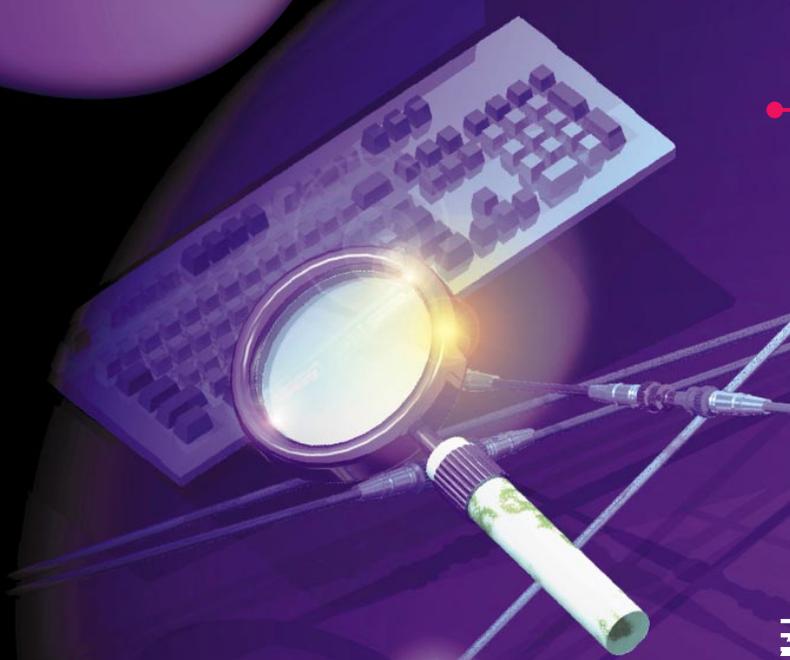


**F-SECURE**



Cryptography  
and  
Data Security  
Toolkit



 **DATA FELLOWS**

World-Wide Web: <http://www.DataFellows.com/>

**F-SECURE**



# F-Secure SSH

for Windows, Macintosh, and UNIX

*Secure Remote Login and System Administration*

User's & Administrator's Guide

All product names referenced herein are trademarks or registered trademarks of their respective companies. Data Fellows Corporation disclaims proprietary interest in the marks and names of others. Although Data Fellows Corporation makes every effort to ensure that this information is accurate, Data Fellows Corporation will not be liable for any errors or omission of facts contained herein. Data Fellows Corporation reserves the right to modify specifications cited in this document without prior notice.

Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Data Fellows Corporation.

Copyright © 1996-1999 Data Fellows Corporation. All rights reserved.

Gummerus Printing, Saarijärvi 1999

#12000006-9K10

# Contents

1. Welcome!.....	1
1.1 About This Guide.....	2
1.2 Overview.....	3
1.3 About the SSH Protocol.....	4
1.4 Public-Key Authentication.....	5
1.5 Tunneling.....	7
2. F-Secure SSH 3.0 Client for Windows 95/98/NT.....	15
2.1 Overview.....	15
2.2 Installation Guide.....	15
2.3 Using F-Secure SSH Client.....	20
2.4 Using the Command Line Applications (ssh2, scp2, sftp2).....	47
2.5 Using scp2.....	51
2.6 Using sftp2.....	53
2.7 Building Tunnels.....	56
2.8 Public-Key Authentication.....	57
3. F-Secure SSH 2 for UNIX.....	65
3.1 Overview.....	65
3.2 Installing F-Secure SSH 2 for UNIX.....	65
3.3 Using F-Secure SSH 2 for UNIX.....	67
3.4 Starting F-Secure sshd2.....	71
3.5 Using scp2.....	72
3.6 Using sftp2.....	74
3.7 Public-key authentication.....	77

3.8	Advanced Installation .....	84
3.9	Configuring F-Secure SSH 2 for UNIX.....	87
3.10	Configuring F-Secure sshd2 for UNIX .....	95
4.	F-Secure SSH 2.0 for Macintosh.....	99
4.1	Overview .....	99
4.2	Installation Guide.....	99
4.3	Using the Menus.....	100
4.4	Connection Manager .....	104
4.5	Group Properties .....	107
4.6	Using Terminals .....	112
4.7	Terminal Properties.....	115
4.8	Tunnels .....	122
4.9	Tunneling Properties.....	124
4.10	F-Secure SSH2 Preferences.....	131
4.11	Public-key authentication .....	139
5.	F-Secure SSH 1 for Windows .....	143
5.1	Compatibility.....	143
5.2	Installing F-Secure SSH.....	143
5.3	F-Secure SSH Wizard.....	147
5.4	Basic Features .....	151
5.5	Forwarding.....	168
5.6	F-SECURE.INI File .....	172
5.7	DEFAULTS.SSH File.....	174
6.	F-Secure SSH 1 for UNIX .....	175
6.1	Overview .....	175
6.2	Portability.....	177
6.3	Installing F-Secure SSH 1 for UNIX.....	179
6.4	Using F-Secure SSH 1 for UNIX.....	180

6.5 Using F-Secure sshd 1.3.7 for UNIX .....	188
6.6 Using scp .....	192
6.7 Public-key authentication.....	194
6.8 Advanced Installation .....	201
6.9 Configuring F-Secure SSH 1 for UNIX .....	204
6.10 Configuring F-Secure sshd for UNIX.....	214
6.11 Troubleshooting .....	227
7. F-Secure SSH 1.0.2 for Macintosh.....	231
7.1 Overview.....	231
7.2 Installation .....	231
Technical Support .....	234
About Data Fellows .....	236



# 1. Welcome!

Welcome to F-Secure SSH, the secure remote login program. F-Secure SSH replaces your existing terminal applications to provide you with secure encrypted and authenticated connections to your UNIX host computers.

F-Secure SSH provides protection for a wide range of security areas. By encrypting interactive terminal and X-window sessions, eliminating plain text passwords, and providing other services, F-Secure SSH closes the most significant authentication and security holes in a distributed computer environment.

This is achieved using strong encryption in the state-of-the-art security protocol SSH. SSH uses both symmetric and asymmetric encryption algorithms to protect your network connections.

## 1.1 About This Guide

This User's and Administrator's Guide covers all the F-Secure SSH products that are based on SSH1 and SSH2. This includes F-Secure SSH products for Windows, Macintosh, and UNIX. This guide is divided into the following chapters.

**Chapter 1. Welcome!** Introduction to SSH, covering features in common for all platforms. Introduction to public-key authentication. Introduction to tunneling and how it is used with ports and hosts.

**Chapter 2. F-Secure SSH 3.0 Client for Windows.** The new version of F-Secure SSH for Windows, based on SSH 2.0.13. New features include the built-in Key Generation Wizard and Key Registration Wizard.

**Chapter 3. F-Secure SSH 2 for UNIX.** Covers SSH 2.0.13 Server and Client for UNIX.

**Chapter 4. F-Secure SSH 2.0 Client for Macintosh.** A new product for Macintosh, based on SSH 2.0.13.

**Chapter 5. F-Secure SSH 1 for Windows.** Covers the original Windows client for SSH1. The information in this chapter is also valid for F-Secure SSH 1 for Macintosh, because the client for Macintosh is identical to the Windows client. Installation of the Macintosh client is covered in Chapter 7.

**Chapter 6. F-Secure SSH 1 for UNIX.** Covers SSH 1.3.7 Server and Client for UNIX.

**Chapter 7. F-Secure SSH 1 for Macintosh.** Installation instructions for the Macintosh client. Information about using F-Secure SSH 1 for Macintosh is covered in Chapter 5.

**Technical Support.** How to contact Data Fellows for assistance.

**About Data Fellows.** Company background and software products.

## 1.2 Overview

### **The F-Secure SSH Product Family**

F-Secure SSH products utilize the SSH protocol as a generic transport-layer encryption mechanism, providing both host authentication and user authentication, along with privacy and integrity protection.

F-Secure SSH UNIX Server can be used together with F-Secure SSH Clients for Windows, Macintosh, and UNIX to make secure login connections to remote offices. F-Secure SSH Server for UNIX includes tools for secure systems administration. Tools are provided for secure file transfer and for tunneling of TCP/IP communications.

The encryption technology has been developed in Europe and does not fall under the U.S. ITAR export regulations. F-Secure products can be used globally in every country where encryption is legal, including the USA. F-Secure products are sold with pre-licensed patented encryption algorithms, which provide the strongest security.

### **F-Secure SSH Server**

F-Secure SSH Server for UNIX provides users with secure login connections, file transfer, and X11 and TCP/IP forwarding over untrusted networks. It uses cryptographic authentication, automatic session encryption, and integrity protection for all transferred data. System administrators can use tools provided in the server package to replace existing rsh, rlogin, rcp, rdist, and telnet protocols. This will enable administrators to perform all remote system administration tasks over secure connections.

F-Secure SSH Server for UNIX supports TCP/IP port-forwarding technology to connect otherwise insecure connections over a secure channel.

Furthermore, secure file transfer (sftp) and secure copy (scp) functionality is supported by F-Secure SSH Server.

## **F-Secure SSH Client**

F-Secure SSH Client provides users with secure login connections over untrusted networks. F-Secure SSH Client acts as a replacement for the telnet protocol, taking advantage of the cryptographic authentication, automatic session encryption, and integrity protection methods defined by the SSH protocol. F-Secure SSH Client fully supports VT100 terminal emulation and ANSI colors.

F-Secure SSH Client also supports secure TCP/IP port-forwarding technology to connect arbitrary and otherwise insecure connections over a secure channel. TCP/IP port forwarding works by creating a proxy server for a source port that a TCP/IP service uses. The proxy server waits on the local machine for a connection from a client program to the source port. F-Secure SSH then forwards the request and the data over the secure channel to the remote system. The F-Secure SSH server on the remote system makes the final connection to the destination host and the destination port.

Most remote services that use TCP/IP can be secured, including custom client-server applications, database systems, and services such as http, telnet, pop, and smtp. F-Secure SSH also provides automatic forwarding for the X11 Windowing System commonly used on UNIX machines.

Furthermore, F-Secure SSH Client enables users to securely transfer files between client and server. All types of files can be transferred, including configuration files, documents, and Web pages.

### **1.3 About the SSH Protocol**

SSH is a packet-based binary protocol that works on top of any transport that will pass a stream of binary data. Normally, TCP/IP is used as the transport, but the implementation also permits using an arbitrary proxy program to pass data to and from the server.

The packet mechanism and related mechanisms for authentication, key exchange, encryption, and integrity implement a transport-layer security mechanism, which is then used to build secure connections.

## 1.4 Public-Key Authentication

The industry-standard IP protocol does not provide any security for data being transmitted across networks. It does not provide authentication, privacy, or data integrity. Higher-level protocols do not provide security to the extent that they rely on lower-level protocols to provide that security. Therefore, security measures must be implemented on the application level.

The SSH protocol is an application-level protocol used by all F-Secure SSH products. SSH guarantees authentication of both ends of the connection, and it guarantees secrecy and integrity of transmitted data.

The server sends its public DSA or RSA host key and a public DSA or RSA "server key" that changes each hour. The client compares the host key it receives against its own database of known host keys. In the SSH1 protocol, RSA is the only option. In the SSH2 protocol, DSA is the default option and RSA is an alternative. In some versions of the SSH2-based software, RSA is not available.

F-Secure SSH Client will normally accept the key of an unknown host and store the key in its database for future reference (making SSH practical to use in most environments). However, F-Secure SSH Client can also be configured to refuse access to any hosts which sends an unknown key.

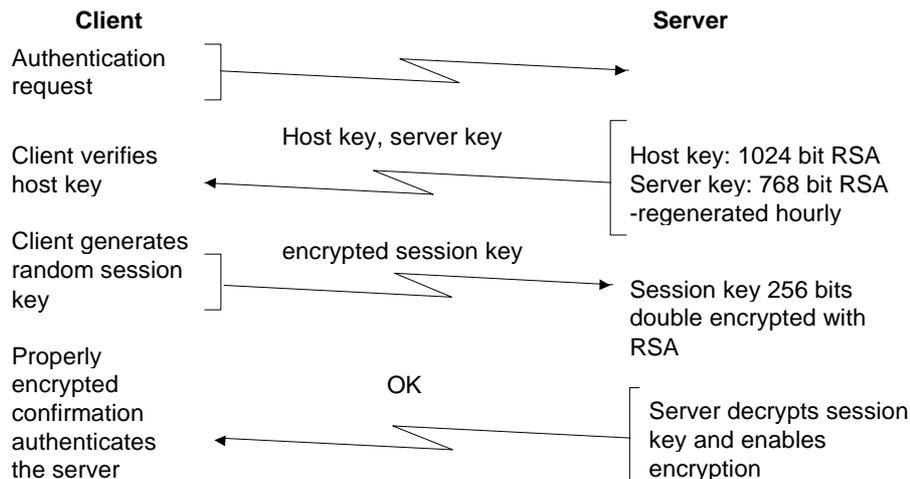


Figure 1: Host Authentication

The client generates a 256-bit random number using a cryptographically strong random number generator, and chooses an encryption algorithm supported by the server (normally Blowfish or three-key triple-DES (3DES)). The client encrypts the random number (the session key) with RSA, using both the host key and the server key. The client then sends the encrypted key to the server.

The host key is used to bind the connection to the desired server machine. The server key is changed every hour. This key is used to make it impossible to decrypt past recorded traffic if the host key has been compromised. In SSH1, the host key is normally a 1024-bit RSA key, and the server key is a 768-bit key. In SSH2, the host key is by default a 1024-bit DSA key. In some versions of SSH2, RSA keys can be used as an alternative. The keys are generated using a cryptographically strong random number generator.

The server decrypts the session key sent by the client. Both parties start using this session key, and the connection is now encrypted. The server sends an encrypted confirmation to the client. When the client receives the confirmation, it knows that the server had the proper private keys needed to decrypt the session key. The server machine has now been authenticated, and transport-level encryption and integrity protection will be in effect.

## **User Authentication**

The user can be authenticated by the server in a number of ways. The user authentication dialogue is driven by the client, which sends requests to the server. The first request always declares what user name to use when logging on. The server responds to each request with a 'success' or 'failure' response (requiring further authentication).

The following authentication methods are supported:

- Traditional password authentication. The password is transmitted over the encrypted channel and cannot be seen by outsiders.
- RSA authentication in SSH1 and in some versions of SSH2. Possession of a particular RSA key serves as authentication. The server keeps a list of accepted public keys.
- DSA authentication by default in SSH2. Possession of a particular DSA key serves as authentication. The server keeps a list of accepted public keys.

## Cryptographic Methods

The SSH protocol provides strong security with cryptography. SSH1 uses only RSA for host and user authentication. SSH2 uses DSA by default; RSA is an option. In some versions, RSA is not available.

The server key that changes every hour is 768 bits by default. It is used to protect intercepted past sessions from being decrypted if the host key is later compromised. The server key is never saved on a disk.

Key exchange is performed by encrypting the 256-bit session key twice using RSA. It is padded with non-zero random bytes before each encryption. Server host authentication happens implicitly with the key exchange. Only the holder of the valid private key can decrypt the session key. Receipt of the encrypted confirmation tells the client that the session key was successfully decrypted.

Client-host authentication and RSA user authentication are accomplished using a challenge-response exchange, where the response is MD5 of the decrypted challenge plus data that binds the result to a specific session (host key and anti-spoofing cookie).

The key exchange transfers 256 bits of keying data to the server. Different encryption methods use varying amounts of the key. Blowfish uses 128 bits. Three-key triple-DES (3DES) uses 168 bits. All random numbers used in SSH are generated with a cryptographically strong random number generator.

## 1.5 Tunneling

When connecting to a remote server using telnet, your user name and password are sent over the Internet in plain text. Anyone can eavesdrop on your connection by using 'sniffer' software, which is freely available from the Internet. A sniffer can pick up your user name, password, and data while they are being transmitted over the Internet.

SSH can be used to effectively block any attempts to steal your password and valuable data. You can safely download your e-mail from your company's internal mail system from anywhere in the world. You can make secure telnet connections, and copy files across an untrusted network without any danger of revealing their contents to anyone. This can be achieved by using SSH tunneling, also known as port-forwarding.

Tunneling works the same in the Windows, Macintosh, and UNIX versions. Only the user interfaces are different.

## Local Tunneling

A local tunnel is a secure, encrypted connection between the SSH client you are using and a remote SSHD server. Data is encrypted while it is traveling through this tunnel. However, data forwarded to a computer outside the tunnel will not be encrypted once it leaves the tunnel.

You set up your SSH client to listen to a specific port on your local host (the host you are making the connection from). When the client receives a request for data on that port, it transfers the request to a port on a remote host that you have specified.

Suppose you want to create a local tunnel from one computer, called *my.computer.com*, to another computer, called *second.host.com*. To encrypt a telnet connection between these two computers, you would do the following:

1. Create an SSH connection from *my.computer.com* to *second.host.com*.
2. Tell the SSH client to listen to a specific port for data requests on *my.computer.com*. The port number has to be greater than 1023 on a UNIX client when not running as root.
3. Tell the SSH client to forward any requests received by *my.computer.com* at the specified port (port 1024, for example) to *second.host.com* at port 23 (the standard telnet port).
4. You can now connect to your own computer at port 1024, using telnet. You can define your own computer as '*127.0.0.1*', '*localhost*', '*my.computer.com*', or by its external IP address. It is recommended that you use either '*localhost*' or '*127.0.0.1*'.
5. As soon as you give the command '*telnet my.computer.com 1024*', the SSH client notices the request at port 1024, and immediately transfers the request through the encrypted tunnel to *second.host.com* at port 23. You are now connected to the telnet port of *second.host.com* through an SSH tunnel. You will be prompted for your user name and password just as if you were connecting directly to the telnetd server, except that all data transferred between your telnet client and the telnetd server is encrypted, making it unreadable by others.

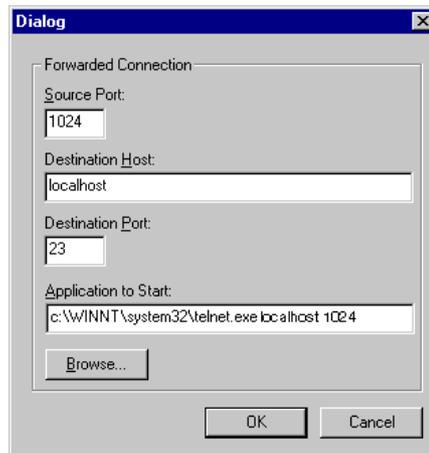
In UNIX, the SSH command is the following:

```
ssh second.host.com -L 1024:localhost:23
```

This command means, "Make an ssh connection to *second.host.com*, and create a local tunnel that listens to local port 1024 and forwards any requests arriving at that port to the telnet port (23) of *second.host.com*."

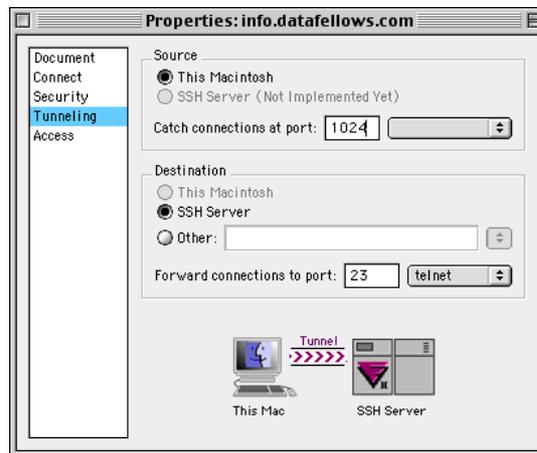
- ☞ **In all tunnels, 'localhost' refers to the host the SSH connection is made to, and NOT the host you are connecting from.**

In F-Secure SSH 3.0 for Windows, you create the tunnel in the Local Tunneling page of the Properties dialog box. The tunnel is opened as soon as you open the SSH connection. The information needed to create the above tunnel is entered as follows:

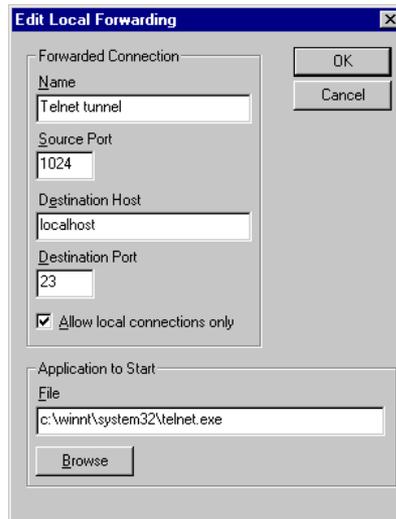


You can start a telnet application by entering it into the Application to Start field, or you can run it from Windows.

With F-Secure SSH 2.0 for Macintosh, you create the tunnel described above by entering the information as shown below.



With F-Secure SSH 1 for Windows and for Macintosh, you create the tunnel described above by entering the information as shown below.



The above procedure can be applied to tunneling many other kinds of data. If you create several tunnels for one connection, you just need to specify a source port that you have not used yet (>1023 for UNIX non-root) and the destination port of the service you want to forward. Example commands:

1. Use the following command to forward your e-mail from a pop3 server on *second.host.com*:

```
ssh second.host.com -L 1110:localhost:110 (143 for imap)
```

Then configure your e-mail client to read mail from 'localhost', port 1110.

2. Use the following command to send e-mail through an smtp server on *second.host.com* using an e-mail editor from your own workstation:

```
ssh second.host.com -L 2525:localhost:25
```

Then configure your e-mail editor to send mail through 'localhost' at port 2525.

3. Use the following command to forward Web content from the http server on *second.host.com*: `ssh second.host.com -L 8080:localhost:80`

You can then connect your Web browser to 'localhost:8080' and receive the content.

## Creating a Tunnel to a Third Host

Another way for using local tunnels is forwarding data through the SSH server to a third host. It is important to know that the data is no longer encrypted when it is transferred between the SSH server and the third host, only between your client and the server. However, when using this functionality to transfer data between your computer and a trusted intranet network, the server-to-third-host security is not an issue.

The main application for this is to enable people to access data, such as e-mail, Web content or news from an intranet server that does not allow direct connections to these servers. In many cases, your only way of accessing your company intranet from the internet is through a single SSH server. Once able to access that server, you can then continue from there to connect to the other servers running inside the intranet.

In SSH tunnels, you can create a connection directly to these servers otherwise inaccessible to you. This is done by specifying the destination host as something other than 'localhost'.

Suppose you want to read your e-mail from a pop3 server in your company intranet, from home. You make an SSH connection to *my.company.com*, tell the SSH client to listen for data requests at some port in your home computer, for example 1110, and to forward any received data requests to the pop3 server at the pop3 port (110) of your company through the SSH server.

In UNIX, the SSH command is the following:

```
ssh my.company.com -L 1110:pop3.company.com:110
```

This command means, "Make an ssh connection to *my.company.com*, and create a local tunnel that listens to local port 1110 and forwards any requests arriving at that port to the pop3 port (110) of *pop3.company.com*."

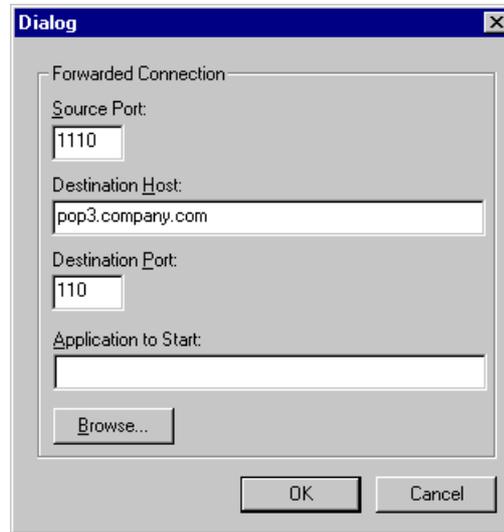
If you are using an imap mail server (port 143), the command is:

```
ssh my.company.com -L 1110:pop3.company.com:143
```

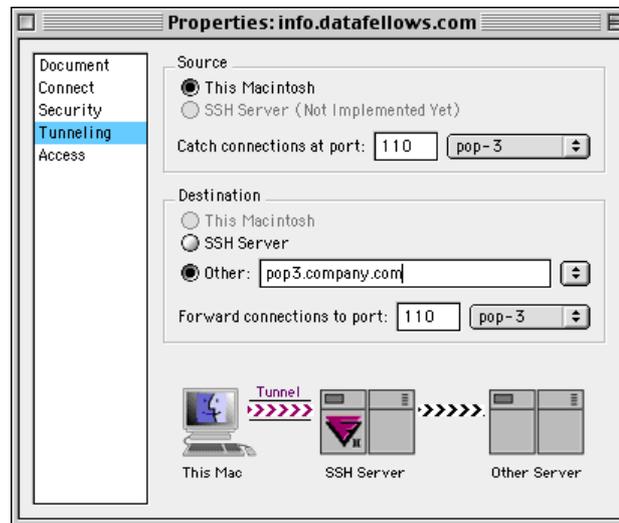
This done, you configure your e-mail client to search for e-mail at 'localhost' at port '1110'. You will be asked for your username and password when the e-mail client first attempts to retrieve mail. You must give the ones for the e-mail server, not the ones for the SSH server.

In the other clients you would enter the required information as explained in their respective Chapters, in the following way:

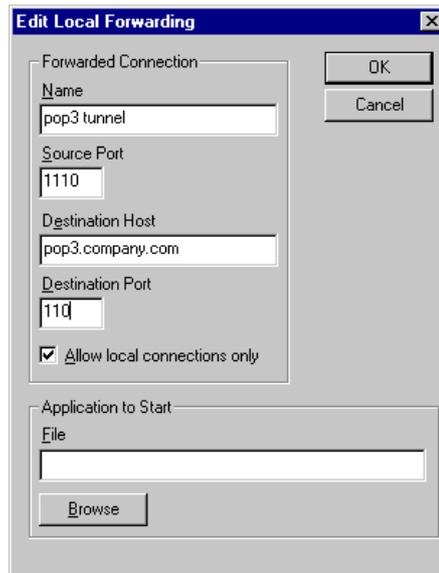
1. F-Secure SSH 3.0 for Windows:



2. F-Secure SSH 2.0 for Macintosh, using 110 as the local port as well:



## 3. F-Secure SSH 1 for Windows and Macintosh:



This can all be applied to tunneling many other kinds of data. If you create several tunnels for one connection, you just need to specify a source port that you have not used yet (>1023 for UNIX non-root), the DNS name or IP address of the server and the destination port of the service you want to forward. Here are two example commands:

1. To forward your telnet connection from a telnet server on *telnet.company.com* through the company's external SSH server:

```
ssh my.company.com -L 2323:telnet.company.com:23
```

Then connect to 'localhost' at port 2323 with a telnet client.

