptography and which version of Secure Shell that you're using. Check out the information on licensing, cryptography laws, and patents on cryptographic algorithms below.

### 1.10.1 Licensing

The licensing for SSH2 as of the 2.1.0 release has been completely revised. You can use Secure Shell for free if you are a university user (student, professor, staff, etc) or if you are using it for non-commercial use (playing games, checking personal email, etc.). For any commercial use, you need to have the appropriate license for Secure Shell. [Click here](#) for the current licensing information and [click here](#) for an FAQ on the licensing from SSH Communications Security.

The UNIX version of ssh 1.2.27 may be used freely for non-commercial purposes and may not be sold commercially as a separate product, as part of a bigger product or project, or otherwise used for financial gain without a separate license. The definition of "commercial use" is generally interpreted as using ssh for anything that would generate financial gain, such as logging into a customers system to do administration, or providing ssh as a secure login to your partners or vendors.

Other licensing is developer-dependent.

## 1.10.2 Cryptography laws

In some countries, particularly France, Russia, Iraq, and Pakistan, it may be illegal to use any encryption at all without a special permit.

If you are in the United States, you should be aware that, while ssh was written outside the United States using information publicly available everywhere, the US Government may consider it a criminal offence to export this software from the US once it has been imported, including putting it on a ftp site. Contact the [Bureau of Export Administration](#), which is under the Department of Commerce.

There's a really good link that keeps up to date with the Wassenaar Agreement and the cryptography laws throughout the world. Check out Bert-Jaap Koops [Crypto Law Survey](#).

## 1.10.3 Patents on Cryptographic algorithms

The algorithms RSA and IDEA, which are used by ssh, are claimed as patented in different countries, including the US. Linking against the RSAREF library, which is possible, may or may not make it legal to use ssh for non-commercial purposes in the US. You may need to obtain licenses for commercial use of IDEA; ssh can be configured without IDEA and works perfectly fine without it.

For information on software patents in general, see the League for Programming Freedom's homepage at <http://lpf.ai.mit.edu/>.

## 1.11 What operating systems does Secure Shell run on?

From the [Secure Shell home page](#):

For SSH1 and the current release of SSH2 (2.2.0), check out the portability page at <http://www.ssh.com/ssh/portability.html>. For compatability with OpenSSH, check out <http://www.openssh.com/portable.html>.

There are also non-commercial ports of Secure Shell for SSH1 including PalmOS, Windows, Macintosh, OS/2, BeOS, WindowsCE, Java, and OpenVMS. See [section 2 of this FAQ](#) for information on how to get Secure Shell.

## 1.12 . Shouldn't I be using only SSH2?

**Maintainer's note:** *Since this brought up an interesting discussion on the mailing list, it seems to be a good idea to incorporate some of the helpful information that folks brought up. Thanks! Also, if someone has a better way to organize this section, please let me know.*

The SSH1 protocol is not being developed anymore, as SSH2 is being developed as the standard. Even if you are not using SSH2, many folks are establishing a path towards it. With three implementations (and growing) of SSH2 currently in the works, there is growing support (especially with the SSH2 protocol in IETF draft). However, there are arguments for and against running SSH1.

**Note:** *If you have any additional arguments either way, I'll post them. -AC*

- There are structural weaknesses in SSH1 which leave it open to additional attacks
- SSH1 is subject to a man-in-the-middle attack
- SSH1 has more supported platforms
- SSH1 supports .rhosts authentication (it's against the draft for SSH2)
- SSH1 has more diverse authentication support (AFS, Kerberos, etc.)
- Performance for SSH2 is not equal to SSH1

Rick Moen posted this software matrix on the mailing list that shows software from diverse authors will perhaps partially explain protocol 1.5's persistence:

Highest protocol version supported in software that is:

## Servers

	Straight Proprietary	Gratis-Usage (non-commercial)	Unconditional Gratis-usage	Open Source [1]
OpenVMS	-	-	-	1.5
OS/2	-	1.5	none	none
UNIX	2.0	2.0	none	2.0
Win32	-	2.0	none	2.0

## Clients

	Straight Proprietary	Gratis-Usage (non-commercial)	Unconditional Gratis-usage	Open Source [1]
Amiga OS	1.5	1.5	none	none

BeOS	-	1.5	none	none
Java	-	-	none	1.5 [2]
Macintosh	2.0	-	1.5	none
OpenVMS	-	-	-	1.5
OS/2	2.0	1.5	none	none
PalmOS	-	-	1.5	none
UNIX	2.0	2.0	1.5	2.0
Win16	2.0	1.4	none	none
Win32	2.0	2.0	1.5	2.0
WinCE	1.5	none	-	none

[1] As is defined by the Open Source Initiative at <http://www.opensource.org/osd.html>. The three columns leftwards are breakdowns of all non-open-source categories, i.e., different classes of proprietary licences.

[2] Mats Andersson says [MindTerm](#) will soon support secsh 2.0.

Diane Yi of the Beckman Institute had some slides that compare the protocols:  
<http://www.beckman.uiuc.edu/biss/security/workshops/2000-02/tsld001.htm>

If you are installing a daemon, check to make sure your remote clients are connecting to you with the right version of Secure Shell. An SSH1 daemon will only work with SSH1 clients. An SSH2 daemon will work with SSH2 clients. However, an SSH2 daemon built with SSH1 compability will support both SSH1 and SSH2 clients. For more information on building SSH2 with SSH1 compability, see Section 9.5.

---

| [Previous Chapter](#) | | [Next Chapter](#) |

| [Table of contents of this chapter](#) | | [General table of contents](#) | | [Beginning of this section](#) |