2. To send e-mail through an smtp server on *smtp.company.com* through the company's external SSH server, using an e-mail editor from your own workstation:

```
ssh my.company.com -L 2525:smtp.company.com:25
```

Then configure your e-mail editor to send mail through 'localhost' at port 2525.

3. To forward Web content from the http server on *intraweb.company.com* through the company's external SSH server:

```
ssh my.company.com -L 8080:intraweb.company.com:80
```

You can then connect your Web browser to 'localhost:8080' to receive the content.

## Remote Tunneling

In remote tunneling, you do quite the opposite from local tunneling. You make an SSH connection to a server, then tell the SSH client to listen to data requests received by the host the SSH server is on, at the port you specify, and to transfer that data request to your computer, or a third host. Again, the data is not encrypted outside the tunnel, that is if you forward the data to a third host. Here are two example commands:

1. To create an SSH tunnel to *my.company.com* and tell the SSH client to listen for data requests at port 2323 on that server, and forward all received data requests to port 23 of your workstation:

```
ssh my.company.com -R 2323:localhost:23
```

2. To create an SSH tunnel to *my.company.com* and tell the SSH client to listen for data requests at port 2323 on that server, and forward all received data requests to port 23 of your *third.host.com*:

```
ssh my.company.com -R 2323:third.host.com:23
```

## 2. F-Secure SSH 3.0 Client for Windows 95/98/NT

### 2.1 Overview

F-Secure SSH Client for Windows provides users with secure login connections over untrusted networks. F-Secure SSH Client acts as a replacement for the telnet protocol.

F-Secure SSH Client enables users to securely transfer files between client and server. All types of files can be transferred, including configuration files, documents, and Web pages.

### 2.2 Installation Guide

To install the F-Secure SSH Client, follow these steps:

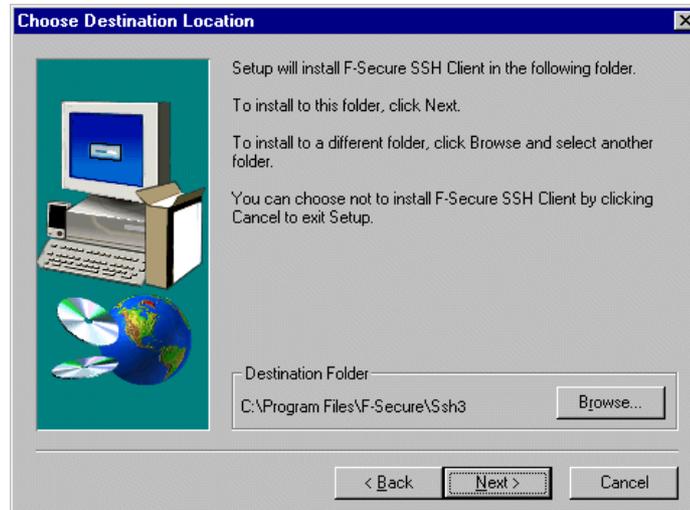
1. Insert the F-Secure CD-ROM into the CD-ROM drive. This should start the autorun feature on the CD-ROM. If nothing happens, go to the main directory of the CD-ROM and double-click on *install.exe*. This will start the F-Secure installer.
2. Select your preferred language from the 'Language' list, and *F-Secure SSH Client 3.0 for Windows 95/98/NT* from 'the Product or document' list.



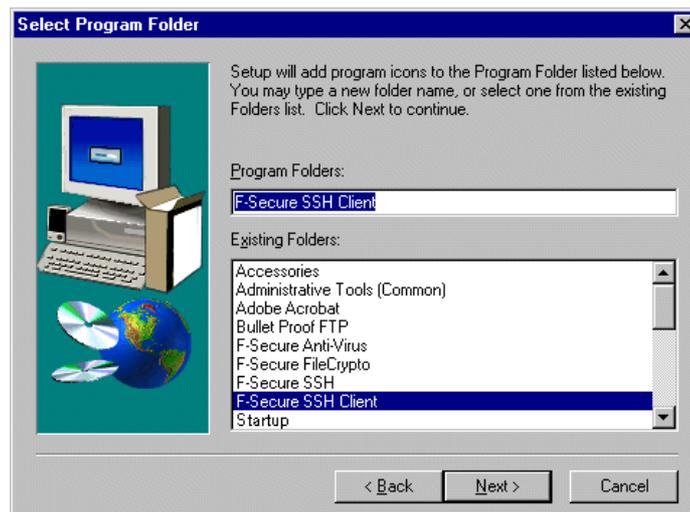
3. In the Welcome screen, click **Next**.



4. In the Choose Destination Location dialog box, click **Browse**, if you want to change the destination where the F-Secure SSH Client will be installed. When you are satisfied with the destination, click **Next**.



5. In the Select Program Folder dialog box, specify the location in the Windows Start menu where you want to install the shortcuts for the F-Secure SSH Client. Click **Next**.

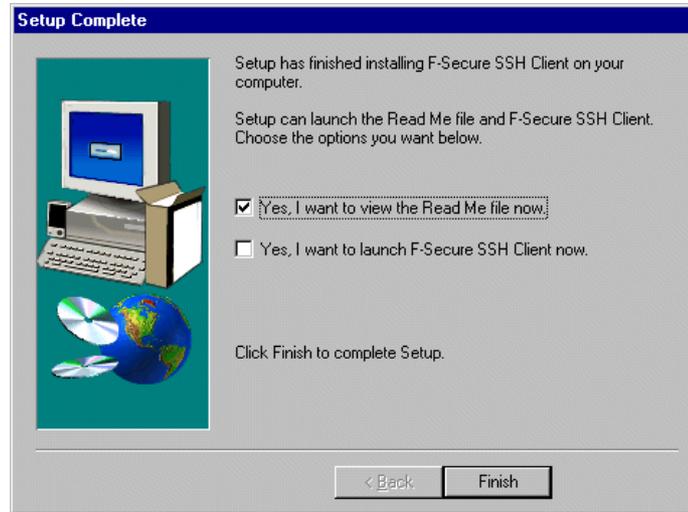


6. In the Select Folder Type dialog box, choose whether you want to install the F-Secure SSH Client for private use only or for the use of anybody with access to the computer. Click **Next** to start the installation.



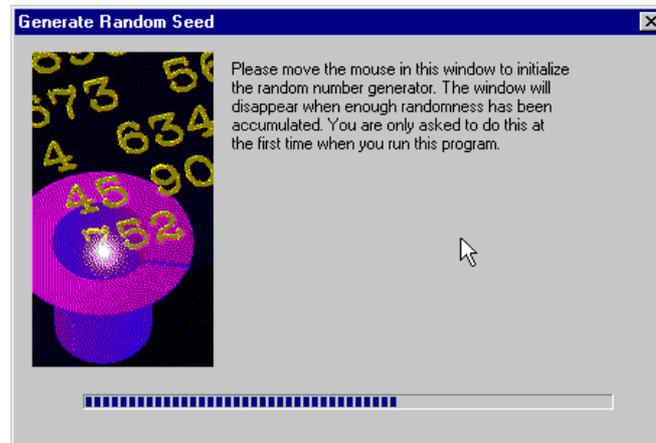
7. You will be asked whether you want to add F-Secure SSH Client directory into *path*. If you answer Yes, you will be prompted to reboot the machine when the installation is finished. After rebooting, you can use the command-line applications from any directory on your machine. If you answer No, you can run the F-Secure SSH command-line application only from the directory in which you installed it.

- When the setup is complete, you can choose to launch F-Secure SSH Client or view the Readme file, which contains the latest information about the software. Then click **Finish**.



## 2.3 Using F-Secure SSH Client

The first time you start the F-Secure SSH Client, a random seed generator will be launched. The random seed needs to be generated before any host or user keys can be generated, as the random seed functions as the starting point for creating them. The random seed is also used to create randomness in all encryption processes and TCP packets.

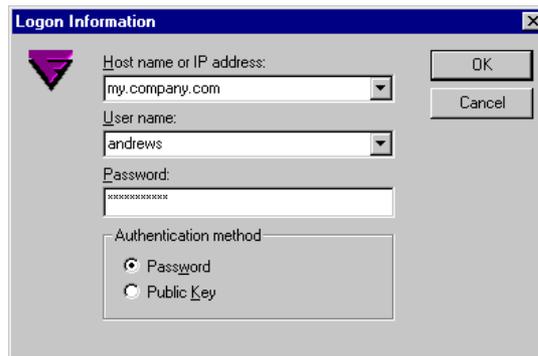


To create the random seed, move your mouse pointer in the window until the progress indicator reaches the end. When the random seed has been generated, F-Secure SSH Client will be launched.

## Connecting to a Remote Host

Start the F-Secure SSH Client from the Windows Start menu under Programs > F-Secure SSH Client. To connect to a remote computer using password authentication, follow these steps:

1. When you start F-Secure SSH Client, the Connection dialog box will open automatically. Type the name or IP address of the host you want to connect to, and your user name and password on the remote host. Set the Authentication Method to Password. For information on using public keys, read Chapter 2.8, Public-Key Authentication, on page 57.



2. After giving the required information, click **OK**.

The first time you connect to a particular host, the following message will be displayed.



Click **OK** to continue.

You will now be connected to the remote host, as long as the information you have provided is correct and the host you are connecting to supports ssh.

## Changing Hosts

If you want to change hosts without quitting the program, do the following:

1. Disconnect from the host you are connected to, either by choosing Disconnect from the File menu, or by logging out of the server using the associated UNIX command (`exit`, `logout`, or `quit`).
2. Press ENTER to open the Connection dialog box, or choose Connect from the File menu.
3. Type the name or IP address of the host you want to connect to, and type your user name and password on the remote host. The authentication method should be set to Password, unless you have already created and transferred a public key to the host. Information about doing that can be read in Chapter 2.8, "Public-Key Authentication," on page 57.
4. Click **OK**.

## Creating Session Files

Session files allow you to save different configurations for your connections. In addition, they allow you to open multiple connections with different host names, user names and passwords without constantly having to return to the Properties dialog box. You can create a session file from an existing open connection or from a previously saved session file. For information on settings you can change in the Properties dialog box, see "Setting Properties," on page 30.

To create a session file, follow these steps:

1. From the Edit menu, choose Properties.
2. Make the changes you want. Note that some changes cannot be made while a connection is open. Therefore, you should not be connected to a host when making changes to your session files.
3. Click the **OK** button on a Property page to activate your changes.

From the File menu, choose Save As, and type a new name for the session file. If you save the session file as *Defaults.ssh*, the properties will be loaded by default every time you start a new F-Secure SSH Client session.

- ☞ **The saved session files are associated with the F-Secure SSH Client program. You can create icons on the Windows desktop for the session files (.ssh files). Double-clicking the icons or dragging them to an open F-Secure SSH Client window will launch the saved connection.**

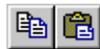
## Toolbar, Status Bar, and Menus

### Toolbar

The toolbar in the F-Secure SSH Client has seven buttons. If buttons are not displayed, you can display them using the View menu. From the left, the first three buttons are the **New Window**, **Open**, and **Save** buttons. These are for session files. 

1. Use the **New Window** button to open a new SSH window that is based on the *Defaults.ssh* file and to open a dialog box for entering required connection information.
2. Use the **Open** button to open an existing session file. For more information about session files, see "Setting Properties" on page 30.
3. Use the **Save** button to save current settings to an open session file without confirmation prompts. The file name *Defaults.ssh* will be used if you have not saved the connection properties with a different name.

The **Copy** and **Paste** buttons are the next two buttons on the toolbar. These are used for the Windows clipboard.



1. Use the **Copy** button to copy selected text to the clipboard. If no text is selected, the **Copy** button is unavailable, indicated by its gray color.
2. Use the **Paste** button to paste the contents of the clipboard to the terminal at the command prompt as user input. If the clipboard is empty, the **Paste** button is unavailable, indicated by its gray color.

The last two buttons are the **Print** button and the **Properties** button.



1. The **Print** button opens a dialog box that allows you to quickly print the terminal output. If you have not selected any text, you can print the whole buffer up to 500 lines; this is the default action. Or you can choose to print the contents of the visible screen. You can select the print quality and number of copies to print. And you can direct the printing to a file if required.
2. The **Properties** button opens the Connection Properties dialog box. The Properties options are described in "Setting Properties" on page 30.

## Status Bar

At the bottom of your terminal window, the status bar displays information about the connection and current settings. To display the status bar, choose Status Bar from the View menu.



The status of your current connection is displayed on the left side of the status bar. It will display "Connected to...", "Press Enter to connect", or "Connecting to...".

The status bar also shows whether compression is enabled or not, which cipher is being used, the cursor position (line and column), and the terminal window size (in characters).

## Menus

File menu option	Function
New Window	Opens a new terminal (based on the <i>Defaults.ssh</i> file) and opens a connection dialog box that displays connection details.
Open Session File...	Open a previously configured session file.
Connect/Disconnect	The Connect option will be displayed if there is no open connection. Choose Connect to open the connection dialog box for starting a connection using your current terminal settings. The Disconnect option will be displayed if you have an open connection. Choose Disconnect to disconnect from the remote host.
Save	Saves changes to your current session file.
Save As...	Saves changes to a different session file under a new name.
Print...	Print the terminal output. You can print the buffer (up to 500 lines), the visible screen, or the selected text. You can direct the output to a file, and select the print quality and number of copies to print.
Print Setup...	Select the printer to use, paper size, paper source and orientation, and printer properties.
1 .. 4	A list of recently used files (up to four files). You can clear this list in the Security pane of the Properties dialog box.
Exit	Closes the terminal and exits F-Secure SSH 3.0 Client. Before exiting, you should disconnect from the server and close all programs that use tunnels created by SSH.

Edit menu option	Function
Copy	Copy selected text to the clipboard.
Paste	Paste the text from the Windows clipboard to the terminal command line.
Paste Selection	Paste the selected text to the command line. This will leave the contents of the clipboard unchanged.
Select All	Select all text in the terminal window and scrollbar buffer.
Clear Scrollback	Clear the terminal buffer, but not the visible screen.
Clear Screen	Clear the visible screen, but leave the buffer and add the contents of the screen to the end of the buffer.
Reset Terminal	Clear the visible screen and the buffer.
Properties	Edit session file properties.
View menu option	Function
Toolbar	If selected, shows the toolbar below the menu bar.
Status Bar	If selected, shows the status bar at the bottom of the terminal window.
Tools menu option	Function
Key Generation Wizard	Start the Key Generation Wizard to easily create a key pair to use for public-key authentication. For information on public-key authentication, see Section 2.8, "Public-Key Authentication," on page 57. For information on the Key Generation Wizard, see "Key Generation Wizard" on page 57.
Key Registration Wizard	Start the Key Registration Wizard to easily utilize your key pairs for use with public-key authentication. For more information on the Key Registration Wizard, see "Key Registration Wizard" on page 60.

Help menu option	Function
Help Topics	Start online Help.
Web Club	Connects to the F-Secure SSH Web Club through the Internet. An Internet connection must exist for this option to work.
About F-Secure SSH	Displays version and copyright information.

## Working with Text in the Terminal Window

### Selecting All Lines in the Scrollback Buffer

To select all the text in the scrollback buffer, do one of the following:

- Drag the mouse across all of the lines.
- Right-click, and click Select All on the shortcut menu.

### Selecting a Word on the Screen

To select a single word on the screen, do one of the following:

- Drag the mouse across the word.
- Right-click on the word, and click Select Word on the shortcut menu.

### Selecting a Line of Text on the Screen

To select a line of text on the screen, do one of the following:

- Drag the mouse across the line of text.
- Right-click on the line, and click Select Line on the shortcut menu,.

### Selecting a URL From the Screen

To select a URL from the screen, do one of the following:

- Drag the mouse across the URL.
- Right-click on the URL, and click Select URL on the shortcut menu,.

## Copying Text to the Clipboard

To copy text to the Clipboard, do one of the following:

- Select the text you want to copy.
- Choose Copy from the Edit menu. Or right-click and click Copy on the shortcut menu. Or press CTRL+INSERT.

After copying text to the clipboard, you can paste it into SSH or another Windows application.

 **You can right-click on the terminal window to open a shortcut menu that has most of the Edit commands.**

## Pasting Text to the Command Line from the Clipboard

To paste text from the clipboard, do one of the following:

- Choose Paste from the Edit menu.
- Press SHIFT+INS.
- Right-click in the terminal window and click Paste from Clipboard on the shortcut menu.

## Pasting the Selected Text Without Affecting the Clipboard

To directly paste text you have selected in the terminal window, do one of the following:

- Choose Paste Selection from the Edit menu.
- Right-click in the terminal window and click Paste Selection on the shortcut menu.
- If you have a three-button mouse, press the middle button.
- You can emulate a three-button mouse with a two-button mouse. Choose Emulate 3-Button Mouse from the Appearance page of the Properties dialog box. Select the text in the terminal window you want to paste, hold down the right mouse button, and click the left mouse button.

## **Clearing the Screen**

To clear the screen while preserving the contents of the scrollback buffer, do one of the following:

- Choose Clear Screen from the Edit menu.
- Right-click the mouse and click Clear Screen on the shortcut menu.

## **Clearing the Scrollback Buffer**

To clear the scrollback buffer while preserving the contents of the visible terminal, do one of the following:

- Choose Clear Scrollback from the Edit menu.
- Right-click the mouse and click Clear Scrollback on the shortcut menu.

## **Resetting the Terminal**

To clear the contents of both the scrollback buffer and the visible terminal window, choose Reset Terminal from the Edit menu.

## Setting Properties

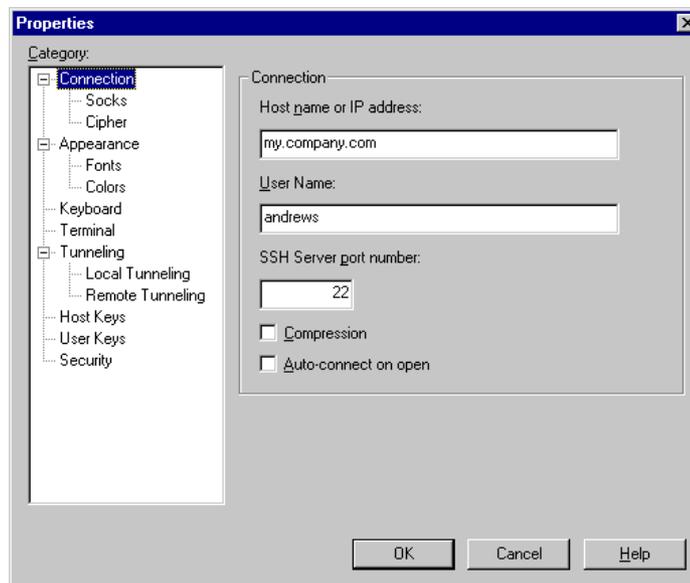
By choosing Properties from the Edit menu, or by clicking the Properties icon on the toolbar, you can modify the settings for the terminal display, connection information, the keyboard, secure TCP/IP connections, and other options.

Property Sheet	Description
Connection	Specify information about the host, user, authentication type, Socks support, port, and cipher used when connecting to a remote system.
Appearance	Show or hide the toolbar and status bar.
Keyboard	Customize keyboard behavior and specify the keyboard map file used to translate keystrokes and received terminal data into characters displayed in the terminal window.
Terminal	Choose the type of terminal you want to emulate.
Tunneling	Specify the local and remote TCP/IP connections to be secured by forwarding them through the F-Secure SSH connection. Identify the names and connection parameters for the forwarded connections.
Host Keys	Import, export, and delete host keys.
User Keys	Import, export, and delete your key pairs, copy them to the clipboard, or change the passphrase.
Security	Remove logged data about your previous connections. Prevent unknown host keys from being accepted.

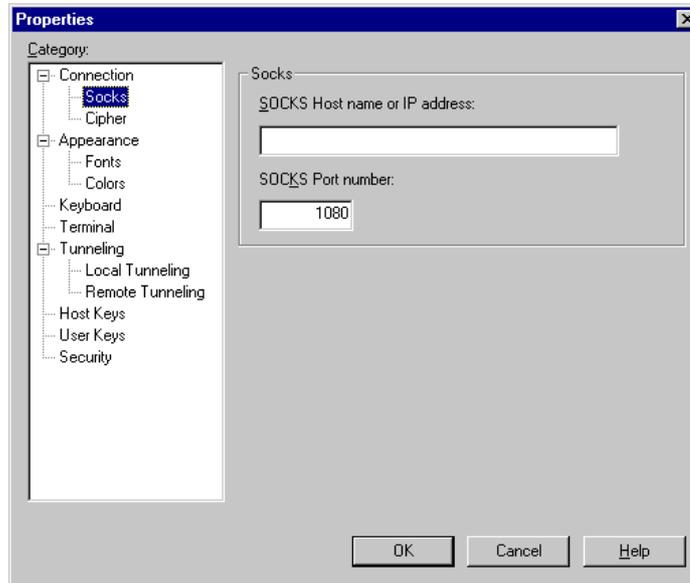
## Connection Properties

Specifies options for the terminal connection and data transfer.

Connection	Function
Host Name or IP address	Specifies the host name or the IP address of the remote machine to log in to.
User Name	Specifies the user name for logging in to the remote machine.
SSH Server Port Number	The connection port number on the remote host. By default, the port number is 22.
Compression	When selected, all data is compressed, including data for forwarded X11 and TCP/IP connections. The compression algorithm is the same as used by the gzip program.
Auto-connect on open	Connects automatically to the host specified in a session file, when the session file is opened.



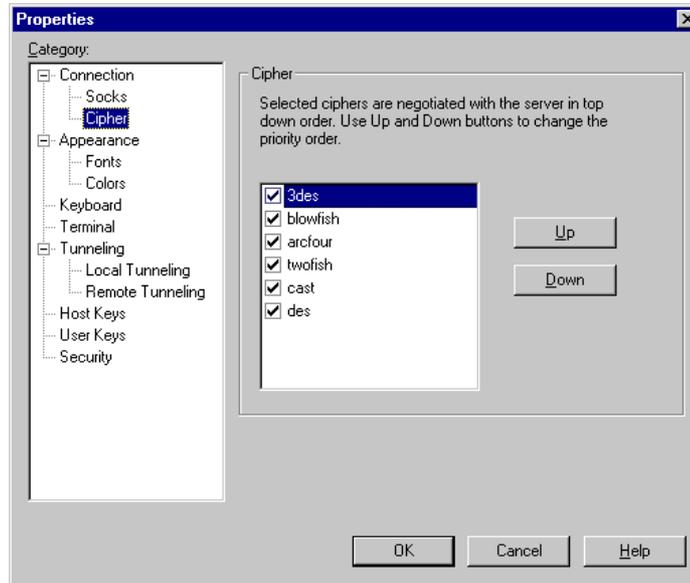
Socks	Function
SOCKS Host name or IP address	Specifies the SOCKS host name or IP address.
SOCKS Port number	Specifies the SOCKS port number being used.



## Cipher

---

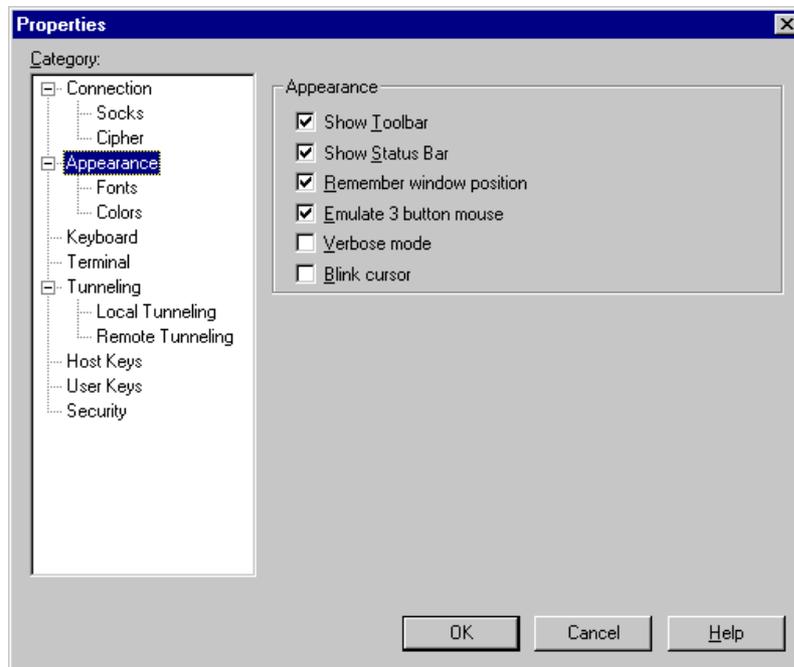
Allows you to select which ciphers to use for encrypting the session. You can move the ciphers you prefer to the top of the list. The ciphers are compared to the list of ciphers on the SSH server in the order you have listed them. The cipher that will be used is the first cipher on your list that matches a cipher available on the remote server.



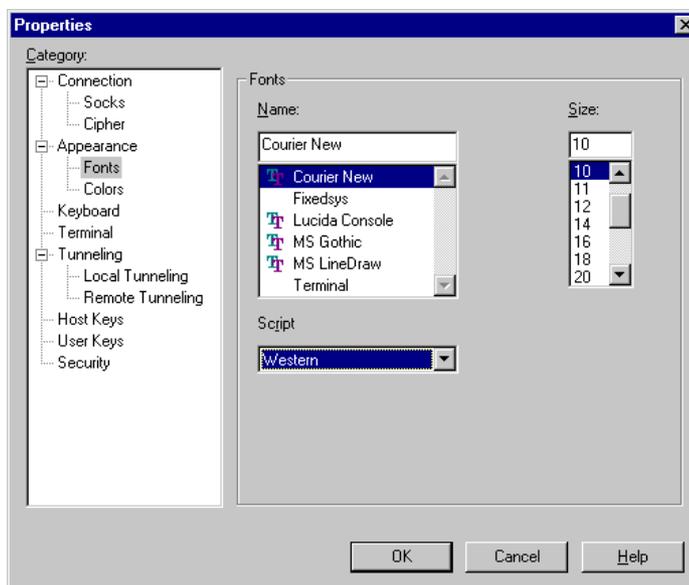
## Appearance Properties

Displays or hides the toolbar and status bar, and specifies terminal window fonts and colors.

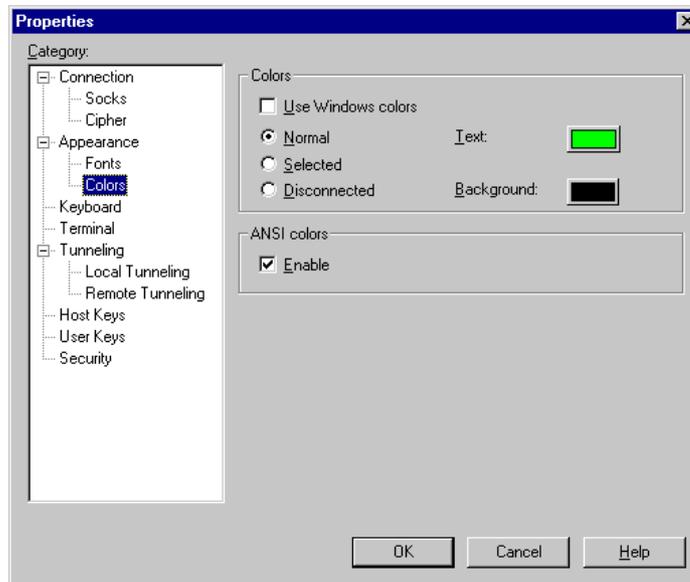
Appearance Options	Function
Show Toolbar	Shows or hides the toolbar.
Show Status Bar	Shows or hides the status bar.
Remember window position	Always opens the window in the same location on the screen.
Emulate 3 button mouse	With a two-button mouse, you can emulate the third button by pressing and holding the right button and clicking the left button. The third button can be used to paste a selection in the terminal window (Paste Selection).
Verbose mode	Displays ssh-specific debug information.
Blink cursor	Causes the block cursor to blink.



Font Options	Function
Name	Specifies the font to be used. The Fonts page lists currently available fonts for the terminal window.
Size	Specifies the font size.
Script	Specifies the character set to use for encoding Web content.

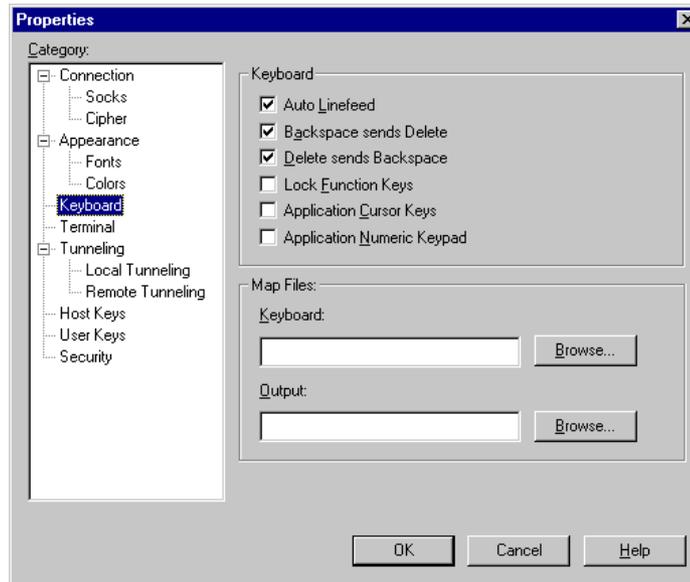


Color Options	Function
Use Windows colors	If selected, your default Windows colors will be used for the terminal connection. If not selected, you can choose colors for the background and text in four different situations: <ol style="list-style-type: none"> <li>1. Normal — Colors displayed in the normal terminal environment.</li> <li>2. Selected — Colors for selected text.</li> <li>3. Disconnected — Colors displayed when you are disconnected but still have the terminal window open.</li> </ol>
Text	Colors for text on the screen.
Background	Background colors.
ANSI colors	Enables the use of ANSI colors in the terminal window.



## Keyboard Properties

Specifies options used to translate key presses and received terminal data into characters displayed in the terminal window.



Keyboard Option	Function
Auto Linefeed	Enables or disables automatic insertion of line-feeds after a carriage return.
Backspace Sends Delete	Causes the BACKSPACE key to send the DELETE key code.
Delete Sends Backspace	Causes the DELETE key to send the BACKSPACE key code.
Lock Function Keys	Locks all VT100 function keys (F1 to F20) in order to prevent them from being programmed by VT100 escape codes.
Application Cursor Keys	Toggles between the application cursor key mode and the VT100 cursor key mode.
Application Numeric Keypad	Toggles between the application numeric keypad mode and the VT100 numeric keypad mode.

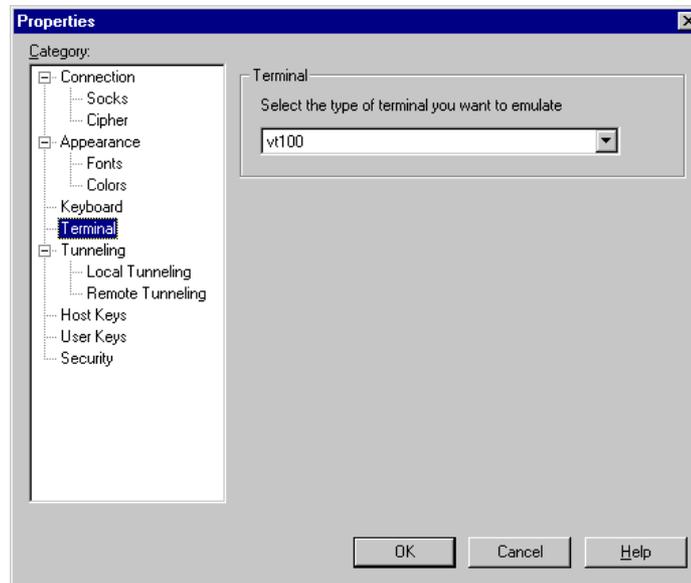
### Map Files Option

---

Keyboard	Allows you to define your keyboard map file. To define a keymap file from a location other than the default location, click the <b>Browse</b> button.
Output	Allows you to define your keyboard map file for output. To define a keymap output file from a location other than the default location, click the <b>Browse</b> button.

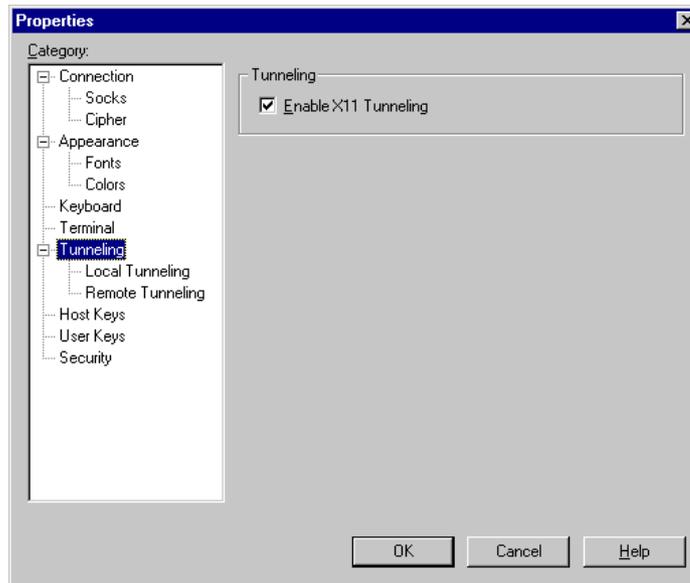
## Terminal Properties

Enables you to choose the terminal you want to emulate. The available options are xterm, xterm-color, and vt100. You can select one from the list box. The vt100 emulation lets you use standard vt100 functions (such as remote printing) in an encrypted format.

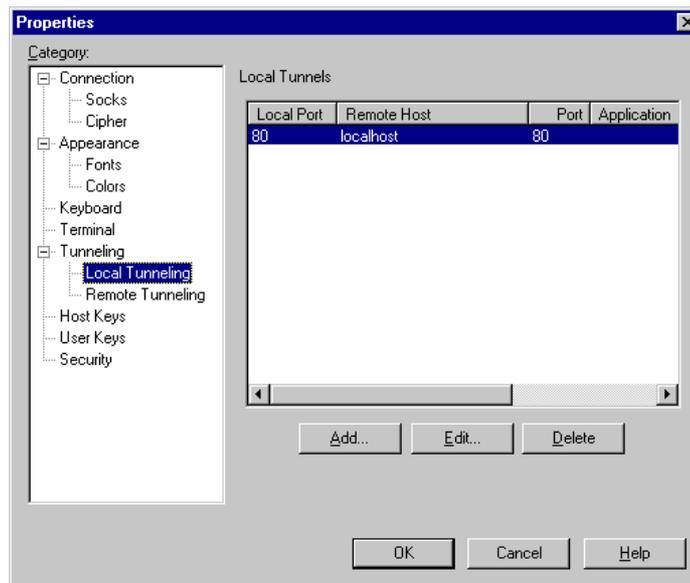


## Tunneling Properties

Specifies options for the local and remote TCP/IP connections to be secured by F-Secure SSH.

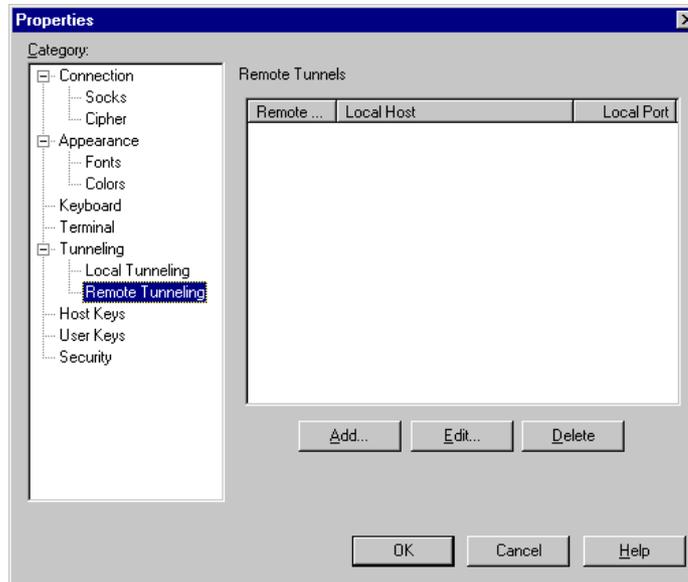


Tunneling Option	Function
Enable X11 Tunneling	Enables X11 forwarding.
Local Tunneling	Displays the local TCP/IP ports which have been forwarded. You can add, edit, and delete local tunnels from this pane. Tunnels can be added even while the connection is open. They are opened when you click the <b>OK</b> button, but they cannot be edited or deleted while the connection is open.



## Remote Tunneling

Displays the remote TCP/IP ports which have been forwarded. You can add, edit, and delete remote tunnels from this pane.



## Button

## Function

New

Define a new TCP/IP connection to be secured. Opens a dialog box for choosing the source port, destination host, and port parameters. New tunnels can be added even while connected.

Edit

Edit a previously defined tunnel. Opens a dialog box for choosing the source port, destination host, and port parameters. This button is disabled when connected to the host.

Delete

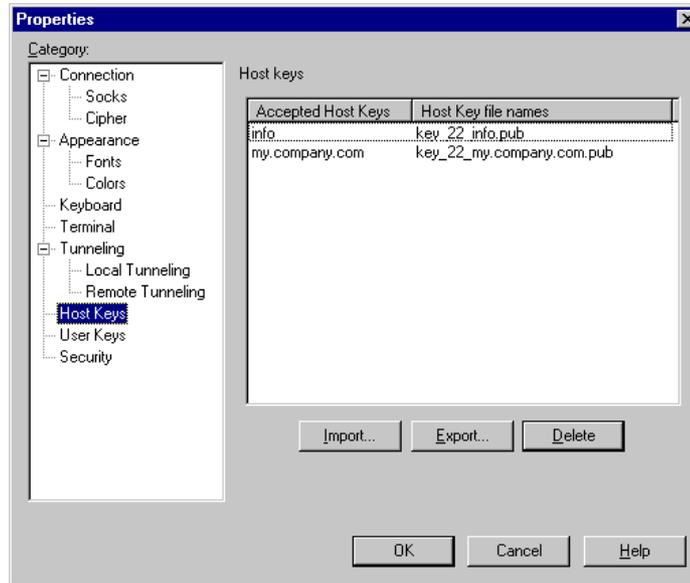
Delete the currently selected tunnel. This will only take effect after you click the **OK** button. To keep the tunnel, click **Cancel**. The button is disabled when connected to the host.

Click the **New** or **Edit** button to display the Forward a TCP/IP Connection dialog box.

In this dialog box, you can specify the local and remote ports for the connection and the remote system to forward the connection to.

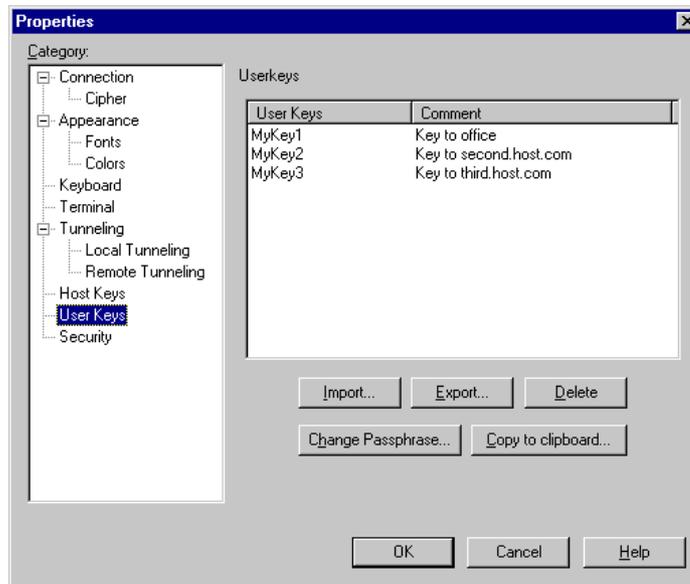
For more information on tunneling, see Section 1.5, "Tunneling," on page 7.

## Host Keys



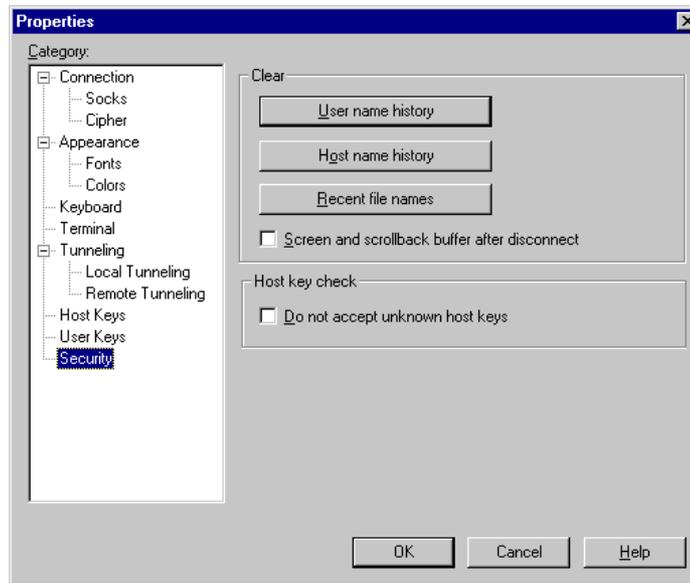
Button	Function
Import	Import a public host key file to the system registry.
Export	Export the selected key from the system registry to a host key file.
Delete	Delete the selected host key from the system registry.

## User Keys



Button	Function
Import	Import a public user key file to the system registry.
Export	Export the selected user key pair from the system registry to files.
Delete	Delete the selected user key pair from the system registry.
Change Passphrase	Change the passphrase for the highlighted key.
Copy to Clipboard	Copy the public key to the clipboard.

## Security



Option	Function
Clear....User name history	Delete recent user name entries from the Logon Information dialog box.
Clear.... Host name history	Delete recent host name entries from the Logon Information dialog box.
Clear.....Recent file names	Delete recent files from the File menu.
Clear.....Screen and Scrollback buffer after disconnect	Clear all data from the visible terminal and scrollback buffer when you disconnect from the server.
Do not accept unknown host keys	Allows connecting only to hosts whose host keys you have received.

## Disconnecting

You can save the connection settings in a session file for later use. From the File menu, choose Save As to save the connection settings. If you save the session file as Defaults.ssh, the settings will affect all future connections based on your default settings.

To close the connection, do the following:

1. If you have a terminal session active in the terminal window, enter the command to log out from the remote system. This command is commonly called logout, exit, or quit.
2. Close any applications that are using the forwarded TCP/IP connections.
3. Choose Disconnect from the File menu.

If you did not close all forwarded TCP/IP connections, these forwardings will be listed on the terminal screen. F-Secure SSH Client will wait for the forwardings to close after closing the terminal connection.

## 2.4 Using the Command Line Applications (ssh2, scp2, sftp2)

F-Secure SSH comes with three applications that can be used from the Windows command line:

- ssh2 – An application for logging on to a remote machine and executing commands.
- scp2 – An application for copying files between hosts on a network.
- sftp2 – An application for starting a secure file transfer session between two hosts.

### Using ssh2

ssh2 is used for starting a secure terminal connection to a remote host, executing commands on a remote host, and creating tunnels for the secure transfer of TCP packets and X11 connections. This utility can be used as a secure substitute for rlogin, rsh, and telnet. It provides secure, encrypted communications between two hosts over an unsecure network.

ssh2 connects and logs in to the specified hostname. A user must prove his identity to the remote machine by using one of the following authentication methods.

- password authentication
- public-key authentication

Use the following format for starting an ssh2 session from the UNIX command prompt:

```
ssh2 [options] host [command]
```

All options start with "-".

If your user name on the remote host is the same as on the host you are connecting from, you only need to give ssh2 the host name. ssh2 logs you in with your default user name and just asks you for your password on the remote host.

Example:

```
my.company.com% ssh2 second.host.com
andrew's password:
Last login: Wed Jun 16 08:48:59 1999
second.host.com%

my.company.com% ssh joseph@third.host.com
Executing /usr/local/bin/ssh1 for ssh1 compatibility.
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)?
yes
Host 'third.host.com' added to the list of known
hosts.
Creating random seed file ~/.ssh/random_seed. This may
take a while.
joseph@third.host.com's password:
Last login: Wed Jun 16 10:22:07 1999 from
my.company.com
Linux 2.0.35.
third.host.com:~$
```

If you have a different user name on the remote host, you need to include the user name in the command line when making a connection. You can either do this with the `-l` option or by using the format `username@remote.host`.

Example:

```
my.company.com% ssh -l josephthird.host.com
joseph's password:
Last login: Wed Jun 16 10:22:07 1999 from
my.company.com
Linux 2.0.35.
third.host.com:~$
```

ssh2 obtains configuration data from the following sources, in this order:

1. System's global configuration file (typically `/etc/ssh2/ssh2_config`).
2. User's configuration file (`$HOME/.ssh2/ssh2_config`).
3. Command line options.

For each parameter, the last obtained value will be in effect.

## Command Line Options

-l login_name	Specifies the user for login to the remote machine.
-n	Redirects input from /dev/null. (Do not read stdin.) This option can also be specified in the configuration file.
+x	Enables X11 connection forwarding. (Default)
-x	Disables X11 connection forwarding.
-F file	Specifies an alternative configuration file to use. Specified options are in addition to those read in the \$HOME/.ssh2/ssh2_config file.
-t	Allocates a tty, even if a command is given. This option can also be specified in the configuration file.
-v	Enables verbose mode. Verbose debugging messages are displayed. Same as '-d 2'. This option can also be specified in the configuration file.
-d debug_level	In this version, only debug level 2 can be defined. Displays verbose debugging messages.
-V	Displays version string.
-q	'Quiet' mode. No warning messages will be displayed. This option can also be specified in the configuration file.
-e char	Set escape character. Use 'none' to disable. This option can also be specified in the configuration file. (Default; ~)
-c cipher	Select encryption algorithm. Multiple -c options are allowed, and a single -c flag can have only one cipher. This option can also be specified in the configuration file.
-p port	Specifies the port to connect to on the remote host. This option can be specified in the configuration file.

- P Do not use privileged source port. Prevents the use of rhosts or rsarhosts authentications, but it can be used to bypass some firewalls that do not allow privileged source ports to pass. This option can also be specified in the configuration file. (Not yet implemented.)
- S Do not request a session channel. This can be used with portforwarding requests if a session channel (and tty) is not needed or provided by the server.
- L port:host:hostport Specifies that the given port on the local (client) host is to be forwarded to the given host and port on the remote side. This works by allocating a socket to listen to on the local side. Whenever a connection is made to this port, the connection is forwarded over the secure channel and a connection is made to host:hostport from the remote machine. Port-forwardings can also be specified in the configuration file. Only root can forward privileged ports.
- R port:host:hostport Specifies that the given port on the remote (server) host is to be forwarded to the given host and port on the local side. This works by allocating a socket to listen to on the remote side. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to host:hostport from the local machine. Privileged ports can be forwarded only when logging in as root on the remote machine.
- +C Enables compression.
- C Disables compression. (Default)
- o 'option' Can be used to give options in the format used in the configuration files. This is useful for specifying options for which there is no separate command-line flag. The option has the same format as a line in the configuration file. Comment lines are not currently accepted by this option.
- h Displays brief help about command-line options.

## 2.5 Using scp2

scp2 (Secure Copy) is used to copy files over the network securely. It uses ssh2 for data transfer. It uses the same authentication and provides the same security as ssh2. Unlike rcp, scp2 will ask for passwords or passphrases if they are needed for authentication.

Any filename may contain a host, user, and port specification to indicate that the file is to be copied to or from that host. Copies between two remote hosts are permitted.

Use the following format for copying files with scp2:

```
scp2 [options] [[username@]host[#port]:]file [[username@]host[#port]:]file_or_dir
```

For example, to copy the file *program.exe* from your local hard drive to *second.host.com*, where your user name is *andrews*, you would enter:

```
scp2 program.exe andrews@second.host.com:program.exe
```

All options start with "-". The first filename is the source file and the second is the destination file. The host name needs to be given only if the host is a remote host. The user name is required only if it is different from the local user name. The port number is required only if it is not port 22, the standard ssh2 port.

## Command Line Options

- D debug\_level\_spec      Prints extensive debug information to stderr. debug\_level\_spec is a number, from 0 to 99, where 99 specifies that all debug information should be displayed.
- d                         With this option, scp2 will make sure that the destination file is a directory. If not, scp2 will exit with an error message.
- p                         Tells scp2 to preserve file attributes and timestamps.
- n                         Makes scp2 show what operations would have been done, without actually copying any files.
- v                         Makes scp2 verbose. This is equal to specifying the '-D2' option.
- V                         Displays version information.
- h                         Displays brief help.
- c cipher                 Selects the encryption algorithm that ssh2 will use. Multiple -c options are allowed, and a single -c flag can have only one cipher.
- S ssh2-path             Specifies the path to ssh2.
- P ssh2-port             Specifies the remote port to ssh2. Ports can also be defined individually for each file using the format *hostname#port:filename*.
- q                         Quiet mode. Does not show the progress indicator while transferring files.

## 2.6 Using sftp2

sftp (Secure File Transfer) is an ftp-like client that can be used in file transfer over the network. sftp uses Ssh2 in data connections, so the file transport is secure.

Use the following format to start an sftp2 session from the UNIX command prompt:

```
sftp [options] [[user]host[#port]]
```

All options start with “-”

### Command Line Options

-d debug_level_spec	Debug mode. Makes sftp send verbose debug output to stderr. The debugging level is either a number (0 to 99), or a comma-separated list of assignments. "ModulePattern=debug_level".
-c cipher	Select encryption algorithm. Multiple -c options are allowed; a single -c flag can have only one cipher.
-v	Enables verbose mode. Displays verbose debugging messages. Same as '-d 2'. This option can also be specified in the configuration file.
-S ssh2_path	Specifies the path to ssh2 used in connecting.
-p remote_port	Specifies the remote port to connect to.

After initiating the sftp session, you can execute the following commands:

? [keyword]	'?' prints the list of the available commands on the screen. '?' followed by a command on the list gives a short description of the command.
bell	Toggles on and off the option of sounding a bell after each file is transferred.
bye	Disconnects from the server and quits the sftp2 client.
cd	Changes the directory on the remote server.

close	Closes the connection, but does not quit sftp2.
delete remote_filename	Deletes the specified remote file. This command does not accept wildcards
dir [remote_dir] [remote_file]	Lists the contents of the remote directory in long format.
disconnect	Same as the 'close' command.
get remote_file [local_file]	Downloads the specified file. If the destination file name is not given, the original name is used.
hash	Toggles the download/upload progress indicator on and off.
help	Same as '?'
lcd	Changes the local directory.
ldelete local_filename	Deletes the specified local file. Wildcards are not accepted.
ldir	Lists the contents of the current local directory in long format.
lls [local_dir] [local_file]	Lists the contents of the current local directory in short format.
lpwd	Prints the current working directory on the local machine.
lrm [local_file]	Deletes the specified local file. This command does not accept wildcards.
ls	Lists the contents of the current remote directory in short format.
mkdir	Creates a new directory on the remote host.
open host [port]	Opens a connection to the specified host, using the specified port number.
page	Toggles paging on and off. When paging is on, the output will be paused after each full screen of text.
prompt	Toggles interactive prompting. Interactive prompting occurs during multiple file transfers to allow the user to selectively retrieve or store files.

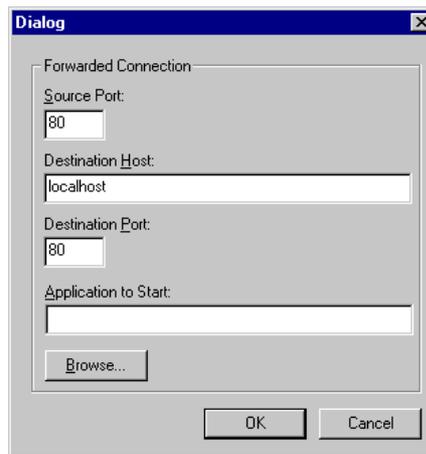
<code>put local_file [remote_file]</code>	Uploads a local file to the remote host.
<code>pwd</code>	Prints the current working directory on the remote machine.
<code>quit</code>	Equivalent to the 'bye' command.
<code>recv remote_file [local_file]</code>	Equivalent to the 'get' command.
<code>rm remote_file</code>	Equivalent to the 'delete' command.
<code>sort</code>	Toggles the sorting of directories on and off.
<code>status</code>	Displays the current status of sftp.
<code>timeout</code>	Sets the timeout for file transfers.
<code>user {user name}</code>	Identifies you to the remote FTP server.
<code>verbose</code>	Toggles verbose mode on and off.

## 2.7 Building Tunnels

Tunneling is one of the three applications for F-Secure SSH Client. In the Windows client, it is very easy to manage TCP/IP tunnels: you open the Properties dialog box from the toolbar or from the Edit menu, activate the Tunneling pane (either Local Tunneling or Remote Tunneling), and add, edit, or delete tunnels. Tunnels can be saved with your session files. Every time you open a session file and connect to a host, the tunnels are automatically opened.

To add a new tunnel, click **Add** on the Local Tunneling pane or the Remote Tunneling pane.

Tunnels can be added while connected. They will be opened as soon as you click **OK**. Tunnels cannot be edited or deleted while connected.



In the above example, a tunnel is being created for all regular http connections. In the Application to Start box, you specify the application to launch after the tunnel has been created. In the above example, the application would be your Web browser.

You edit a tunnel by selecting it with the mouse and clicking **Edit**. The dialog box that you used to create a tunnel will be displayed.

To delete a tunnel, select it and click **Delete**.

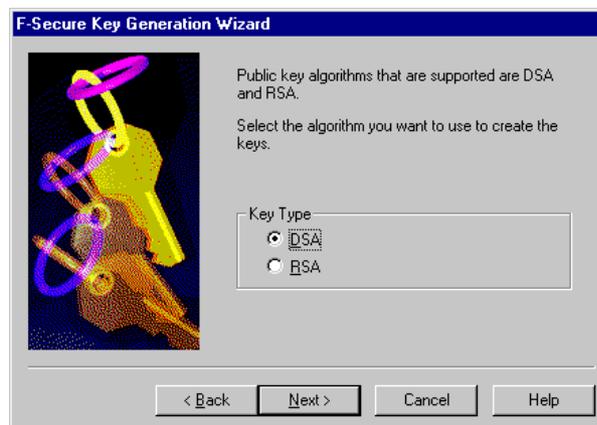
For information on using local and remote tunnels, see Section 1.5, "Tunneling," on page 7.

## 2.8 Public-Key Authentication

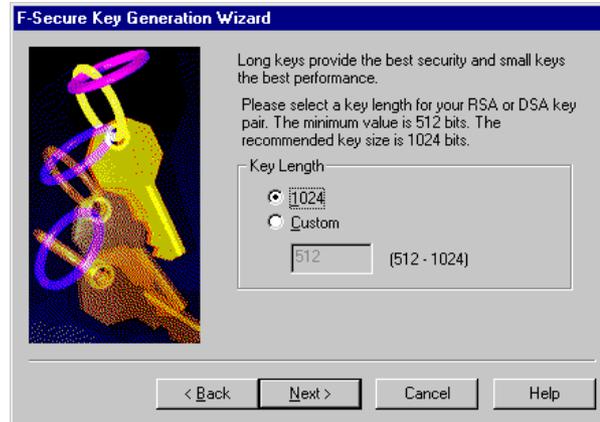
Using public-key authentication is an alternative to using passwords. In F-Secure SSH Client, it is simple to create and implement public keys. You can use the Key Generation Wizard to create a new key pair, and transfer the new public key to the server using the Key Registration Wizard.

### Key Generation Wizard

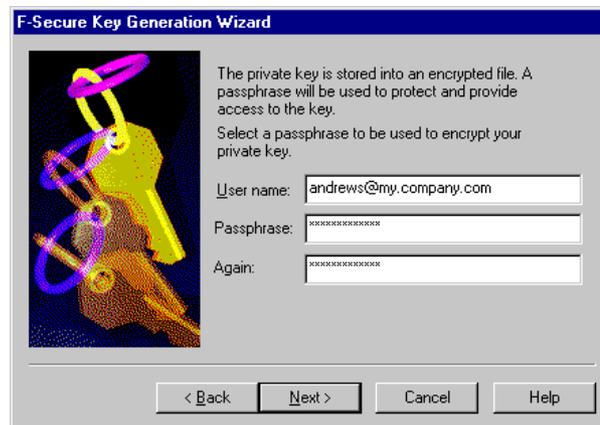
You can start the Key Generation Wizard from the Tools menu or from the Windows Start menu under Programs > F-Secure SSH Client.



Choose whether to create a DSA or an RSA key. DSA is the default value used by F-Secure SSH. In some versions of the software, the RSA option is disabled. In such cases, you can only use DSA keys. Click **Next**.



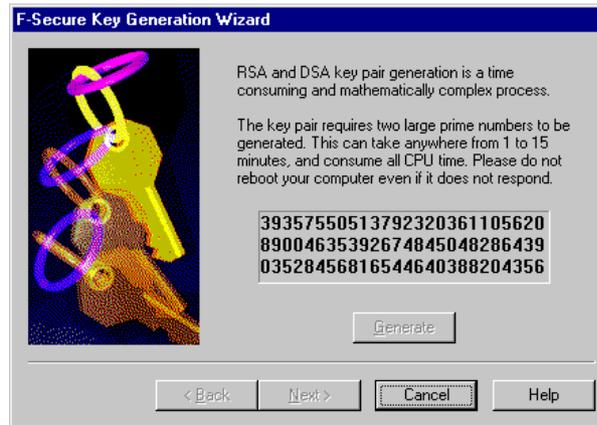
Select the desired key length for the key pair. The recommended key length is 1024 bits, which is also the maximum key length for a DSA key. The maximum key length for an RSA key is 4096 bits. Increasing the key-length increases security, but long keys slow down the RSA authentication process. The minimum key-length is 512 bits. After making your selection, click **Next**.



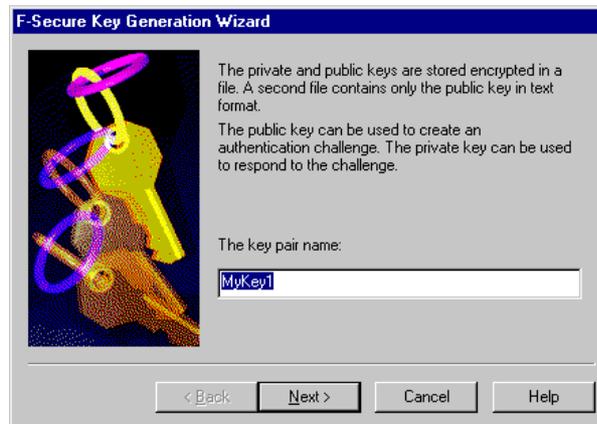
You can type a comment in the Comment field in this dialog box. You can use this field to identify the key later. For example, you can type in the name of the intended destination server.

You can enter a passphrase to be used with this key pair. Every time you connect to a server, you will be asked this passphrase. For maximum security, the passphrase should consist of at least 11 characters and should include

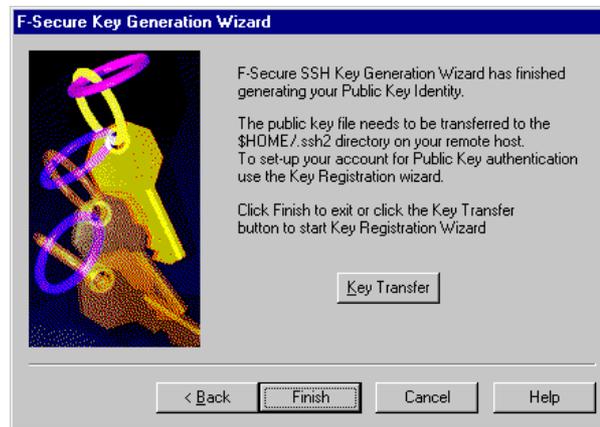
capital letters, numbers, and special characters. When you are typing your passphrase, the **Next** button is disabled until the contents of the Again field match the contents of the Passphrase field. You can leave the passphrase field empty, but this would let anyone connect from your computer to the servers utilizing this public key. This is highly discouraged. Click **Next** to continue.



To generate the key pair, click **Generate**. Once the keys are ready, click **Next**.



Give a name to the key pair. The key pair is saved into your system registry in `HKEY_CURRENT_USER\Software\Data Fellows\F-Secure SSH 2.0\Userkeys\key name`. The default key name is `MyKey1`, but if that name has already been used, the next available number is used. You can also define a completely different name to your key pair. Click **Next** to continue.



After you have created a key pair, you can either click **Finish** and close the Key Generation Wizard, or click **Key Transfer** to start the Key Registration Wizard.

## Key Registration Wizard

To use the Key Registration Wizard, you need to have created a key pair with the Key Generation Wizard. The Key Registration Wizard will transfer the public key to a host so that public-key authentication can be used with the host. The same public key can be transferred to any number of servers, but for security, you should use a different key with a different passphrase for each host.

You can launch the Key Registration Wizard from three different locations:

- Key Generation Wizard, immediately after you have created a key pair.
- Tools menu in the F-Secure SSH Client.
- Windows Start menu under Programs > F-Secure SSH Client.



In the Startup dialog box, click **Next** to start transferring keys.



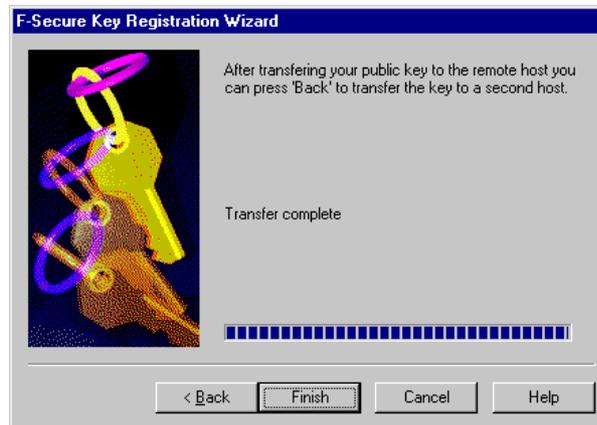
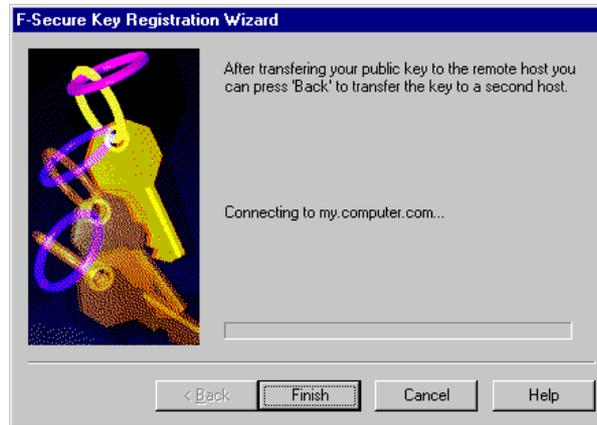
Select the key you want to transfer, by clicking on it. Click **Next** to continue.



Give the host name or IP address of the host you want to transfer the public key to, your user name on that host, and your password on that host. If that host uses non-default settings for ssh, click the **Advanced** button.



In the Advanced Settings dialog box, you can specify the port number on the remote host that ssh2 listens to, the directory used by ssh, and the name of the authorization file used on the remote host. The default values are shown in the screen shot. Click **OK** to close the Advanced settings dialog box. Click **Next** to initiate the key transfer.



If you have entered all the details correctly, the key will be transferred and properly initialized, so you can start using it immediately. To start using it, disconnect from the server, press ENTER, and select Public Key in the Authentication Method pane. Click **OK**. If you provided a passphrase when you created the key, you will be prompted for it. If you did not provide a passphrase, you will just be connected to the server.



## 3. F-Secure SSH 2 for UNIX

### 3.1 Overview

F-Secure SSH2 for UNIX consists of two separate programs: SSH 2 Client (ssh2) and SSH2 Server (sshd2).

F-Secure SSH2 server for UNIX provides secure login connections, file transfer, X11, and TCP/IP connections for F-Secure SSH2 clients over untrusted networks. The server uses cryptographic authentication, automatic session encryption, and integrity protection for all connections.

### 3.2 Installing F-Secure SSH 2 for UNIX

ssh2 should be installed as root to allow all users in the system access to it. To install ssh2 on your UNIX system, follow these steps:

1. Log in to the UNIX server as **root**.
2. Read the Readme.txt file included with the software for instructions on how to extract the distribution file and transfer it to your UNIX machine. The distribution file should go into the directory `/usr/local/src`
3. Change to the `/usr/local/src` directory.
4. Uncompress the distribution file as shown below. This will create a subdirectory `f-secure-ssh-2.0.x`, where `x` is the last two digits of the version number of SSH2. Uncompress the file in that directory.

```
$ tar xvfz f-secure-ssh-2.0.x.tar.gz
```

5. Change to the subdirectory `f-secure-ssh-2.0.x`.

```
$ cd f-secure-ssh-2.0.x
```

6. Configure ssh2. For configuration options, see the advanced installation section.

```
$ ./configure
```

7. Compile ssh2.

```
$ make
```

8. Install ssh2.

```
$ make install
```

ssh2 is now installed on your system. To recover hard disk space, you can delete the temporary files created by the installation, as follows:

```
$ make clean
```

To test for a successful installation, follow these steps:

1. Start the sshd2 daemon.

```
$ /usr/local/sbin/sshd2
```

2. Make an ssh connection to your local host.

```
$ ssh2 localhost
```

## Starting The Server

The server should be started at boot from `/etc/rc`, or the equivalent. It requires no arguments, an optional `"-b bits"` flag may be used to specify host key size. (The default size is 768 bits.) Key sizes less than 512 bits can be broken. Larger key sizes are usually more secure but they require more time to generate and to use. 2048 bits is secure with current technology.

The server is not started using `inetd`, because it needs to generate the key before serving the connection. This can take about a minute on slow machines. On a fast machine using a small (but breakable) key size of less than 512 bits, it may be feasible to start the server from `inetd` on every connection. The server must be given the `"-i"` flag if it is started from `inetd`.

## Common Installation Problems

### Backwards compatibility

In order to preserve backwards compatibility, you should not remove your old `sshd(1)` server, as the old SSH connection will be transferred to the old SSH server.

### 3.3 Using F-Secure SSH 2 for UNIX

ssh2 is used for starting a secure terminal connection to a remote host, executing commands on a remote host, and creating tunnels for the secure transfer of TCP packets and X11 connections. You can use ssh2 as a secure substitute for rlogin, rsh, and telnet. It provides secure, encrypted communications between two hosts over an unsecure network.

ssh2 connects and logs in to the specified hostname. A user must prove his identity to the remote machine by using one of the following authentication methods:

- password authentication
- public-key authentication

The format for starting an ssh2 session from the UNIX command prompt is:

```
ssh2 [options] host [command]
```

All options start with "-".

If your user name on the remote host is the same as on the host you are connecting from, you only need to give ssh2 the host name. ssh2 logs you in with your default user name and prompts you for your password on the remote host.

Example:

```
my.company.com% ssh2 second.host.com
andrew's password:
Last login: Wed Jun 16 08:48:59 1999
second.host.com%

my.company.com% ssh joseph@third.host.com
Executing /usr/local/bin/ssh1 for ssh1 compatibility.
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)? yes
Host 'third.host.com' added to the list of known hosts.
Creating random seed file ~/.ssh/random_seed. This may
take a while.
joseph@third.host.com's password:
Last login: Wed Jun 16 10:22:07 1999 from
my.company.com
Linux 2.0.35.
third.host.com:~$
```

If you have a different user name on the remote host, you need to include the user name in the command line when making a connection. You can either do this with the “-l” option or by using the format `username@remote.host`.

Example:

```
my.company.com% ssh -l josephthird.host.com
joseph's password:
Last login: Wed Jun 16 10:22:07 1999 from
my.company.com
Linux 2.0.35.
third.host.com:~$
```

ssh2 obtains configuration data from the following sources, in this order: (1) system's global configuration file (typically `/etc/ssh2/ssh2_config`); (2) user's configuration file (`$/HOME/.ssh2/ssh2_config`); and (3) command-line options. For each parameter, the last obtained value will be effective.

## Command Line Options

-l login_name	Specifies the user for login to the remote machine.
-n	Redirects input from <code>/dev/null</code> . (Do not read stdin.) This option can also be specified in the configuration file.
+a	Enables authentication agent forwarding. (Default)
-a	Disables authentication agent forwarding.
+x	Enables X11 connection forwarding. (Default)
-x	Disables X11 connection forwarding.
-i file	Specifies the identity file for public-key authentication. This option can also be specified in the configuration file.
-F file	Specifies an alternative configuration file to use. Specified options are in addition to those read in the <code>\$/HOME/.ssh2/ssh2_config</code> file.
-t	Allocates a tty, even if a command is given. This option can also be specified in the configuration file.

- v Enables verbose mode. Verbose debugging messages are displayed. Same as '-d 2'. This option can also be specified in the configuration file.
- d debug\_level Prints extensive debug information to stderr. 'debug\_level' is either a number (0 to 99) or a comma-separated list of assignments. Number 99 specifies that all debug information should be displayed. "ModulePattern=debug\_level".
- V Displays version string.
- q 'Quiet' mode. No warning messages will be displayed. This option can also be specified in the configuration file.
- f Fork into background after authentication. This option can also be specified in the configuration file.
- e char Set escape character. Use 'none' to disable. This option can also be specified in the configuration file. (Default; ~)
- c cipher Select encryption algorithm. Multiple -c options are allowed, and a single -c flag can have only one cipher. This option can also be specified in the configuration file.
- p port Specifies the port to connect to on the remote host. This option can be specified in the configuration file.
- P Do not use privileged source port. Prevents use of rhosts or rsarhosts authentications, but it can be used to bypass some firewalls that do not allow privileged source ports to pass. This option can also be specified in the configuration file. (Not yet implemented.)
- S Do not request a session channel. This can be used with port-forwarding requests if a session channel (and tty) is not needed or provided by the server.

- L port:host:hostport Specifies that the given port on the local (client) host is to be forwarded to the given host and port on the remote side. This works by allocating a socket to listen to on the local side. Whenever a connection is made to this port, the connection is forwarded over the secure channel and a connection is made to host:hostport from the remote machine. Port-forwardings can also be specified in the configuration file. Only root can forward privileged ports.
- R port:host:hostport Specifies that the given port on the remote (server) host is to be forwarded to the given host and port on the local side. This works by allocating a socket to listen to on the remote side. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to host:hostport from the local machine. Privileged ports can be forwarded only when logging in as root on the remote machine.
- +C Enables compression.
- C Disables compression. (Default)
- o 'option' Can be used to give options in the format used in the configuration files. This is useful for specifying options for which there is no separate command-line flag. The option has the same format as a line in the configuration file. Comment lines are not currently accepted by this option.
- h Displays brief help about command-line options.

## 3.4 Starting F-Secure sshd2

sshd2 (Secure Shell Daemon) is the daemon program for ssh2. Together, these programs replace rlogin and rsh programs, and provide secure encrypted communications between two untrusted hosts over an insecure network. The programs are intended to be as easy to install and to use as possible.

sshd2 is normally started at boot from `/etc/rc.local` or the equivalent. It forks a new daemon for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange.

sshd2 can be configured using command line options or a configuration file. Command line options override values specified in the configuration file.

The format for starting sshd2 from the UNIX command prompt is:

```
sshd2 [options]
```

### Command Line Options

- |                                    |  |
|------------------------------------|--|
| <code>-d debug_level_spec</code>   | Debug mode. The server sends verbose debug output to stderr. This option is only intended for debugging for the server. The debugging level is either a number, or a comma-separated list of assignments. "ModulePattern=debug_level".   |
| <code>-f configuration_file</code> | Specifies the name of the configuration file. The default is <code>/etc/ssh2/sshd2_config</code> .   |
| <code>-h host_key_file</code>      | Specifies the file from which the host key is read (default <code>/etc/ssh2/hostkey</code> ). If sshd2 is not run as root, the default host key file will be <code>HOME\$/.ssh2/hostkey</code> .   |
| <code>-o 'option'</code>           | Can be used to give options in the format used in the configuration files. This is useful for specifying options for which there is no separate command-line flag. The option has the same format as a line in the configuration file. Comment lines are not currently accepted. |

-p port	Specifies the port on which the server listens for connections. The default is 22.
-v	Enables verbose mode. Displays verbose debugging messages. Same as '-d 2'. This option can also be specified in the configuration file.
-q	Quiet mode. Nothing is sent to the system log. Normally the beginning, authentication, and termination of each connection is logged. This option can also be specified in the configuration file. (Logging is not implemented yet.)
-i	Specifies that sshd is being run from inetd.

## 3.5 Using scp2

scp2 (Secure Copy) is used in copying files over the network securely. It uses ssh2 for data transfer. It uses the same authentication, and provides the same security, as F-Secure ssh2. Unlike rcp, scp2 will prompt for any passwords or passphrases that are needed for authentication.

Any filename may contain a host, user, and port specification to indicate that the file is to be copied to or from that host. Copies between two remote hosts are permitted.

The format for copying files with scp2 is:

```
scp2 [options] [[username@]host[#port]:]file [[username@]host[#port]:]file_or_dir
```

For example, to copy the file program.exe from your local hard drive to second.host.com, where your username is *andrews*, you would enter:

```
scp2 program.exe andrews@second.host.com:program.exe
```

All options start with "-". The first file name is the source file and the second file name is the destination file. The host name needs to be given only if the host is a remote host. The user name is required only if it is different from the local user name. The port number is required only if it is not the standard ssh2 port (port 22).

## Command Line Options

-D debug_level_spec	Prints extensive debug information to stderr. debug_level_spec is a number from 0 to 99, where 99 specifies that all debug information should be displayed.
-d	With this option, scp2 will make sure that the destination file is a directory. If not, scp2 will exit with an error message.
-p	Tells scp2 to preserve file attributes and timestamps.
-n	Makes scp2 show what operations would have been done, without actually copying any files.
-v	Makes scp2 verbose. This is equal to specifying the '-D2' option.
-V	Displays version information.
-h	Displays brief help.
-c cipher	Selects the encryption algorithm. Multiple -c options are allowed, and a single -c flag can have only one cipher.
-S ssh2-path	Specifies the path to ssh2 used in connecting.
-P ssh2-port	Specifies the remote port to ssh2. Ports can also be defined on a file-by-file basis.
-t or -f	These options are reserved for ssh1-compatibility mode. If they are used with scp2, it executes ssh1 to handle the connection.

## 3.6 Using sftp2

Sftp (Secure File Transfer) is an ftp-like client that can be used in file transfer over the network. Sftp uses Ssh2 in data connections, so the file transport is secure.

The format for starting an sftp2 session from the UNIX command prompt is:

```
sftp [options] [[user]host[#port]]
```

All options start with "-".

### Command Line Options

-d debug_level_spec	Debug mode. Makes sftp send verbose debug output to stderr. The debugging level is either a number (0 to 99), or a comma-separated list of assignments. "ModulePattern=debug_level".
-c cipher	Select encryption algorithm. Multiple -c options are allowed; a single -c flag can have only one cipher.
-v	Enables verbose mode. Displays verbose debugging messages. Same as '-d 2'. This option can also be specified in the configuration file.
-S ssh2_path	Specifies the path to ssh2 used in connecting.
-p remote_port	Specifies the remote port to connect to.

After initiating the sftp session, you can execute the following commands:

? [keyword]	'?' prints the list of the available commands on the screen. '?' followed by a listed command gives a short description of the command.
bell	Enables and disables the option of ringing a bell after each file transfer.
bye	Disconnects from the server and quits the sftp2 client.
cd	Changes the directory on the remote server.
close	Closes the connection, but does not quit sftp2.

delete remote_filename	Deletes the specified remote file. This command does not accept wildcards
dir [remote_dir] [remote_file]	Lists the contents of the remote directory in long format.
disconnect	Same as the 'close' command.
get remote_file [local_file]	Downloads the specified file. If the destination file name is not given, the original name is used.
hash	Toggles the download/upload progress indicator on and off.
help	Same as '?'. Same as '?'.
lcd	Changes the local directory.
ldelete local_filename	Deletes the specified local file. Wildcards are not accepted.
ldir	Lists the contents of the current local directory in long format.
lls [local_dir] [local_file]	Lists the contents of the current local directory in short format.
lpwd	Prints the current working directory on the local machine.
lrm [local_file]	Deletes the specified local file. Wildcards are not accepted.
ls	Lists the contents of the current remote directory in short format.
mkdir	Creates a new directory on the remote host.
open host [port]	Opens a connection to the specified host, using the specified port number.
page	Toggles paging on and off. When paging is on, the output will be paused after each full screen of text.
prompt	Toggles interactive prompting. Interactive prompting occurs during multiple file transfers to allow the user to selectively retrieve or store files.
put local_file [remote_file]	Uploads a local file to the remote host.

<code>pwd</code>	Prints the current working directory on the remote machine.
<code>quit</code>	Equivalent to the 'bye' command.
<code>recv remote_file [local_file]</code>	Equivalent to the 'get' command.
<code>rm remote_file</code>	Equivalent to the 'delete' command.
<code>sort</code>	Toggles the sorting of directories on and off.
<code>status</code>	Displays the current status of sftp.
<code>timeout</code>	Sets the timeout for file transfers.
<code>user {user name}</code>	Identifies yourself to the remote FTP server.
<code>verbose</code>	Toggles verbose mode on and off.

## 3.7 Public-key authentication

### Generating Key Pairs with ssh-keygen2

Ssh-keygen2 generates and manages authentication keys for ssh2. Normally each user wanting to use ssh2 with public-key authentication runs this to create authentication keys in `$HOME/.ssh2/identity`. Additionally, the system administrator can use this to generate host keys for sshd2.

The format for creating an authentication key pair from the UNIX command prompt is:

```
ssh-keygen2 [options]
```

All options start with “-”.

By default, ssh-keygen2 creates a 1024-bit DSA key pair, prompts for a passphrase and passphrase verification, saves the private key to `$HOME/.ssh2/id_dsa_1024_a`, and saves the public key to `$HOME/.ssh2/id_dsa_1024_a.pub`.

Example:

```
% ssh-keygen2
Generating 1024-bit dsa key pair
 4 .oOo.oOOo.oO
Key generated.
1024-bit dsa, andrews@my.company.com, Thu Jul 22 1999
11:32:57 +0300
Passphrase :
Again      :
Private key saved to /home/andrews/.ssh2/id_dsa_1024_a
Public key saved to /home/andrews/.ssh2/id_dsa_1024_a.pub
%
Command Line Options
```

-b bits	Length of the key, in bits. For example, 2048 bits.
-t key_algorithm	The algorithm used in key generation. Currently, only DSS (Digital Signature Standard) is supported. (In some distributions, RSA is also supported.)
-c comment_string	Specifies the key's comment string.
-e file	Edit the specified key. Makes ssh-keygen2 interactive. You can change the key's passphrase or comment.
-p passphrase	Specify the passphrase used. -P specifies that the key will be saved with an empty passphrase.
-l file	Convert key from ssh1 format to ssh2 format.
-o output_file	Specify the output filename used.
-v	Print version string and exit.
-h   -\?	Print a short summary of ssh-keygen2 commands.
-q	Hide the progress indicator.
-r	Stir in data from stdin to the random pool.
-i	Display all information about a key. (Not yet implemented.)

## Files

### **\$HOME/.ssh2/random\_seed**

Used for seeding the random number generator. This file should not be readable by anyone but the user. This file is created the first time the program is run, and is updated every time.

### **\$HOME/.ssh2/id\_KEYTYPE\_KEYLEN\_X**

Private authentication keys.

### **\$HOME/.ssh2/id\_KEYTYPE\_KEYLEN\_X.pub**

Public authentication keys.

### **/etc/ssh2/hostkey**

### **/etc/ssh2/hostkey.pub**

Private and public sshd2(8) host keys

## Copying Public Keys to the Remote Host

To enable public-key authentication, you must copy your public key to the remote system you want to connect to. A secure way to do this is by using `scp2`.

1. If you have not yet done so, create a key pair using `ssh-keygen2`. The key pair will be created in your `$HOME/.ssh2` directory.
2. Go to your `$HOME/.ssh2` directory, and check the name of the private key you want to use for this connection. The private key has no file extension, and is usually something like `id_dsa_1024_a`.
3. Create or edit an existing file called `identification`. Add a line identifying your private key with "IdKey file\_name". Example:

```
IdKey id_dsa_1024_a
```

4. Now copy the public key of the same key pair to the remote host using `scp2`. Example:

```
% scp2 id_dsa_1024_a.pub
andrews@second.host.com:../ssh2/id_dsa_1024_a.pub
Accepting host second.host.com key without checking.
andrews@second.host.com's password:
Transferring id_dsa_1024_a.pub ->
second.host.com:../ssh2/id_dsa_1024_a.pub (1k)
|.....|
.....|
758 bytes transferred in 0.11 seconds [6.19 kB/sec].
%
```

5. Log in to the remote host with `ssh2` using password authentication.
6. Go to the `$HOME/.ssh2` directory on the remote server.
7. Create or edit a file called `authorization`, and add a line like `Key id_dsa_1024_a.pub` in the file. This is case-sensitive, so type `Key` with a capital 'K'.
8. You can now log off from the remote server. When you log back in, you will be prompted for your passphrase instead of your password, provided you entered a passphrase when creating the key pair. Otherwise, you will be logged in automatically.

## ssh-agent2

*ssh-agent2* is a program that holds authentication private keys. *ssh-agent2* is started in the beginning of an X-session or a login session, and all other windows or programs are started as children of the *ssh-agent2* program. (The command normally starts X or is the user shell.) Programs started under the agent inherit a connection to the agent, and the agent is automatically used for public-key authentication when logging in to other machines using ssh.

The format for starting the *ssh-agent2* service from the UNIX command prompt is:

```
ssh-agent2 [command]
```

If you have logged into a UNIX system from another non-native UNIX platform, you will not be able to successfully fork the *ssh-agent2* to the background. In this case, start the *ssh-agent2* service with the following command:

```
ssh-agent2 $SHELL
```

This will start another shell on top of the original shell, with *ssh-agent2* running in the background. When you exit the shell with a standard UNIX command, such as `exit`, `logout`, and `quit`, you will be brought back to the original shell from which you started *ssh-agent2*.

If *ssh-agent2* is started without any arguments (with no command), it will fork and launch the agent in the background. The agent also prints a command that can be evaluated in `sh` or `csh`, like shells. The command will set the `SSH_AUTHENTICATION_SOCKET` and `SSH_AGENT_PID` environment variables.

The `SSH_AGENT_PID` environment variable can be used to kill the agent when it is no longer needed (for example, when you log out of X-session). If no options are given, *ssh-agent2* uses the `SHELL` environment variable to detect what kind of shell you have (`csh`- or `sh`-style shell). The `-c` option will force the `csh`-style shell, and the `-s` option will force the `sh`-style shell.

In SysV variants (for IRIX and Solaris, at least), the environment variable `SHELL` might not contain the actual value of the shell executing the evaluation. If `ALTSHELL` is set to `YES` in `/etc/default/login`, the `SHELL` environment variable is set to the login shell of the user.

Initially, the agent does not have any private keys. Keys are added using *ssh-add2*. When executed without arguments, *ssh-add2* adds the `$HOME/.ssh2/identity` file. If the identity has a passphrase, *ssh-add2* requests the passphrase (using a small X11 application if running under X11, or from the

terminal if running without X11). *ssh-add2* then sends the identity to the agent. Several identities can be stored in the agent, and any of them can be used automatically by the agent. *ssh-add2 -l* displays the identities currently held by the agent.

The agent is run on a user's local PC, laptop, or terminal. Authentication data need not be stored on any other machine, and authentication passphrases never travel over the network. However, the connection to the agent is forwarded over SSH remote logins. Thus, the user can use the privileges given by the identities anywhere on the network securely.

A connection to the agent is inherited by child programs. A UNIX domain socket is created (*/tmp/ssh- $\$$ USER/agent-socket- $\langle$ pid $\rangle$* ), where ' $\langle$ pid $\rangle$ ' is the process id of the listener (agent or *sshd* proxying the agent). The name of this socket is stored in the *SSH\_AUTHENTICATION\_SOCKET* environment variable. The socket is made accessible only to the current user. This method is easily abused by root or another instance of the same user. Older versions of SSH used inherited file descriptors for contacting the agent and used the UNIX-domain sockets in an incompatible way.

If the command is given as an argument for *ssh-agent2*, the agent exits automatically when the command (from the command line) terminates. The command is executed even if the agent fails to start its key-storing and challenge-processing services.

## Files

### ***$\$$ HOME/.ssh2/id\_KEYTYPE\_KEYLEN\_XX***

Contains the private-key authentication identity of the user. This file should not be readable by anyone but the user. It is possible to specify a passphrase when generating the key. This passphrase will be used to encrypt the private part of this file. This file is not used by *ssh-agent2*, but is normally added to the agent by using *ssh-add2* when logging in.

### ***/tmp/ssh- $\$$ USER/agent-socket- $\langle$ pid $\rangle$***

UNIX-domain sockets used to contain the connection to the authentication agent. These sockets should only be readable by the owner. The sockets should get automatically removed when the agent exits. The parent directory of *ssh2- $\$$ USER* must have its sticky bit set.

## ssh-add2

ssh-add2 adds identities to the authentication agent, ssh-agent2. When run without arguments, it adds the file \$HOME/.ssh2/identity. Alternative file names can be given on the command line. The authentication agent must be running and must be an ancestor of the current process for ssh-add2 to work.

The format for starting *ssh-add2* from the UNIX command prompt is:

```
ssh-add2 [options] [files...]
```

If any file requires a passphrase, ssh-add2 requests the passphrase from the user. If the -p option is given, the passphrase is read from stdin. If the -p option is not given, and if the user is using X11, the passphrase is requested using a small X11 program. Otherwise, it is read from the user's tty. (It may be necessary to redirect stdin from /dev/null to get the passphrase requested using X11.)

If ssh-add2 needs a passphrase, it will read the passphrase from the current terminal if it was run from a terminal. If ssh-add2 does not have a terminal associated with it, but DISPLAY is set, it will open an X11 window to read the passphrase. This is particularly useful when calling ssh-add2 from an .Xsession or related script.

 **On some machines, it may be necessary to redirect the input from /dev/null to make this work.**

## Command Line Options

Option	Description
-p	Read passphrase from stdin (or pipe).
-l	Lists all identities currently represented by the agent.
-d	Instead of adding the identity, removes the identity from the agent.
-D	Deletes all identities from the agent.

## Return Status

ssh-add2 returns one of the following exit statuses, which may be useful in scripts.

- 0 The requested operation was performed successfully.
- 1 No connection could be made to the authentication agent. Presumably, there is no authentication agent active in the execution environment of ssh-add2.
- 2 The user did not supply a required passphrase.
- 3 An identify file could not be found, was not readable, or was in bad format.
- 4 The agent does not have the requested identity.
- 5 An unspecified error has occurred. This is a catch-all for errors not listed above.

## Files

`$HOME/.ssh2/identification` Contains names of the private keys that are to be used in authentication. See `ssh2(1)` for more information.

`$HOME/.ssh2/id_KEYTYPE_KEYLEN_X`

`$HOME/.ssh2/id_KEYTYPE_KEYLEN_X.pub`

## 3.8 Advanced Installation

### Configuration Options

The F-Secure SSH 2 for UNIX comes with a configuration script generated by **autoconf**. This script has several options.

**Usage:** `configure [options] [host]`

**Options:** [defaults in brackets after descriptions]

**Configuration:**

<code>--cache-file=FILE</code>	Cache test results in FILE.
<code>--help</code>	Print the command line options.
<code>--no-create</code>	Do not create output files.
<code>--quiet, --silent</code>	Do not print 'checking...' messages.
<code>--version</code>	Print the version of <i>autoconf</i> that created configure.

**Directory and file names:**

<code>--prefix=PREFIX</code>	Install architecture-independent files in PREFIX [usr/local].
<code>--exec-prefix=EPREFIX</code>	Install architecture-dependent files in EPREFIX [same as prefix].
<code>--bindir=DIR</code>	User executables in DIR [EPREFIX/bin].
<code>--sbindir=DIR</code>	System admin executables in DIR [EPREFIX/sbin].
<code>--libexecdir=DIR</code>	Program executables in DIR [EPREFIX/libexec].
<code>--datadir=DIR</code>	Read-only, architecture-independent data in DIR [PREFIX/share].
<code>--sysconfdir=DIR</code>	Read-only, single-machine data in DIR [PREFIX/etc].
<code>--sharedstatedir=DIR</code>	Modifiable architecture-independent data in DIR [PREFIX/com].

<code>--localstatedir=DIR</code>	Modifiable single-machine data in DIR [PREFIX/var].
<code>--libdir=DIR</code>	Object code libraries in DIR [EPREFIX/lib].
<code>--includedir=DIR</code>	C header files in DIR [PREFIX/include].
<code>--oldincludedir=DIR</code>	C header files for non-gcc in DIR [/usr/include].
<code>--infodir=DIR</code>	Info documentation in DIR [PREFIX/info].
<code>--mandir=DIR</code>	Man documentation in DIR [PREFIX/man].
<code>--srcdir=DIR</code>	Find the sources in DIR [configure dir or ..].
<code>--program-prefix=PREFIX</code>	Attach PREFIX to installed program names.
<code>--program-suffix=SUFFIX</code>	Append SUFFIX to installed program names.
<code>--program-transform-name=PROGRAM</code>	Run <i>sed</i> PROGRAM on installed program names

**Host type:**

<code>--build=BUILD</code>	Configure for building on BUILD [BUILD=HOST].
<code>--host=HOST</code>	Configure for HOST [guessed].
<code>--target=TARGET</code>	Configure for TARGET [TARGET=HOST].

**Features and packages:**

<code>--disable-FEATURE</code>	Do not include FEATURE (same as <code>--enable-FEATURE=no</code> ).
<code>--enable-FEATURE[=ARG]</code>	Include FEATURE [ARG=yes].
<code>--with-PACKAGE[=ARG]</code>	Use PACKAGE [ARG=yes].
<code>--without-PACKAGE</code>	Do not use PACKAGE (same as <code>--with-PACKAGE=no</code> ).
<code>--x-includes=DIR</code>	X include files are in DIR.
<code>--x-libraries=DIR</code>	X library files are in DIR.

**--enable and --with options recognized:**

<code>--enable-verbose-warnings</code>	Enable verbose compiler warnings..
<code>--enable-debug</code>	Enable light debugging.
<code>--enable-debug-heavy</code>	Enable heavy debugging.

<code>--enable-efence</code>	Enable EFENCE memory allocation debugger.
<code>--disable-asm</code>	Disable assembler optimized subroutines.
<code>--with-x</code>	Use the X Window System.
<code>--with-securid[=PATH]</code>	Enable support for Security Dynamics SecurID card. [Not implemented yet.]
<code>--with-tis[=DIR]</code>	Enable support for TIS authentication server. [Not implemented yet.]
<code>--with-libwrap[=PATH]</code>	Compile in libwrap (tcp_wrappers) support.
<code>--enable-group-writeability</code>	Allow group writeability in host-based and public-key authentication.
<code>--with-ssh-agent1-compat</code>	Include ssh-agent1 compatibility. (default)
<code>--without-ssh-agent1-compat</code>	Leave out ssh-agent1 compatibility.
<code>--with-socks-server=VALUE</code>	Use VALUE as default SSH SOCKS_SERVER value.
<code>--with-etcdir=PATH</code>	Directory containing ssh system files. (default /etc)
<code>--disable-suid-ssh-signer</code>	Install ssh-signer without suid bit.
<code>--enable-tcp-port-forwarding</code>	Enable port forwarding support. (default)
<code>--disable-tcp-port-forwarding</code>	Disable port forwarding support. (default)
<code>--enable-X11-forwarding</code>	Enable X11 forwarding support.
<code>--disable-X11-forwarding</code>	Disable X11 forwarding support.
<code>--with-idea</code>	Include IDEA cipher.
<code>--without-idea</code>	Do not use IDEA cipher. (default)
<code>--enable-tcp-nodelay</code>	Enable TCP_NODELAY socket option. (default=off)
<code>--enable-blocking-connect</code>	Use blocking connect system call. (default=off)
<code>--with-internal-localtime</code>	Use only ssh internal localtime function.
<code>--without-internal-localtime</code>	Use system provided localtime function.
<code>--with-pgp</code>	Include PGP library. (default)
<code>--without-pgp</code>	Do not include PGP library.

(If you have already configured, and later want to supply some values from the command line, you may need to run *make distclean* before reconfiguring.)

## Makefile

The Makefile is generated from *Makefile.in* by running `configure`. It supports the following targets (among others):

<code>all:</code>	Compile everything.
<code>install:</code>	Install in <code>\$exec_prefix/bin</code> and <code>\$prefix/man/man1</code> .
<code>hostinstall:</code>	Generate host-key and install config files.
<code>clean:</code>	Remove object files and executables.
<code>distclean:</code>	Remove everything not in the distribution.

## 3.9 Configuring F-Secure SSH 2 for UNIX

### Configuration Files

`ssh2` obtains configuration data from the following sources, in this order: system's global configuration file (typically `/etc/ssh2/ssh2_config`), user's configuration file (`$HOME/.ssh2/ssh2_config`); and command-line options. For each parameter, the last obtained value will be effective.

The configuration file has the following format:

```
    'expression:'
```

'*Expression*' denotes the start of a per-host configuration block, where 'expression' is an arbitrary string which distinguishes this block from others. 'Expression' can contain wildcards. 'Expression' will be compared with the hostname obtained from the command line, and if it matches, the block will be evaluated. Evaluation stops at the next 'expression:' statement. If more than one match is found, all will be evaluated and the last obtained values for parameters will be effective. Note that 'expression' does not have to be a real hostname, as long as the 'expression' block contains a 'Host' configuration parameter, where the real hostname to connect is defined.

Empty lines and lines starting with '#' are ignored as comments. Otherwise the format of a line is "keyword arguments". It is possible to enclose arguments in quotation marks, and use standard C convention. The possible keywords and

their meanings are as follows. (Note that the configuration files are case-sensitive, but keywords are not case-sensitive.)

'*Host*' specifies the real host name to log in to. With 'expression' above, this can be used to specify nicknames or abbreviations for hosts. The default is the name given on the command line. Numeric IP addresses are also permitted (both on the command line and in HostName specifications).

'*User*' specifies the user to log in as. This can be useful if you have a different user name on different machines. You will not need to enter the user name on the command line.

Option	Description
ForwardAgent	Specifies whether the connection to the authentication agent (if any) will be forwarded to the remote machine. The argument must be "yes" or "no".
ForwardX11	Specifies whether X11 connections will be automatically redirected over the secure channel and DISPLAY set. The argument must be "yes" or "no".
PasswordAuthentication	Specifies whether to use password authentication. The argument must be "yes" or "no".
RhostsAuthentication	Specifies whether to try rhosts-based authentication. Note that this declaration only affects the client side and has no effect whatsoever on security. Disabling rhosts authentication may reduce authentication time on slow connections when rhosts authentication is not used. Most servers do not permit RhostsAuthentication because it is not secure (see RhostsRSAAuthentication). The argument must be "yes" or "no". (Not yet implemented.)
RhostsPubKeyAuthentication	Specifies whether to try rhosts-based authentication with public-key host-authentication. This is the primary authentication method for most sites.

RHostsRSAAuthentication	A synonym for this keyword, and it is defined for backwards compatibility with ssh1. The argument must be "yes" or "no". (Not yet implemented.)
PubKeyAuthentication	Specifies whether to try public-key authentication. Public-key authentication will only be attempted if the identity file exists or an authentication agent is running. RSAAuthentication is a synonym for this keyword and is defined for backward compatibility with ssh1. The argument must be "yes" or "no".
Port	Specifies the port number to connect on the remote host. Default is 22.
Ciphers	Specifies the ciphers to use for encrypting the session. Currently, DES, 3DES, Blowfish, Idea, and Arcfour are supported. DES, 3DES, and Arcfour are included in all distributions. Multiple ciphers can be specified as a comma-separated list. Special values to this option are 'any' and 'anycipher'. Both of these values either allow any available cipher or only exclude 'none' (the non-encrypting cipher mode).
IdentityFile	Specifies the name of the user's identification file.
AuthorizationFile	Specifies the name of the user's authorization file.
RandomSeedFile	Specifies the name of the user's random seed file.
ForcePTTYAllocation	For tty allocation; that is, allocate a tty even if a command is given. The argument must be "yes" or "no". (Not yet implemented.)
VerboseMode	Causes ssh2 to print debugging messages about its progress. This is helpful in debugging connection, authentication, and configuration problems.
QuietMode	Causes all warnings and diagnostic messages to be suppressed. Only fatal errors are displayed. The argument must be "yes" or "no".
Compression	Specifies whether to use compression. The argument must be "yes" or "no".

FallbackToRsh	Specifies that if connecting via ssh2 fails due to a connection-refused error (there is no sshd2 listening on the remote host), rsh should automatically be used instead (after a suitable warning about the session being unencrypted). The argument must be "yes" or "no". (Not yet implemented.)
KeepAlive	Specifies whether the system should send keepalive messages to the other side. If they are sent, death of the connection or crash of one of the machines will be properly noticed. However, this means that connections will die if the route is down temporarily (which can be annoying). The default is "yes" (to send keepalives), and the client will notice if the network goes down or the remote host dies. This is important in scripts and many users want it. To disable keepalives, the value should be set to "no" in both the server and the client configuration files. (Not yet implemented.)
UseRsh	Specifies that rlogin/rsh should be used. It is possible that the host does not support the ssh2 protocol. This causes ssh2 to immediately exec rsh. All other options are ignored if this has been specified. The argument must be "yes" or "no". (Not yet implemented.)
BatchMode	If set to "yes", passphrase/password querying will be disabled. This option is useful in scripts and other batch jobs where you have no user to supply the password. The argument must be "yes" or "no". (Not yet implemented.)
StrictHostKeyChecking	If this flag is set to "yes", ssh2 will not automatically add host keys. This provides maximum protection against Trojan horse attacks. However, it can be somewhat annoying if you frequently connect to new hosts. If this is set to "no", then the new host will automatically be added to the known host files. In both cases, the host keys of known hosts will be verified automatically. (Not yet implemented.)

EscapeChar	Sets the escape character (default: ~). The escape character can also be set from the command line. The argument should be the single character '^' followed by a letter or the word "none" (which will disable the escape character entirely, making the connection transparent for binary data).
PasswordPrompt	Sets the password prompt that the user sees when connecting to a host. Variables '%U' and '%H' can be used to give the user's login name and host, respectively.
GoBackground	Requests ssh2 to run in the background after authentication is done and forwardings have been established. This is useful if ssh2 is going to ask for passwords or passphrases, but the user wants it in the background. The argument must be "yes" or "no". (Not yet implemented.)
UseNonPrivilegedPort	Specifies whether to use a privileged port when connecting to other end. The default is "no" if rhosts or rsarhosts-authentications are enabled. The argument must be "yes" or "no". (Not yet implemented.)
DontReadStdin	Redirect input from /dev/null (that is, do not read stdin). The argument must be "yes" or "no".
SSH1Path	Specifies the path to ssh1 client, which is executed if the server supports only ssh1.x protocols. The arguments for ssh2 are passed to the ssh1 client.

## Environment

ssh2 will normally set the following environment variables:

Variable	Description
DISPLAY	The DISPLAY variable indicates the location of the X11 server. It is automatically set by ssh2 to point to a value in the form "hostname:n", where 'hostname' indicates the host where the shell runs, and 'n' is an integer greater or equal to 1. ssh2 uses this special value to forward X11 connections over the secure channel. The user should normally not explicitly set DISPLAY, because that will render the X11 connection insecure, and will require the user to manually copy any required authorization cookies.
HOME	Set to the path of the user's home directory.
LOGNAME	Synonym for USER. Set for compatibility with systems using this variable.
MAIL	Set to point the user's mailbox.
PATH	Set to the default PATH, as specified when compiling ssh2 or (on some systems) /etc/environment or /etc/default/login.
SSH SOCKS_SERVER	If SOCKS is used, it is configured with this variable. The format of the variable is: <i>socks://user-name@socks_server:port/network/netmask, network/netmask...</i> For example, setting the environment variable, SSH SOCKS_SERVER, to <i>socks://mylogin@socks.ssh.fi:1080/203.123.0.0/16,198.74.23.0/24</i> uses the host socks.ssh.fi port 1080 as your SOCKS server if a connection is attempted outside of networks 203.123.0.0 (16-bit domain) and 198.74.23.0 (8-bit domain), which are connected directly.

Default value for SSH SOCKS\_SERVER variable can be specified at compile time by specifying `--with-socks-server=VALUE` on the configure command line when compiling ssh2. The default value can be canceled by setting SSH SOCKS\_SERVER to an empty string, and overridden by setting SSH SOCKS\_SERVER to a new value.

#### SSH2\_AUTHENTICATION\_SOCKET

If it exists, this variable is used to indicate the path of a UNIX-domain socket used to communicate with the authentication agent (or its local representative).

#### SSH2\_CLIENT

Identifies the client end of the connection. The variable contains three values (separated by spaces): client IP-address, client port number, and server port number.

#### SSH2\_TTY

This is set to the name of the tty (path to the device) associated with the current shell or command. If the current session has no tty, this variable is not set.

#### TZ

The timezone variable is set to indicate the present time zone, if it was set when the daemon was started. (That is, the daemon passes the value to new connections.)

#### USER

Set to the name of the user logging in. Additionally, ssh2 reads `/etc/environment` and `$HOME/.ssh2/environment`, and adds lines in the format `VARNAME=value` to the environment. Some systems may still have additional mechanisms for setting up the environment, such as `/etc/default/login` on Solaris.

FILES\$HOME/.ssh2/random_seed	Used for seeding the random number generator. This file contains sensitive data and should be read/write for the user and not accessible for others. This file is created the first time the program is run and updated automatically. The user should never need to read or modify this file.
\$HOME/.ssh2/ssh2_config	The per-user configuration file. The format of this file is described above. This file is used by the ssh2 client. This file does not usually contain any sensitive information, but the recommended permissions are read/write for the user, and not accessible by others.
\$HOME/.ssh2/identification	Contains information on how the user wants to authenticate himself when contacting a specific host. The identification file has the same general syntax as the configuration files. The following keywords can be used:
IdKey	This is followed by the file name of a private key in the \$HOME/.ssh2 directory used for identification when contacting a host. If there is more than one IdKey, they are tried in the order that they appear in the identification file.
\$HOME/.ssh2/authorization	Contains information on how the server will verify the identity of an user. The authorization file has the same general syntax as the configuration files. The following keywords may be used:
Key	This is followed by the filename of a public key in the \$HOME/.ssh2 directory that is used for identification when contacting the host. If there is more than one key, they are all acceptable for logging in.
Command	When used, this keyword must follow the "Key"-keyword above. This is used to specify a "forced command" that will be executed on the server side instead of anything else when the user is authenticated. A command supplied by the user is put in the environment variable.

"SSH2\_ORIGINAL\_COMMAND". The command is run on a pty if the connection requests a pty; otherwise it is run without a tty. A quote may be included in the command by quoting it with a backslash. This option might be useful to restrict certain RSA keys to perform just a specific operation. An example might be a key that permits remote backups, but nothing else.

The client may specify TCP/IP and/or X11 forwardings unless they are explicitly prohibited.

## 3.10 Configuring F-Secure sshd2 for UNIX

### Configuration File

sshd2 reads configuration data from `/etc/ssh2/sshd2_config` (or the file specified with `-f` on the command line). The file contains keyword value pairs, one per line. Lines starting with `#` and empty lines are interpreted as comments.

The following keywords are possible. Keywords are not case sensitive.

ForwardAgent	Specifies whether the connection to the authentication agent (if any) will be forwarded to the remote machine. The argument must be "yes" or "no".
ForwardX11	Specifies whether X11 connections will be automatically redirected over the secure channel and DISPLAY set. The argument must be "yes" or "no".
PasswordAuthentication	Specifies whether to use password authentication. The argument must be "yes" or "no".

RHostsAuthentication	Specifies whether authentication using rhosts or /etc/hosts.equiv files is sufficient. Normally, this method should not be permitted because it is insecure. RhostsPubKeyAuthentication should be used instead, because it performs RSA-based host authentication in addition to normal rhosts or /etc/hosts.equiv authentication. The argument must be "yes" or "no". (Not yet implemented.)
RHostsPubKeyAuthentication	Specifies whether rhosts or /etc/hosts.equiv authentication, together with successful public-key host authentication, is allowed. RHostsRSAAuthentication is a synonym for this keyword, and it is defined for backwards compatibility with ssh1. The argument must be "yes" or "no". (Not yet implemented.)
PubKeyAuthentication	Specifies whether to try public-key authentication. RSAAuthentication is a synonym for this keyword, and it is defined for backwards compatibility with ssh1. The argument must be "yes" or "no".
Port	Specifies the port number that sshd2 listens on. The current default is 22.
Ciphers	Specifies the ciphers to use for encrypting the session. DES, 3DES, Blowfish, Idea and Arcfour are currently supported. DES, 3DES, and Arc-four are in all distributions. Multiple ciphers can be specified as a comma-separated list. Special values to this option are 'any' and 'anycipher'. These values either allow any available cipher or only exclude 'none' (the non-encrypting cipher mode).
AuthorizationFile	Specifies the name of the user's authorization file.
RandomSeedFile	Specifies the name of the random-seed file.
ForcePTYAllocation	Allocates a tty even if a command is given. The argument must be "yes" or "no". (Not yet implemented.)

VerboseMode	Verbose mode. Causes sshd2 to print debugging messages about its progress. This is helpful in debugging connection, authentication, and configuration problems.
QuietMode	Specifies whether the system runs in quiet mode. In quiet mode, only fatal errors are logged in the system log. The argument must be "yes" or "no".
KeepAlive	<p>Specifies whether the system should send keep-alive messages to the other side. If they are sent, death of the connection or crash of one of the machines will be properly noticed. However, this means that connections will die if the route is down temporarily, which can be annoying. On the other hand, if keepalives are not sent, sessions may hang indefinitely on the server, leaving "ghost" users and consuming server resources.</p> <p>The default is "yes" (to send keepalives), and the server will notice if the network goes down or the client host reboots. This avoids continuously hanging sessions.</p> <p>To disable keepalives, the value should be set to "no" in both the server and the client configuration files. (Not yet implemented.)</p>
IgnoreRhosts	Specifies that rhosts and shosts files will not be used in authentication. /etc/hosts.equiv and /etc/shosts.equiv are still used. (Not yet implemented.)
PermitRootLogin	Specifies whether the root can log in using ssh2. The argument must be "yes" or "no".
PermitEmptyPasswords	When password authentication is allowed, it specifies whether the server allows login to accounts with empty password strings. The argument must be "yes" or "no".

StrictModes	Specifies whether sshd2 should check file modes and ownership of the user's home directory and rhosts files before accepting login. This is normally desirable because novices sometimes accidentally leave their directory or files world-writable. The default is "yes". The argument must be "yes" or "no".
PrintMotd	Specifies whether sshd2 should print /etc/motd when a user logs in interactively. The default is "yes". The argument must be "yes" or "no".
ListenAddress	Specifies the IP address of the interface where the sshd2 server socket is bind.
HostKeyFile	Specifies the file containing the private host key (default /etc/ssh2/hostkey).
PublicHostKeyFile	Specifies the file containing the public host key (default /etc/ssh2/hostkey.pub). Note: In most cases the order of config parameters is not an issue. Here it is safe if you first specify HostKeyFile before this parameter.
LoginGraceTime	The server disconnects after a designated time (in seconds) if the user has not successfully logged in. If the value is 0, there is no time limit. The default is 600 seconds. (Not yet implemented.)
PasswordGuesses	Specifies the number of attempts allowed by the user when using password authentication. The default is 3.
Sshd1Path	Specifies the path to the sshd1 daemon, which is executed if the client supports only SSH 1.x protocols. The arguments for sshd2 are passed on to sshd1.

## 4. F-Secure SSH 2.0 for Macintosh

### 4.1 Overview

The new F-Secure SSH 2.0 for Macintosh has been totally rewritten. New functionality and many user-friendly options have been added. The user interface has also been improved.

F-Secure Client for Macintosh provides users with secure login connections over untrusted networks. The program supports secure TCP/IP port-forwarding technology to connect arbitrary and otherwise insecure connections over a secure channel.

The F-Secure SSH 2 for Macintosh client is only compatible with F-Secure SSH for UNIX server version 2.0.12 or later.

### 4.2 Installation Guide

To install F-Secure SSH Client 2.0 for Macintosh, follow these steps:

1. Insert the F-Secure CD into the CD-ROM drive.
2. Double-click the F-Secure CD icon on your desktop.
3. Drag and drop the F-Secure SSH 2.0 folder from the CD window to the desired destination on your workstation.
4. To add an alias to the F-Secure SSH 2.0 Client on the desktop, go to the F-Secure SSH 2.0 folder on your local hard drive and drag the F-Secure SSH 2.0 icon to the desktop while holding down **OPTION+⌘**.

## 4.3 Using the Menus

### File Menu

Menu Item	Description
New Group	Create a new group in the Connection Manager. Selecting this item adds a new group to the Connection Manager. See Section 4.4, "Managing Groups".
New Terminal	Create a new terminal connection to an SSH server. Opens the Terminal Properties dialog box. See Section 4.6, "Creating a new Terminal Connection," on page 112.
New TCP Tunnel	Create a new TCP tunnel. Opens the Tunneling Properties dialog box. See Section 4.8, "Tunnels" on page 122.
Open	Open an existing group, terminal, or tunnel.
Close	Hide the currently active window.
Save	Save a group, terminal, or tunnel connection. Only top-level items can be saved. If you want to save an item that is within a lower-level group, either move it to the top level first or select it and choose Put Away, in which case you will be prompted to save it. The Put Away command removes the selected item and everything underneath it from the Connection Manager, and the change will be saved along with the top-level group.
Save As...	Save a group, terminal, or tunnel that you have opened from a file under a different name.
Clone	Make an exact copy of the selected group, terminal, or tunnel in the Connection Manager.
Put Away	Remove the selected group, terminal, or tunnel from the Connection Manager.
Create Public Key Files	Create a key-pair for public-key authentication. For more information on public-key authentication, see Section 4.11 on page 139.
Quit	Close all connections and quit SSH.

## Edit Menu

The Edit menu is a standard Macintosh menu. The Undo command has no application in F-Secure SSH 2.0 Client and the Cut command can only be used in the text entry fields of dialog boxes.

Menu Item	Description
Undo	Undo has no application in F-Secure SSH 2.0 Client.
Cut	Cut the selected contents of a text entry field in a dialog box to the Clipboard. Text displayed in the terminal window cannot be cut; it can only be copied.
Copy	Copy the selected text in any window to the Clipboard.
Paste	Paste text from the Clipboard to the terminal prompt, to a text entry field in the dialog boxes or to the Known Host Keys window.
Clear	Delete the selected text in a text entry field in the Properties or Preferences dialog boxes.
Copy Table	Copy the selected text from the terminal to the Clipboard in tabulated format.

## SSH Menu

Menu Item	Description
Known Host Keys	Show a list of hosts whose host keys you have verified and accepted.
Connection Manager	Shows the Connection Manager as the active window.
Preferences	Opens the general SSH preferences dialog box. For more information on SSH preferences, see Section 4.10, "F-Secure SSH2 Preferences," on page 131.
Show Properties/Show Terminal	Open the Properties dialog box of the currently selected group, terminal, or tunnel. If a Terminal Properties dialog box is the active window, the menu item is Show Terminal. If any other Properties dialog box is the active window, this menu item is disabled. For more information on terminal properties, see Section 4.7, "Terminal Properties," on page 115. For tunneling properties, see Section 4.9, "Tunnels," on page 124.
Connect	Connect the selected group, terminal, or tunnel to the server defined in its Properties dialog box. This option is only available if the active group, terminal, or tunnel is not connected to the server.
Disconnect	Disconnect the selected group, terminal, or tunnel from the server. This option is only available if the selected group, terminal, or tunnel is connected to the server.
Switch Back	Switch to previous terminal in the Connection Manager.
Switch Forward	Switch to next terminal in the Connection Manager.
Open connections	A list of all the terminal connections you have open. You can select the active terminal from this list. The active terminal is checked.

## Shortcuts Menu

You can save groups, terminals, and tunnels in the Shortcuts folder under the F-Secure SSH 2.0 Client for Macintosh folder, and they will appear in the Shortcuts menu. When saving the file, if you add `/[ANY CHARACTER]` after the name, you can open the item at any time by pressing `⌘[CHARACTER]`. For example, saving a terminal with the name *myterm/J* in the Shortcuts folder will enable you to open the terminal from the F-Secure SSH 2.0 Client by pressing `⌘J`.

Menu Item	Description
Read Me	Opens the F-Secure SSH 2.0 Client Readme file.

## Help Menu

Menu Item	Description
About Balloon Help	View information about Balloon Help.
Show/Hide Balloons	Select whether or not you want to use Balloon Help. There is no SSH-specific Balloon Help available in this version of the program.

## 4.4 Connection Manager

The Connection Manager is a utility for managing all your terminal connections and TCP tunnels from a single window. By grouping tunnels and terminals together, you can open them all by opening the group. Groups can include other groups.

### Managing Groups

To create a new group, go to the File menu and select New Group, or click the **New Group** button on the Connection Manager. The new group will be placed in the Connection Manager. The first group will be placed on the left side of the Connection Manager.

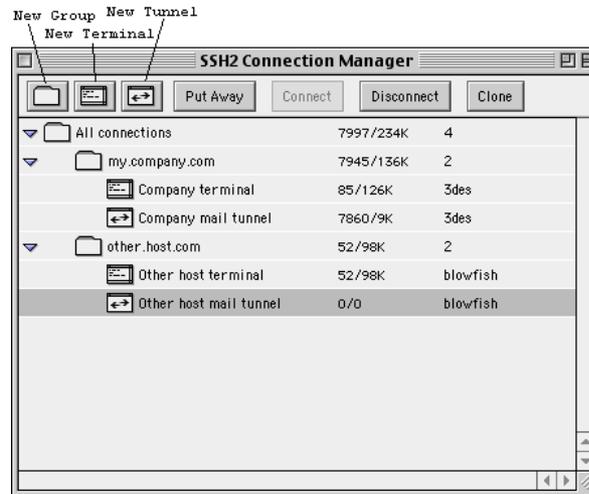
All items in the Connection Manager can be arranged hierarchically. The hierarchy runs from left to right in the Connection Manager; the main group is on the left. New items will appear on the same level as the selected item when the new item is added. If the Connection Manager is empty, the new item will be placed on the top level (the left side of the Connection Manager). Opened items will always appear on the top level.

This hierarchy allows you to simultaneously open several terminals and tunnels and groups of terminals and tunnels. The number of levels is limited by the width of the screen. If you need to edit deeper hierarchies, you can move embedded groups to the top level, edit them and move them back to their place.

Tunnels and terminals can only be created inside a group or as stand-alone items at the top level; not beneath each other.

You can move an item to a different level by dragging it to a group or to the top level.

Below is an example of a two-level grouping. The smaller groups, *my.company.com* and *other.host.com*, can be opened separately if only one group is needed, as long as they have been saved separately. If all connections are needed at the same time, a top-level group called 'All connections' has been created; it includes both groups. In this way, all of the connections can be opened at once simply by opening the 'All connections' group.



## Connection Manager Window

The Connection Manager window consists of the connection information screen and a toolbar with the following buttons:

Button	Description
New Group	Create a new group in the Connection Manager. Selecting this item adds a new group to the Connection Manager. See Section 4.4, "Managing Groups".
New Terminal	Create a new terminal connection to an SSH server. Selecting this item opens the terminal properties dialog box. See Section 4.6, "Creating a new Terminal Connection" on page 112.
New TCP Tunnel	Create a new TCP tunnel. Selecting this item opens the tunneling properties dialog box. See Section 4.8, "Tunnels" on page 122.
Put Away	Remove the selected group, terminal or tunnel from the Connection Manager.
Connect	Connect the selected group, terminal or tunnel to the server defined in its Properties dialog box. This option is only available if the active group, terminal or tunnel has not yet been connected to the server.

Disconnect	Disconnect the selected group, terminal or tunnel from the server. This option is only available if the selected group, terminal, or tunnel is connected to the server.
Clone	Make an exact copy of the selected group, terminal, or tunnel in the Connection Manager.

The connection information screen shows all currently opened items and information about their connection status.

For groups, you are shown an outline triangle for expanding and collapsing the contents of the group. When the triangle is pointing to the right, the group is collapsed, hiding its contents. When the triangle points down, the group is expanded, displaying its contents.

After the group icon you see the name you have given to the group, or the address of the host you are connected to, if no name has been given. If no tunnels or terminals inside that group, or in any groups inside the group, are connected to a host, no further information is shown. If there are one or more connections, you will see the combined amount of data transferred in/out through all the connections within that group hierarchy, followed by the number of open connections within that group hierarchy.

The item's name is displayed next to the icons. If the item has not been named, the address of the host it is connected to is displayed. If the terminal or tunnel is connected, the amount of data transferred in and out through that connection will also be shown, along with the cipher used for encrypting the data.

The data in/out transfer indicator is especially useful when setting up tunnels. When a tunnel is first opened, the indicator will show 0/0. When the tunnel is being accessed, the indicator immediately changes. The indicator will remain at 0/0 if the tunnel is never accessed, which can happen if your Web browser or e-mail client is not correctly configured to use the tunnel.

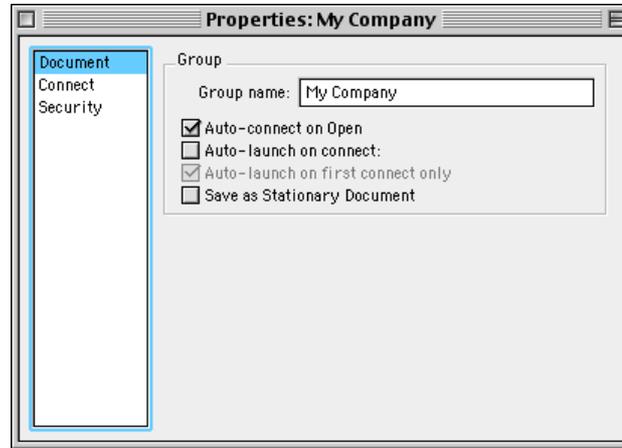
To select an item in the Connection Manager window, use the up and down arrow keys. To open a terminal screen, select the terminal and press ENTER or double-click on it. To open the terminal's Properties dialog box, press **⌘**-ENTER or press OPTION-ENTER. You can also open a terminal's Properties dialog box by clicking on the status box in the bottom left of the terminal window.

To open the Properties box for a tunnel or a group, select it and press ENTER, or double-click on it.

## 4.5 Group Properties

### Group Properties — Document

Properties Item	Description
Group Name	Give a name to the group. This name will be shown beside the icon of the group in the Connection Manager. The name can be used to quickly identify the group. If no name is given, the address of the host being connected to is shown instead.
Auto-connect on Open	This option applies to groups that have been saved as files that can be opened in SSH. When this option is selected, opening a file will automatically connect you to the server named in the Connect page of the Properties dialog box.
Auto-launch on connect:	When checked, the selected application will automatically be launched as soon as the connection has been established.
Auto-launch on first connect only	When checked, the application selected in the previous item will only be opened once as long as the group remains open in the Connection Manager. This will disable multiple instances of the application (for example when the group is cloned).
Save as Stationary Document	Save the group as a stationary document. This option is cleared when the document is opened again, so you will need to select this option again if you want to make changes to the stationary document.



## Group Properties — Connect

Connect Item	Description
Override Connection Parameters	Values given in the Connect Properties of the group will override all the Connect and Security Properties in the groups, tunnels, and terminals beneath it in the Connection Manager hierarchy.
SSH Server	Give the DNS or IP address of the server you are connecting to, or select it from the list of your favorite servers. See Section 4.10, "F-Secure SSH2 Preferences," on page 131 for information on setting your favorite servers.
Port	Enter the port number if the SSH2 server you are connecting to is listening to a port other than the standard port 22.
User name	Enter your user name on the server you are connecting to, or select it from your list of user names. See Section 4.10, "F-Secure SSH2 Preferences," on page 131 for information on setting the list of user names.

**Authentication**

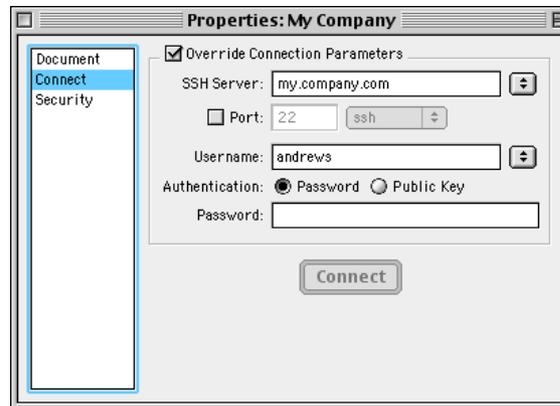
Select whether to use password authentication or public-key authentication. For information on how to setup your account for public-key authentication, see Section 4.11, “Public-key authentication,” on page 139.

**Password**

If you selected password authentication, enter your password here.

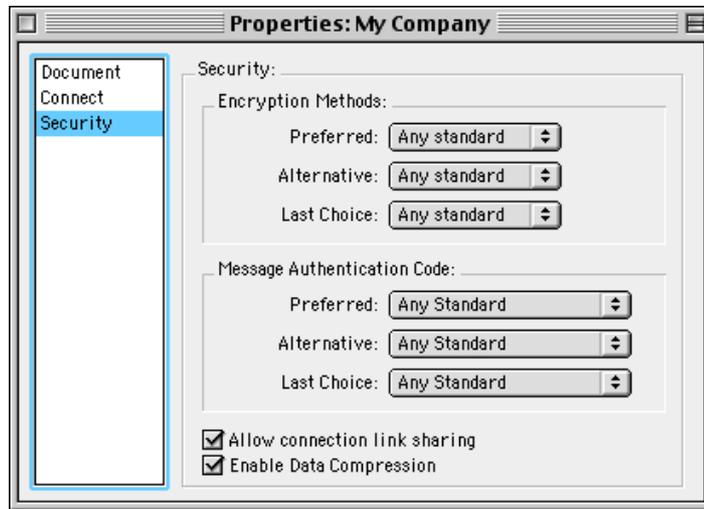
**Connect**

When you have given all the required information, click **Connect** to connect to the server.



## Terminal Properties — Security

Security Option	Description
Encryption Methods	<p>Select your preferred encryption methods from the pop-up menus. To force SSH to use a specific encryption method, select the same item from all three menus:</p> <ul style="list-style-type: none"> <li>• Preferred — Your preferred cipher</li> <li>• Alternative — Your second choice</li> <li>• Last Choice — Used if the above ciphers are not available</li> </ul>
Message Authentication Code	<p>Select your preferred methods for message authentication from the pop-up menus:</p> <ul style="list-style-type: none"> <li>• Preferred — Your preferred cipher</li> <li>• Alternative — Your second choice</li> <li>• Last Choice — Used if the above ciphers are not available</li> </ul>
Allow connection link sharing	<p>Allow terminals and tunnels connecting to the same host to use the same connection. When using connection link sharing, you need to authenticate yourself to the server only once for multiple connections that use the same authentication and security parameters. If the connection has hung for some reason, and you want to open a new connection, disable connection link sharing before opening the connection, so that a new TCP connection will be created.</p>
Enable Data Compression	<p>When this box is checked, all data traveling through any items in the group will be compressed to speed up the connection.</p>



## 4.6 Using Terminals

### Creating a new Terminal Connection

To create a new terminal connection in F-Secure SSH2, click the **New Group** button in the Connection Manager, press **⌘N**, or select New Terminal from the File menu. This will open the Properties dialog box for the new terminal connection.

The Properties dialog box will open on the Connect page. The simplest way to connect is to just enter all the required information on this page and click **Connect**. However, by going through all the pages, you can change several properties of the terminal connection.

All terminal properties are explained in Section 4.7, "Terminal Properties," on page 115.

### Opening an Existing Terminal Connection

You can open an existing connection in the following ways:

- Choose Open from the File menu, or press **⌘O**, and select the file you want to open.
- Choose the connection you want to open from the Shortcuts menu. For more information on using shortcuts, see Section 4.3, "Shortcuts Menu," on page 103.
- Double-click on the icon of the file you want to open.
- Drag the icon of the file you want to open onto the F-Secure SSH2 icon.

### Saving a Terminal Connection

After you have entered all the required information in the Terminal Properties dialog box, you can save the properties in a file by doing one of the following:

- Choose Save from the File menu, or press **⌘S**.
- Choose Save As from the File menu if you want to save an existing connection under a new name.

The file will be associated with SSH, so you can open it later by just double-clicking it.

## Selecting a Terminal Window as the Active Window

To select a terminal window that has been opened, do one of the following.

- Select it from the SSH menu, or press the **⌘** key and the number assigned to the terminal window.
- Click on its icon in the Connection Manager.
- Switch between the terminal windows by pressing the **⌘+** and **⌘-** keys.

## Closing a Terminal Window

To close a terminal window, select the terminal and do one of the following:

- Choose Close from the File menu, or press **⌘W**.
- Click the Close box in the upper left corner of the terminal window. If you select the “Put away When a Terminal Window is Closed” option in the Interface page of the Preferences dialog box, the terminal will be closed and removed from the Connection Manager.

## Connecting and Disconnecting from the Host

To connect to the host defined in the Properties dialog box, select the terminal window, and do one of the following.

- Choose Connect from the SSH menu, or press **⌘G**.
- Select the terminal and click the **Connect** button in the Connection Manager.
- Click on the word “Disconnected” in the status bar at the bottom of the terminal window.
- Open the terminal Properties dialog box, and click the **Connect** button.

To disconnect from a host, do one of the following.

- Choose Disconnect from the SSH menu, or press **⌘**.
- Log out from the host with the usual UNIX commands (logout, exit, or quit).
- Click the **Disconnect** button in the Connection Manager.

## Cloning the Terminal

To clone a terminal, select the terminal and do one of the following.

- Choose Clone from the File menu, or press **⌘D**.
- Click the **Clone** button in the Connection Manager.

If the original terminal is already connected to a server, cloning the terminal will automatically connect the cloned terminal to the server as well.

## Removing a Terminal from the Connection Manager

To disconnect a terminal from a host and remove the terminal from the Connection Manager, select the terminal and do one of the following.

- Choose Put Away from the File menu, or press **⌘Y**.
- Click the **Put Away** button in the Connection Manager.

## Working with Text in the Terminal Window

You can scroll the terminal and the scrollback buffer with the mouse. Pressing **⌘** and the up or down arrow at the same time will scroll the terminal up or down one screen at a time.

To select text in the terminal window, drag your mouse over the text. To select a single word, double-click on the word. To select a line of text, click three times anywhere in the line. To select a paragraph, click four times anywhere in the paragraph.

You can extend your previous selection by pressing **SHIFT** when selecting text. You can select URLs by pressing **OPTION** or **⌘** and clicking on the URL.

To copy text from the terminal window to the Clipboard, select the text you want to copy and either choose Copy from the Edit menu or press **⌘C**.

To copy text in a tabulated format, select the text you want to copy and choose Copy Table from the Edit menu. Text in tabulated format can be properly pasted into a spreadsheet.

To paste text into the terminal window, either choose Paste from the Edit menu or press **⌘V**.

## 4.7 Terminal Properties

To edit the terminal properties, do one of the following.

- Select the terminal and choose Show Properties from the SSH menu, or press **⌘P**.
- Double-click on the terminal in the Connection Manager.

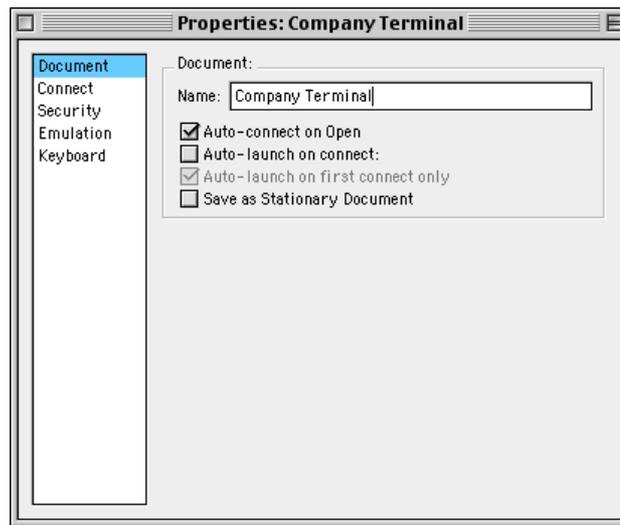
To toggle between the terminal and its properties, press **⌘P**.

### Terminal Properties — Document

Properties Item	Description
Name	Give a name to the connection. This name will be shown on the title bar of the terminal window and beside the icon of the terminal in the Connection Manager. The name can be used to quickly identify the connection.
Auto-connect on Open	This option applies to terminal connections that have been saved as files that can be opened in SSH. When this option is selected, opening a file will automatically connect you to the server.
Auto-launch on Connect	When checked, the selected application or document will automatically be launched as soon as the connection has been established. For example, you could launch a Web browser when the connection is established, or you can open a specific URL file.
Auto-launch on first connect only	When checked, the application selected in the previous item will only be opened once as long as the terminal remains open. This will disable multiple instances of the application (for example when the terminal window is cloned).

**Save as Stationary Document**

Save the terminal as a stationary document. When you open a terminal saved like this, you will be taken to the Properties page of the terminal just as if you were creating a new terminal, but the properties are by default as you saved them. This option is cleared when the document is opened again, so you will need to select this option again if you want to make changes to the stationary document.



## Terminal Properties — Connect

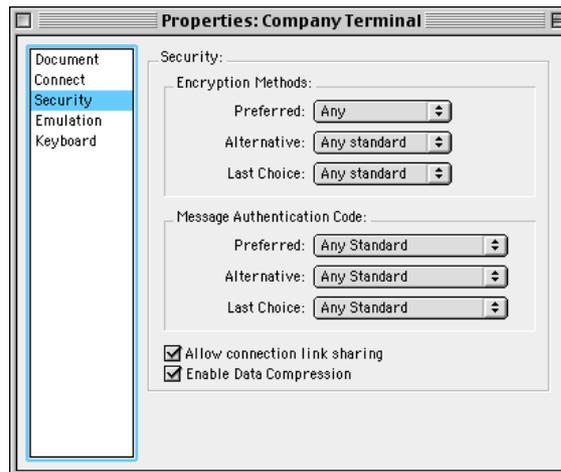
Connect Item	Description
SSH Server	Give the DNS or IP address of the server you are connecting to, or select it from the list of your favorite servers. See Section 4.10, “F-Secure SSH2 Preferences,” on page 131 for information on setting your favorite servers.
Port	Enter the port number if the SSH2 server you are connecting to is listening to a port other than the standard port 22.
User name	Enter your user name on the server you are connecting to, or select it from your list of user names. See Section 4.10, “F-Secure SSH2 Preferences,” on page 131 for information on setting the list of user names.
Authentication	Select whether to use password authentication or public-key authentication. For information on how to setup your account for public-key authentication, see Section 4.11, “Public-key authentication” on page 139.
Password	If you selected password authentication, enter your password here.
Connect button	When you have given all the required information, click <b>Connect</b> to connect to the server.



## Terminal Properties — Security

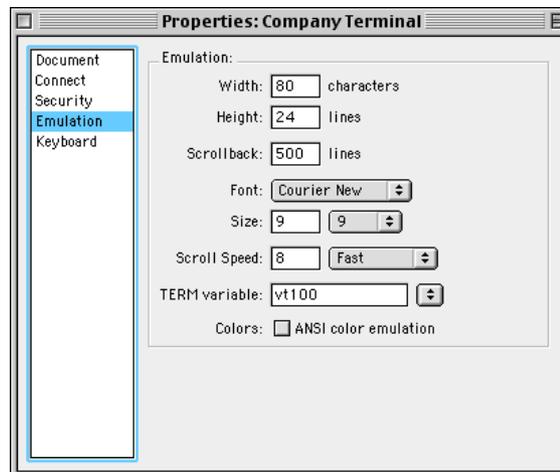
Security Option	Description
Encryption Methods	<p>Select your preferred encryption methods from the pop-up lists. To force SSH to use a specific encryption method, select the same item from all three menus:</p> <ul style="list-style-type: none"> <li>• Preferred — Your preferred cipher</li> <li>• Alternative — Your second choice</li> <li>• Last Choice — Used if the above ciphers are not available</li> </ul>
Message Authentication Code	<p>Select your preferred methods for message authentication from the pop-up menus:</p> <ul style="list-style-type: none"> <li>• Preferred — Your preferred cipher</li> <li>• Alternative — Your second choice</li> <li>• Last Choice — Used if the above ciphers are not available</li> </ul>

- Allow connection link sharing** Allow terminals and tunnels connecting to the same host to use the same connection. When using connection link sharing, you only need to authenticate yourself to the server once for multiple connections using the same authentication and security parameters. If the connection has hung for some reason, and you want to open a new one, disable connection link sharing before opening the connection, so that a new TCP connection will be created.
- Enable Data Compression** When this box is checked, all data traveling through the terminal will be compressed to speed up the connection.



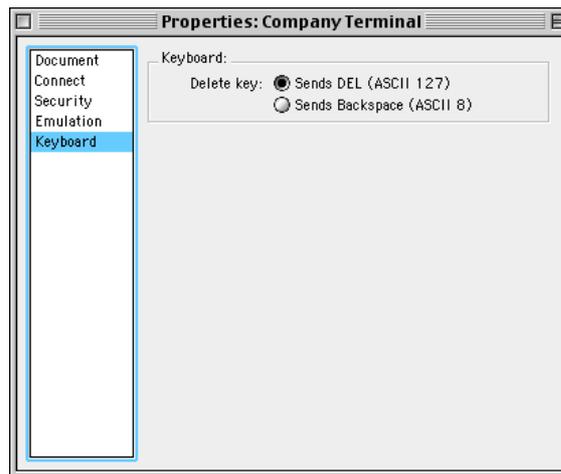
## Terminal Properties — Emulation

Emulation Option	Description
Width:	Enter the width of the terminal window in characters.
Height:	Enter the height of the terminal window in lines.
Scrollback:	Enter the number of lines to have in the scrollbar buffer. If memory is low, the F-Secure SSH Client may limit the number of scrollbar lines to conserve memory.
Font:	Select the font you want to use in the terminal from the pop-up menu.
Size:	Enter the size of the font, or select it from the pop-up menu.
Scroll Speed:	Enter the desired scrolling speed of the Terminal window, or select it from the pop-up menu. The speed is measured in lines per second.
TERM variable:	Select your preferred mode of terminal emulation from the pop-up menu.
Colors:	Checking the box selects ANSI color emulation.



## Terminal Properties — Keyboard

Keyboard Option	Description
Delete key:	Select whether pressing the DELETE key on the keyboard sends DEL or BACKSPACE to the terminal. By default, pressing SHIFT+DELETE sends BACKSPACE, and pressing DELETE by itself sends DEL. You can use this option to reverse this action.



## 4.8 Tunnels

### Creating a New Tunnel

To create a new tunnel in F-Secure SSH2, select New Tunnel from the File menu. This will open the Properties dialog box for the new tunnel.

The Properties dialog box will open on the Connect page. The simplest way to connect is to just enter all the required information on this page and click the **Connect** button. However, you can change several properties of the tunnel by going through all the pages.

### Opening an Existing Tunnel

To open an existing connection, do one of the following.

- Choose Open from the File menu (or press **⌘O**), and select the file you want to open.
- Double-click on the icon of the file you want to open.
- Drag the icon of the file you want to open onto the F-Secure SSH2 icon.

### Saving a Tunnel

After you have entered all the required information in the Tunnel Properties dialog box, save the properties in a file by one of the following methods.

- Choose Save from the File menu, or press **⌘S**.
- Choosing Save As from the File menu if you want to save an existing connection under a new name.

The file will be associated with SSH, so you can open it later by just double-clicking it.

### Selecting a Tunnel

To select a tunnel that has been opened, click on its icon in the Connection Manager.

## Connecting and Disconnecting from the Host

To connect to the host defined in the Properties dialog box, select the tunnel and do one of the following.

- Choose Connect from the SSH menu, or press **⌘G**.
- Select the tunnel and click the **Connect** button in the Connection Manager.
- Open the tunnel Properties dialog box and click the **Connect** button.

To disconnect from a host, do one of the following.

- Choose Disconnect from the SSH menu, or press **⌘**.
- Log out from the host with the usual UNIX commands (logout, exit, or quit).
- Click the **Disconnect** button in the Connection Manager.

## Cloning the Tunnel

To clone a tunnel, select the tunnel and do one of the following.

- Choose Clone from the File menu, or press **⌘D**.
- Click the **Clone** button in the Connection Manager.

## Removing a Tunnel from the Connection Manager

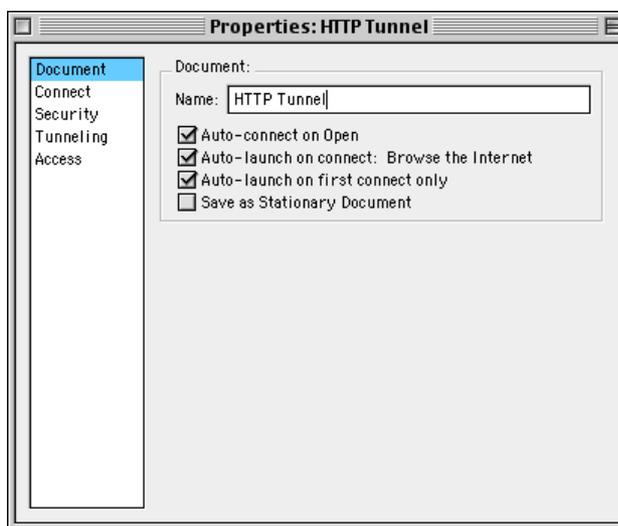
To disconnect a tunnel from a host and remove the tunnel from the Connection Manager, select it and do one of the following.

- Choose Put Away from the File menu, or press **⌘Y**.
- Click the **Put Away** button in the Connection Manager.

## 4.9 Tunneling Properties

### Tunneling Properties — Document

Document Option	Description
Name	Name of the connection. Displayed next to the icon for the tunnel in the Connection Manager.
Auto-connect on Open	This option applies to tunnels that have been saved as files that can be opened in SSH. When this option is selected, opening a file will automatically connect you to the server.
Auto-launch on connect	When checked, the application or document you select will automatically be launched as soon as the connection has been established. For example, you can open a URL file that uses the tunnel for transferring data.
Auto-launch on First Connect Only	When checked, the application selected in the previous item will only be opened once as long as the tunnel remains open. This will disable multiple instances of the application, for example when the tunnel window is cloned.
Save as Stationary Document	Save the tunnel as a stationary document. When you open a tunnel saved like this, you will be taken to the properties page of the tunnel just as if you were creating a new tunnel, but the properties are the same as when you saved it. This option is cleared when the document is opened again, so you will need to select this option again if you want to make changes to the stationary document.

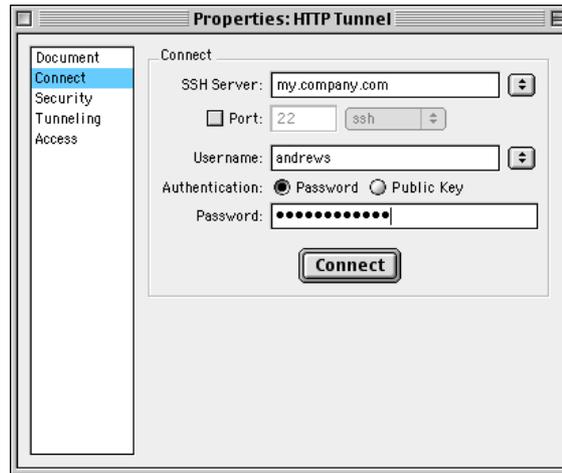


## Tunneling Properties — Connect

Connect Item	Description
SSH Server	Give the DNS or IP address of the server you are connecting the tunnel to, or select it from the list of your favorite servers. For information on setting your favorite servers, see Section 4.10, "F-Secure SSH2 Preferences," on page 131.
Port	If the SSH2 server you are connecting the tunnel to is listening to some other port than the standard 22, you can enter the port number here.
User name	Enter your user name on the server you are connecting the tunnel to, or select it from your list of user names. See Section 4.10, "F-Secure SSH2 Preferences," on page 131 for information on setting the list of user names.
Authentication	Select whether to use password authentication or public-key authentication. For information on how to set up your account for public-key authentication, see Section 4.11, "Public-key authentication," on page 139.

Password If you selected password authentication, enter your password here.

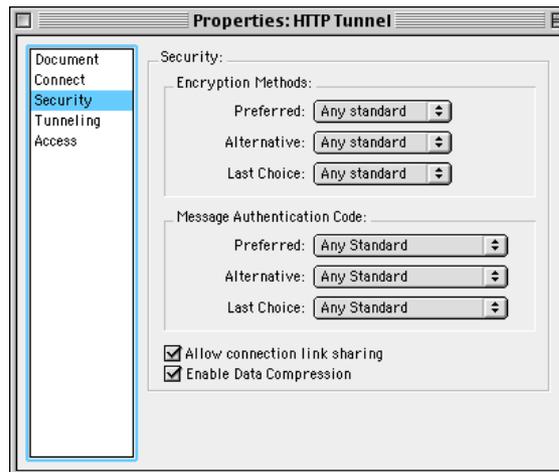
Connect When you have given all the required information, click **Connect** to connect the tunnel to the server.



## Tunneling Properties — Security

Security Option	Description
Encryption Methods	<p>Select your preferred encryption methods from the pop-up menus. To force SSH to use a specific encryption method, select the same item from all three menus:</p> <ul style="list-style-type: none"> <li>• Preferred — Your preferred cipher</li> <li>• Alternative — Your second choice</li> <li>• Last Choice — Used if the above ciphers are not available</li> </ul>
Message Authentication Code	<p>Select your preferred methods for message authentication from the pop-up menus:</p> <ul style="list-style-type: none"> <li>• Preferred — Your preferred cipher</li> <li>• Alternative — Your second choice</li> <li>• Last Choice — Used if the above ciphers are not available</li> </ul>

- Allow connection link sharing** Allow terminals and tunnels connecting to the same host to use the same connection. When using connection link sharing, you only need to authenticate yourself to the server once for multiple connections using the same authentication and security parameters. If the connection has hung for some reason, and you want to open a new one, disable connection link sharing before opening the connection, so that a new TCP connection will be created.
- Enable Data Compression** When this box is checked, all data traveling through the tunnel will be compressed to speed up the connection.

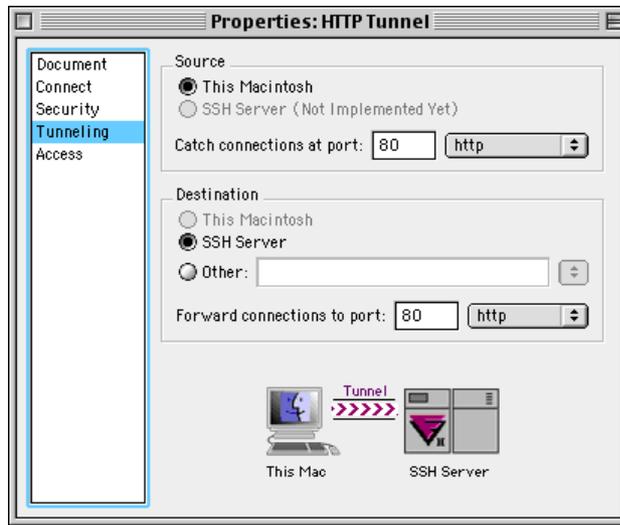


## Tunneling Properties - Tunneling

In the current version, only local tunneling is possible. You cannot change the source or the destination of the tunnel.

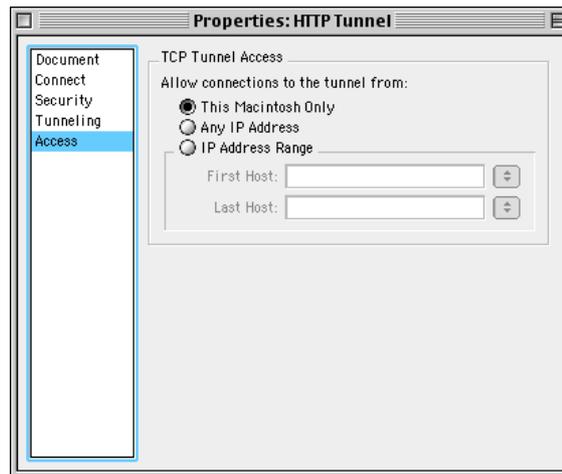
For more information on tunneling, see Section 1.5, "Tunneling," on page 7.

Tunneling Option	Description
Source	Select the source of the tunnel. In this version, the source of the tunnel can only be 127.0.0.1. 'localhost' may also work, depending on the DNS server being used.
Catch connections at port:	Select which port F-Secure SSH2 will listen to on your Macintosh. You can enter the port number manually or you can select one of the predefined ports from the pop-up menu.
Destination	Select the destination of the tunnel. As remote tunneling is not available in this version, the Macintosh you are connecting the tunnel from cannot be used as a destination. You can select the SSH2 server to which you are connecting as the destination server of the tunnel, or you can select a different destination.
Forward connections to port	Select which port on the selected destination host the tunnel is connected to. You can enter the port number manually or select one of the predefined ports from the pop-up menu.



## Tunneling Properties — Access

Access Option	Description
Allow connections to the tunnel from:	
This Macintosh only	Do not allow connections to the tunnel from anywhere else but the Macintosh you are creating the tunnel from.
Any IP Address	Allow anyone to use the tunnel you create from any computer in the world that can access the source of the tunnel.
IP Address Range	Give a range of IP addresses or DNS names from which the tunnel can be accessed (for example 10.10.10.1 and 10.10.10.255). You can also define a single IP address or DNS name to allow connections only from that address.

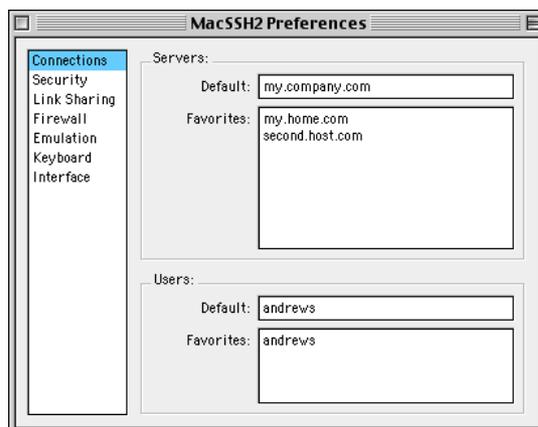


## 4.10 F-Secure SSH2 Preferences

You can edit the general preferences of F-Secure SSH2 for MAC by selecting Preferences from the SSH menu. The Preferences dialog box and Properties dialog box share many of the same options. The preferences set in the Preferences dialog box are used as defaults for all your connections.

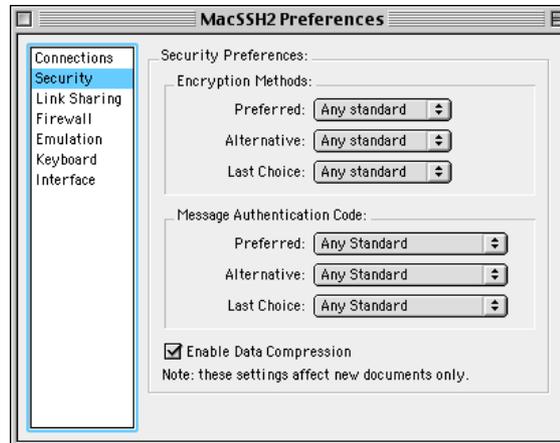
### Document Preferences

Connections Option	Description
Servers: Default:	Enter your default server address. This will then be used as the default address in all new terminals and tunnels that you create.
Servers: Favorites:	List your other favorite server addresses. You can then select them from a pop-up menu in the Properties dialog box of any new terminal or tunnel.
Users: Default:	Enter your default user name. This will be used as the default user name in all new terminals and tunnels that you create.
Users: Favorites:	List the user names you use often. You can then select them from a pop-up menu in the Properties dialog box of any new terminal or tunnel.



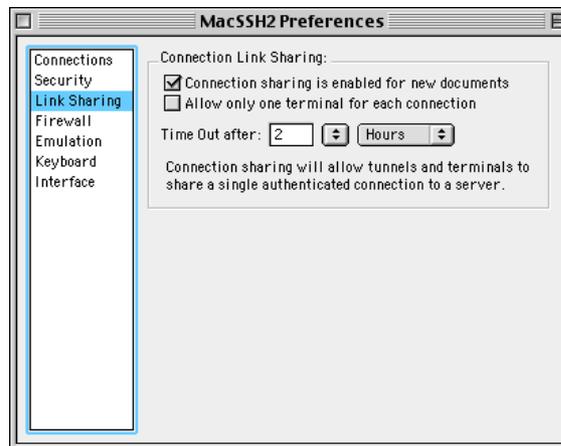
## Security Preferences

Security Option	Description
Encryption Methods	<p>Select your preferred encryption methods from the pop-up menus:</p> <ul style="list-style-type: none"> <li>• Preferred — your first choice</li> <li>• Alternative — your second choice</li> <li>• Last Choice — used if your first two choices are not available</li> </ul>
Message Authentication Code	<p>Select your preferred methods for message authentication from the pop-up menus:</p> <ul style="list-style-type: none"> <li>• Preferred — your first choice</li> <li>• Alternative — your second choice</li> <li>• Last Choice — used if your first two choices are not available</li> </ul>
Enable Data Compression	<p>When this box is checked, all data traveling through all your connections will be compressed to speed up the connection.</p>



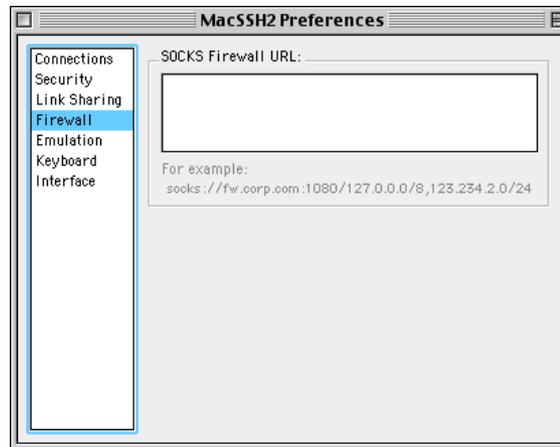
## Link Sharing Preferences

Link Sharing Option	Description
Connection sharing is enabled for new documents	When checked, new terminals and tunnels are allowed to use an existing connection for transferring data, instead of making a new connection.
Allow only one terminal for each connection	When checked, only one terminal may use one connection, that is, each terminal will have its own connection. However, the number of tunnels for one connection is not limited.
Time Out after:	Sets the amount of time after which new connections cannot automatically share the existing connection, but will have to be authenticated again. It is especially useful to set the time to a few minutes if you keep a few tunnels open for a long time, and you want to make sure that other people accessing your computer will not be able to use the tunnels to open an authenticated connection to your UNIX account.



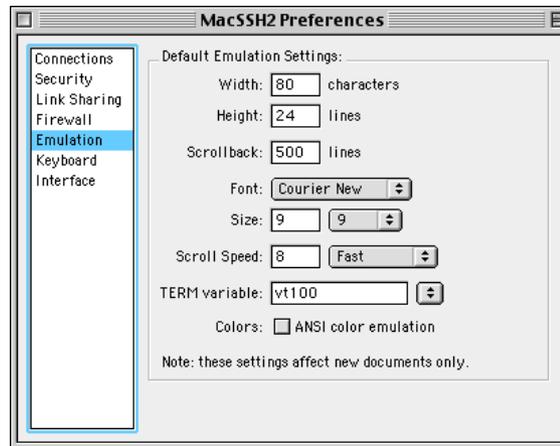
## Firewall Preferences

Firewall Option	Description
Socks Firewall URL:	If you are connecting from behind a SOCKS firewall, you can enter the URL and the port number of the firewall here.



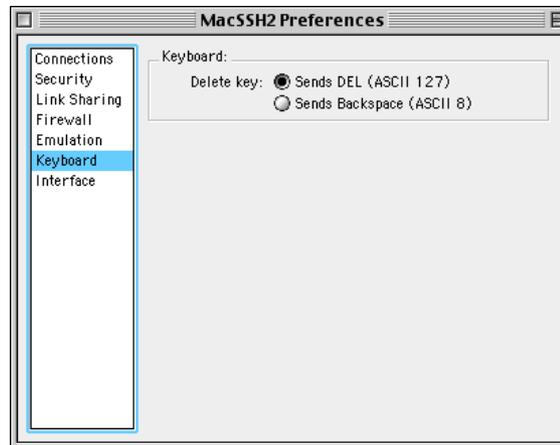
## Emulation Preferences

Emulation Option	Description
Width:	Enter the default width of terminal windows in characters.
Height:	Enter the default height of terminal windows in lines.
Scrollback:	Enter the number of lines you want to keep in the scrollback buffer. If memory is low, F-Secure SSH Client may limit the number of scrollback lines to conserve memory.
Font:	Select the font you want to use in terminal windows from the pop-up menu.
Size:	Enter the size of the font, or select it from the pop-up menu.
Scroll Speed:	Enter the default scrolling speed of terminal windows, or select it from the pop-up menu. The speed is measured in lines per second.
TERM variable:	Select your preferred mode of terminal emulation from the pop-up menu. This value changes only during each logon. If the value is changed while connected, the changes will take place the next time the connection is opened.
Colors:	Checking the box selects ANSI color emulation.



## Keyboard Preferences

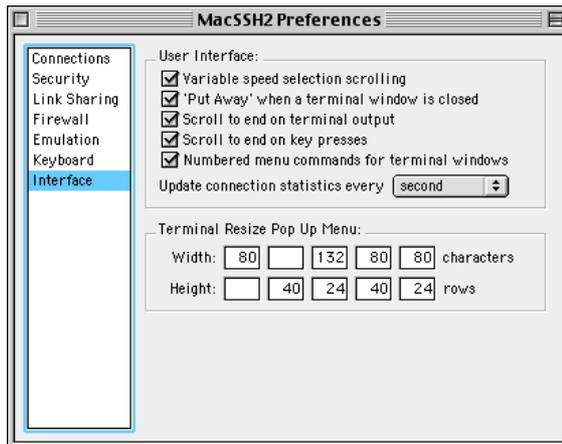
Keyboard Option	Description
Delete key:	Select whether pressing the DEL key on the keyboard sends DEL or BACKSPACE to the terminal.



## Interface Preferences

Interface Option	Description
Variable speed selection scrolling	The speed with which the screen scrolls when you are selecting text with the mouse will depend on the distance of the mouse pointer from the terminal window. If this option is not selected, the screen will always scroll at the same speed.
'Put Away' when a terminal window is closed	A terminal window that is closed is also removed from the Connection Manager. If this option is not selected, closing a terminal window does not remove it from the Connection Manager or disconnect the terminal connection from the server.
Scroll to end on terminal output	If you have scrolled the terminal screen back and this box is checked, as soon as you get new output on the terminal window, the window is scrolled back to the end.
Scroll to end on key presses	If you have scrolled the terminal screen back and this box is checked, as soon as you press a key, the window is scrolled back to the end.
Numbered menu commands for terminal windows	When checked, the terminal windows will be numbered on the list in the SSH menu. They can then be activated by pressing <b>⌘1</b> , <b>⌘2</b> , etc.
Update Connection Statistics Every	Select from the pop-up menu how often the connection statistics will be updated in the Connection Manager.
Terminal Resize Pop Up Menu	

Define up to five different terminal size settings that can be directly changed from the pop-up menu in the status bar of the Connection Manager. If a box is left empty, the current terminal setting is used for it in the pop-up menu.



## 4.11 Public-key authentication

F-Secure SSH Client for Macintosh creates a new random seed every time it is started. It then uses user actions to add true random elements to the pseudo random number generator. When generating new key pairs, a large amount of randomness is desirable. Therefore, you should use SSH for a while, or launch it and let it run in the background while you are working in some other program, before generating a new key pair. For normal tunneling or terminal use, a few seconds of activity will create sufficient randomness for secure connections.

To start using public-key authentication, you have to create a key-pair. Follow these steps:

1. Select Create Public Key Files from the File menu. The Public-key authentication Key Generator will open.
2. Enter a comment in the Comment field. This only identifies the key.
3. Select the Key Type (DSA or RSA) and the key length. The recommended key length is at least 1024 bits. Key lengths shorter than this are a security risk. Key lengths longer than 2048 bits do not provide a significant increase in security.
4. Enter a Passphrase used to encrypt the private key. For good security, the passphrase should be at least 11 characters long and include both lowercase and uppercase letters, numbers, and special characters. If the private key is secured by other means, the passphrase does not have to be this long.
5. Select one of the following options in the Clipboard pop-up menu:
  - 'Copy install script' copies a script to the Clipboard for transferring the public key to the server.
  - 'Copy public key' copies only the contents of the public key to the Clipboard.
  - 'Leave alone' will not copy anything to the Clipboard.
6. Click on the **Create New Key Pair** button.



7. You will be prompted for the name of the private key file and the public key file, and the location for saving them. Do not give a file extension to the private key file. By default, the public key is named by adding the extension *.pub* to the private key file name.

## Transferring the Public Key

There are two ways to transfer the public key to the server.

- If you used the 'Copy install script' option when creating the key pair, you can connect to the server using password authentication. Then, immediately after connecting, you can paste the contents of the Clipboard to the terminal. After this you should be able to connect using public-key authentication.
  - ☞ **Note:** The script disables any keys that you are already using on the server. The script creates backup files of the old '*authentication*' and '*identity.pub*' files. Old backup files will be overwritten if the script is used again, so use the script only if you do not want to use the old keys anymore. If you want to preserve old keys, use the following method.

If you used the 'Copy public key' option when creating the key pair, or if you want to transfer an existing key to the server, follow these steps:

1. Copy the contents of the public key to the Clipboard. You do not need to do this if you have just created a key-pair using the 'Copy public key' option.
2. Connect to the SSH2 server using password authentication.
3. If you do not have a directory `$HOME/.ssh2`, create it with the following command:  

```
mkdir $HOME/.ssh2
```
4. Change to the `$HOME/.ssh2` directory:  

```
cd $HOME/.ssh2
```
5. Open or create a file called *authorization* in your favorite UNIX text editor. Add the following line:  

```
Key filename
```

where `filename` is any name you want to give to the public key on the server. It does not need to be the same as on your Macintosh. Save the *authorization* file and exit.
6. Create a new file for the public key and open it. Paste the public key from the Clipboard to the file, save the file and close it.
7. Change the permissions of the two files with the following commands:  

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

where `filename` is the name that you gave to the public key file on the server.

Disconnect from the server and connect again, now using public-key authentication to verify that the key has been installed correctly.



## 5. F-Secure SSH 1 for Windows

### 5.1 Compatibility

F-Secure SSH Client 1.1 for Windows is not compatible with the new and improved SSH2 protocol. If you want to connect to an SSH2 server, you need to use F-Secure SSH 3.0 Client for Windows.

Also, if you are trying to connect to an SSH1 server, you will have to use F-Secure SSH Client 1.1 for Windows, as F-Secure Client 3.0 does not support the old SSH1 protocol.

### 5.2 Installing F-Secure SSH

To install the F-Secure SSH Client, follow these steps:

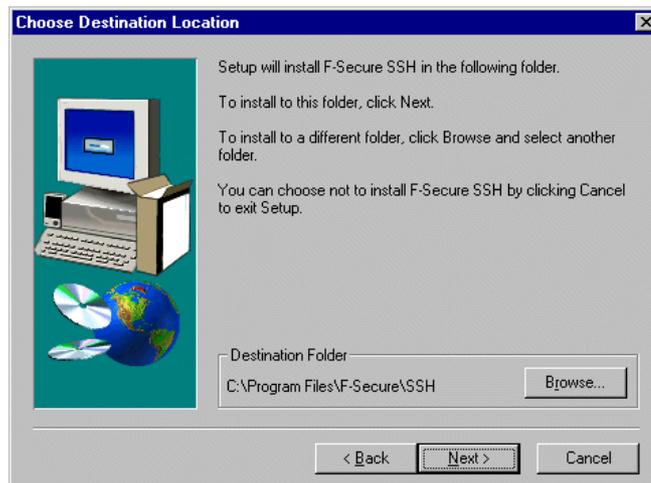
1. Insert the F-Secure CD-ROM into the CD-ROM drive. This should start the autorun feature. If nothing happens, go to the root directory of the CD-ROM and double-click on *install.exe*. This will start the F-Secure installer program.
2. Select your preferred language from the 'Language' list, and select 'F-Secure SSH Client 1.1 for Windows 95/98/NT' from the 'Product or document' list.



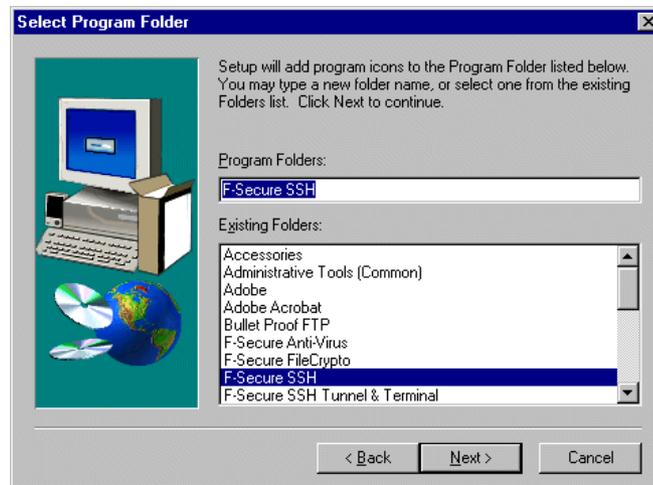
3. Read the Welcome screen, and click Next.



4. In the Choose Destination Location dialog box, choose the location to install the F-Secure SSH Client. Then click **Next**.



- In the Select Program Folder dialog box, specify the location in the Windows Start menu where you want shortcuts for the F-Secure SSH Client software. Click **Next**.



- In the Select Components dialog box, select Personal Program Group if you want to install F-Secure SSH Client for personal use only. Select Common Program Group to allow anyone using the computer to use the same installation. Click **Next** to install the files.



- When the setup is complete, you can choose to launch F-Secure SSH or to view the Readme file, which contains the latest information on the product. Then click **Finish**.



## 5.3 F-Secure SSH Wizard

### Creating an RSA Identity

The SSH protocol implements RSA authentication as the strongest means for user authentication. To use RSA authentication, you must create an RSA key pair with the help of the F-Secure SSH Key Generation Wizard.

To use the F-Secure SSH Key Generation Wizard, click the Key Generation Wizard icon in the F-Secure SSH Client program group. The Wizard is started automatically during setup.

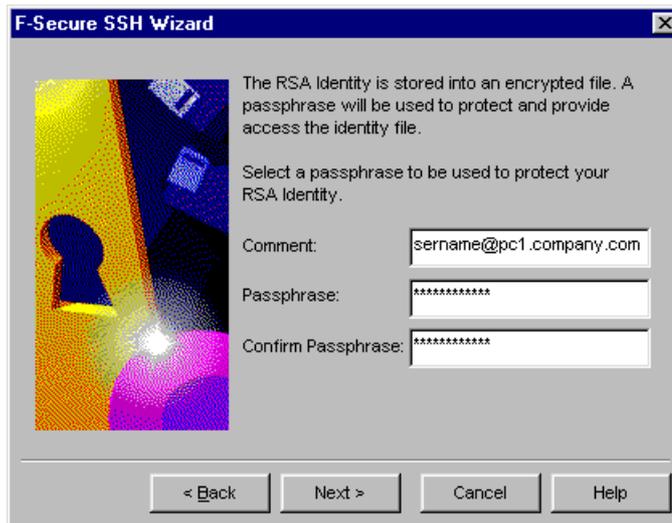
Do the following procedure to run the F-Secure SSH Key Generation Wizard and generate a key pair.



Read the information on the first Wizard page. Then click **Next**.



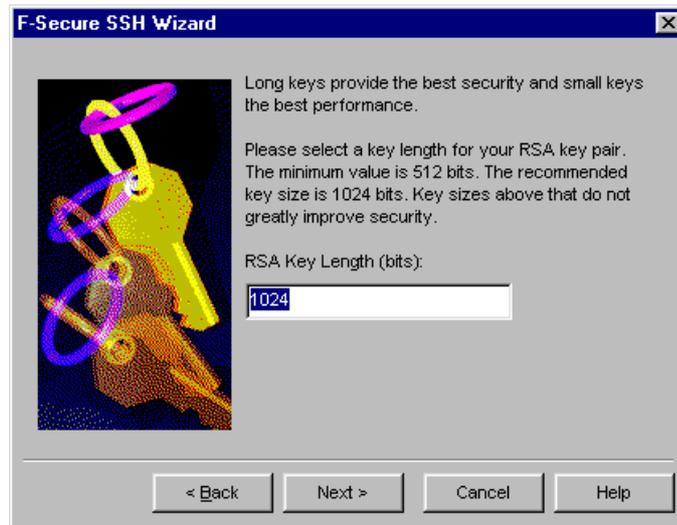
Enter the file name to use for your identity files. No file extension should be given. The private key has no file extension, and the public key has a *.pub* extension. We recommend using the default file name 'IDENTITY'. Click **Next**.



Enter a comment for the RSA Identity file. The comment is attached to the public key file. The comment is for your own information; it is not used by F-Secure SSH. You could type in your user name and the host name storing the identity files (for example, "bob@pc1.company.com").

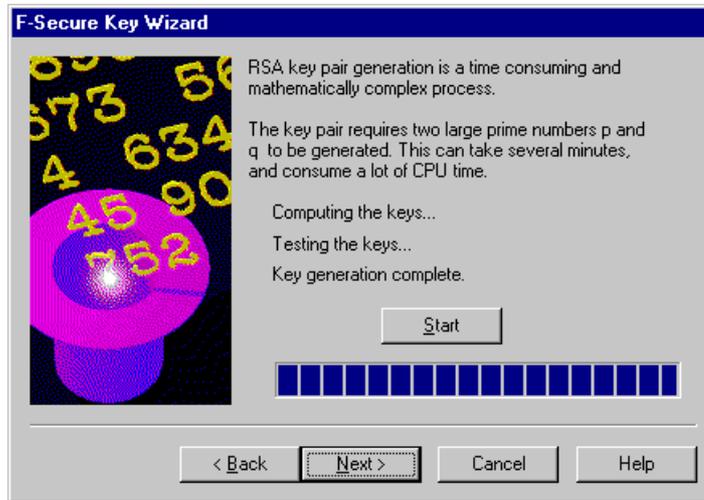
A passphrase will be used to protect your RSA Identity file. Enter your passphrase into the Passphrase field and Confirm Passphrase field. The fields are case-sensitive. For good security, the passphrase should be at least 11 characters long, and should include both upper- and lower-case letters, numerals, and special characters.

Click the **Next** button.



Specify the number of bits you want for the key length for the RSA key pair. We recommend 1024 bits. Larger key sizes do not greatly improve security. Increasing the key length increases security, but long keys slow down the RSA authentication process. The minimum key length is 512 bits. Click **Next**.

If you have created a random seed by moving your mouse cursor over the Random Number Generation dialog box, go directly to the next step. If you have not created a random seed, do so by moving your mouse randomly in the dialog box. A random seed will be created for F-Secure SSH's cryptographically strong random number generator. Click **Next** to continue.



The F-Secure SSH Key Generation Wizard generates two prime numbers. It then calculates and tests an RSA key pair to be used as the Identity. This process can take from one to fifteen minutes. Do not reboot your machine, even if it does not respond.

Click **Start**. Wait for the key generation to finish. Click **Next**.



The F-Secure SSH Key Generation Wizard is now ready to save the identity files. Click **Finish** to save the RSA Identity and quit the Wizard.

## 5.4 Basic Features

F-Secure SSH provides you with a secure, encrypted, and authenticated channel for connecting to remote UNIX hosts. The software is intended as a replacement for insecure terminal applications such as TELNET and RLOGIN.

Furthermore, X11 connections and arbitrary TCP/IP connections can be secured with the port-forwarding feature of F-Secure SSH.

### Connecting to a Remote Host

To start F-Secure SSH, double-click the F-Secure SSH icon in the F-Secure SSH program group.

To connect to a remote computer using password authentication, do the following steps:

1. From the Edit menu, choose Properties.
2. In the Connection dialog box, type the name of the remote host you want to connect to, and type the user name for your account.
3. From the Options group, select Password as your authentication type.
4. Click the **OK** button.
5. Press ENTER, or choose Connect from the File menu.
6. Type in the password for your account in the Connect Using Password Authentication dialog box.
7. Click the **OK** button to connect.



If this is the first time you are connecting to the host, SSH may prompt you to accept the host's previously unknown host key.



- ☞ The easiest and quickest way to make a connection is to press ENTER once in the empty terminal window. This causes F-Secure SSH to display the Connection dialog box.

## Creating Connection Templates

Connection templates allow you to save different sets of settings for your connections. In addition, they allow you to open multiple connections with different host names, user names, and passwords without repeatedly going to the Properties dialog box. You can create a connection template from an existing open connection or from a previously saved connection template.

To create a connection template, do the following steps:

1. From the Edit menu, choose Properties.
  2. Make the changes you want. You can edit the Connection, RSA Identity, Forward, Font, Terminal, and Keyboard properties.
  3. Click the **OK** button on the Property pages to activate the changes.
  4. From the File menu, choose Save. All changes are saved to a new connection template or to the connection template that you were editing.
- ☞ The saved connection templates are associated with the F-Secure SSH program. You can create icons on the Windows desktop for the connection template files (.SSH files) and double-click the icons to launch the saved connection.

## Working with Text in the Terminal Window

### Copying Text to the Clipboard

To copy text to the Clipboard, do the following:

1. Select the text by dragging the mouse over it.
2. From the Edit menu, choose Copy.

Once a selection is copied to the Clipboard, it can be inserted into SSH or another Windows application with the Paste command.

 **You can right-click the mouse to open a shortcut menu which contains most of the Edit commands.**

### Selecting All Lines on the Screen

From the Edit menu, choose Select Screen.

The Select Screen command selects all lines currently visible in the terminal window.

### Selecting All Lines in the Scrollback Buffer

From the Edit menu, choose Select All.

The Select All command selects the entire contents of the scrollback buffer.

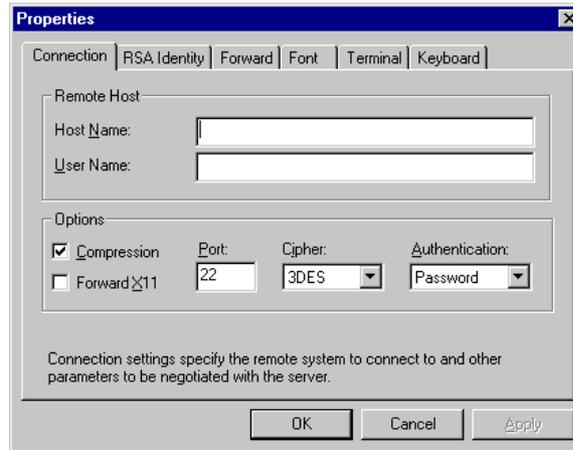
## Setting Connection Properties

The Properties command (Edit menu) modifies settings that control the terminal display, connection information, keyboard settings, secure TCP/IP connections, and other options.

Property Sheet	Description
Connection	Specifies information about the host, user, authentication type, compression, support for X11 Windowing System, port, and cipher used when connecting to a remote system.
RSA Identity	Specifies information about your RSA identity. Here you can select the location of the identity files, change the passphrase used to protect your RSA identity, and generate a new RSA key pair.
Forward	Specifies the local and remote TCP/IP connections to be secured by forwarding them through the F-Secure SSH connection. Identifies the names and connection parameters for the forwarded connections.
Font	Specifies the font and the colors for text, background, and inactivity.
Terminal	Specifies the number of lines stored in the scrollback buffer, and specifies settings that control the behavior of the terminal window.
Keyboard	Specifies the keyboard map file used to translate keystrokes and received terminal data into characters displayed in the terminal window. The Keyboard Property Sheet allows you to customize keyboard behavior.

## Connection Property Sheet

Connection tab of Properties command (Edit menu). Specifies options for a remote connection.



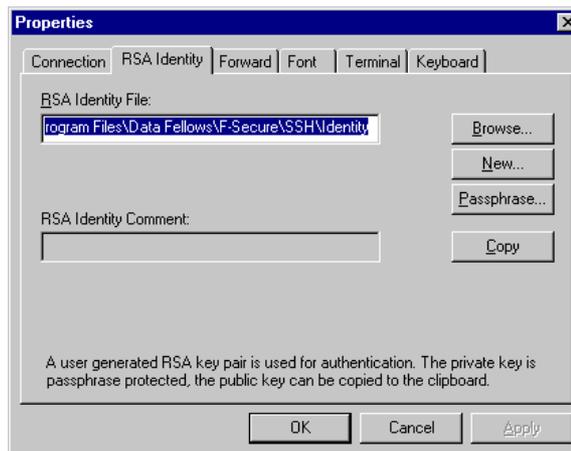
### Remote System Option Function

Host Name	Specifies the host name or the IP address of the remote machine to log in to.
User Name	Specifies the user name for logging into the remote machine.
Other Options	Function
Compression	Requests compression of all data, including data for forwarded X11 and TCP/IP connections. The compression algorithm is the same as used by the gzip program.
Forward X11	Instructs F-Secure SSH to create a proxy X server on the server machine for forwarding the X11 connections over the encrypted channel.
Port	Connection port on the remote host.
Cipher	Selects the cipher to use for encrypting the session. 3DES is the default.

**Authentication**                      Selects the authentication method for the session. RSA authentication is available only if you have specified an identity file to be used on the RSA Identity property page. Password is the default.

## RSA Identity Property Sheet

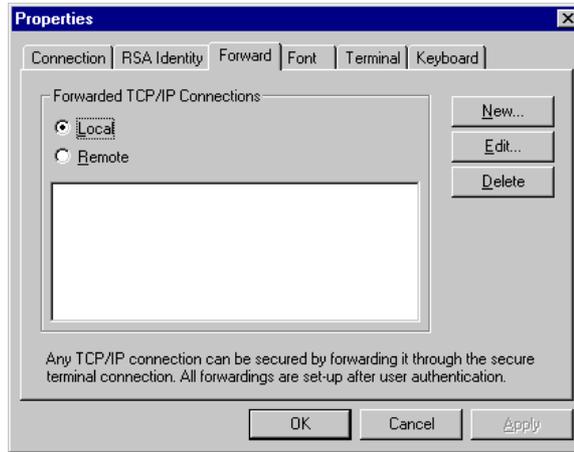
RSA Identity tab of Properties command (Edit menu). Specifies information about your RSA identity.



Option	Function
RSA Identity File	Selects the file from which the identity (private key) for RSA authentication is read. The default is the identity in the installation directory. Different identity files may be specified for each connection.
Comment	An optional comment in the <i>identity.pub</i> file. Comments typically specify the user and the host that created the RSA Identity file.
Button	Function
Browse	Displays a dialog box for browsing and selecting the RSA Identity file from locations other than the default location.
Copy	Copies the RSA public key and the comment to the Clipboard. You can then paste the public key to the <i>authorized_keys</i> file on the server.
New	Runs the Key Generation Wizard to create a new RSA key pair for your RSA Identity.
Passphrase	In order to prevent unauthorized people from using your RSA Identity and gaining access to your connections, you can use a passphrase to protect the identity file. You should change your passphrase periodically.

## Forward Property Sheet

The Forward tab of the Properties dialog box specifies options for the local and remote TCP/IP connections to be secured by F-Secure SSH.



Option	Function
Local Forwardings	Displays the local TCP/IP ports that have been forwarded. The <b>New</b> , <b>Edit</b> , and <b>Delete</b> buttons will then apply to local forwardings.
Remote Forwardings	Displays the local TCP/IP ports that have been forwarded. The <b>New</b> , <b>Edit</b> , and <b>Delete</b> buttons will then apply to remote forwardings.
Button	Function
New	Define a new TCP/IP connection to be secured. Specify source port, destination host, and port parameters.
Edit	Edit a previously defined forwarding. Specify source port, destination host, and port parameters.
Delete	Delete the selected forwarding.

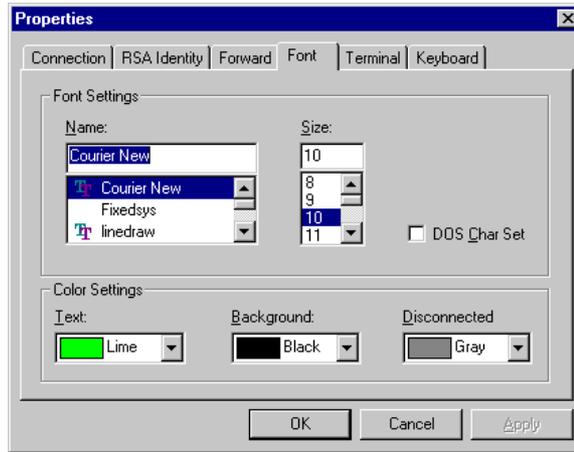
Click the **New** or **Edit** button to display the Forward a TCP/IP Connection dialog box.

The Forward a TCP/IP Connection dialog box specifies the local and remote ports for the connection, the remote system to forward the connection to, and a descriptive name for the forwarded TCP/IP connection.

Local or Remote Option	Function
Name	Each forwarding can be given a descriptive name that is displayed in the Forwardings box.
Source Port Number	Source port is the port that the clients connect to.
Destination host	Specify the host that runs the target TCP/IP service. The destination host can be specified as 127.0.0.1 (that is, <i>localhost</i> ), if the service is running on the same remote system that is used by the user terminal session.
Destination Port Number	Destination port is the port used by the TCP/IP service on the destination host.
Allow Local Connections Only	Protects your forwarded connections against use by any computer other than your own computer. This option disallows any connection other than those from the localhost address 127.0.0.1.
Application to start	Defines the application to be started when SSH Client is run by double-clicking on a file with an <i>.ssh</i> extension. Unless this box is empty, the user is prompted for authentication, the forwardings are set up, SSH Client is minimized, and the application is started automatically. If the box is empty, SSH Client is not minimized after user authentication.

## Font Property Sheet

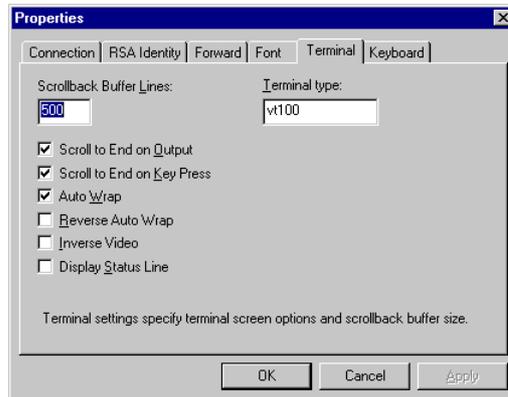
Font tab of Properties command (Edit menu). Specifies options for terminal window font and colors.



Font Option	Function
Name	Type or select a font name. The Font property page lists currently available fonts for the terminal window.
Size	Type or select a font size.
Color Option	Function
Font	Type or select one of the 16 predefined colors to be used with the characters displayed on the screen.
Background	Type or select one of the 16 predefined colors to be used as the background color.
Disconnected	Type or select one of the 16 predefined colors to be used to indicate that a connection has been terminated.

## Terminal Property Sheet

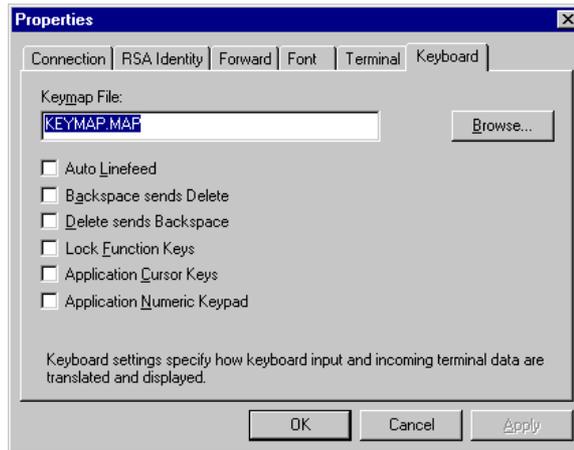
Terminal tab of Properties command (Edit menu). Specifies options for the terminal window.



Option	Function
Scrollback Buffer Lines	The number of lines to be allocated for the scrollback buffer.
Scroll to End on Output	Instructs F-Secure SSH to scroll to the end of the scrollback buffer when data is received from the connection.
Scroll to End on Key Press	Instructs F-Secure SSH to scroll to the end of the scrollback buffer when you press a key.
Auto Wrap	Allows auto-wraparound. The cursor will automatically wrap to the beginning of the next line when it reaches the end of the line.
Reverse Auto Wrap	Allows reverse wrap-around. The cursor will move from the first column of a line to the last column of the previous line.
Inverse Video	Displays the terminal window characters with the background color. Displays the background with the font color.
Status Line	Displays a VT100 status line on the terminal window.

## Keyboard Property Sheet

Keyboard tab of Properties command (Edit menu). Specifies options used to translate key presses and received terminal data to characters displayed in the terminal window.



Option	Function
Keyboard Map File	Selects the file from which the keymap for keyboard remapping is read. Default is <i>keymap.map</i> in the installation directory. Different keymap files may also be specified on a per-connection basis.
Auto Linefeed	Toggles automatic insertion of linefeeds after each carriage return.
Backspace Sends Delete	Causes the backspace key to send the delete key code.
Delete Sends Backspace	Causes the delete key to send the backspace key code.
Lock Function Keys	Locks all VT100 function keys (F1 through F20) in order to prevent them from being programmed by VT100 escape codes.

Option	Function
Application Cursor Keys	Toggles between the application cursor key mode and the VT100 cursor key mode.
Application Numeric Keypad	Toggles between application numeric keypad mode and VT100 numeric keypad mode.
Button	Function
Browse	Displays a dialog box to browse and select the keymap file from locations other than the default location.

## Disconnecting

You can save the connection settings in a connection template file for later use. From the File menu, choose *Save As* to save the connection settings.

To disconnect your connection, do the following:

1. If you have a terminal session active in the terminal window, enter the command to logout from the remote system. This command is typically called *logout*, *exit*, or *quit*.
2. Close any applications that are using the forwarded TCP/IP connections.
3. From the File menu, choose *Disconnect*.

If you did not close all forwarded TCP/IP connections, these forwardings will be listed on the terminal screen. The F-Secure SSH Client will wait for the forwardings to close after closing the terminal connection.

## Setting Up Your Account for RSA Authentication

SSH protocol implements RSA authentication automatically. You create your personal RSA key pair by running the F-Secure SSH Key Generation Wizard during setup. The RSA identity files are saved to the installation directory. The private key is saved in the file named *identity*. The public key is saved in a file named *identity.pub*. You can then add the contents of the *identity.pub* file to the *authorized\_keys* file on the remote host (normally, *\$HOME/.ssh/authorized\_keys*).

To add an RSA public key to the *authorized\_keys* file, do the following steps:

1. Connect to your account using password authentication.
  2. Choose Edit > Properties > RSA Identity.
  3. If the Comment field is empty, browse for the RSA identity file, or create a new RSA identity by clicking the **New** button. Repeat step #2. If the RSA Identity Comment field is not empty, go to step #4.
  4. Click the **Copy** button to copy the RSA public key to the clipboard.
  5. Change to the *.ssh* directory in your home directory on the remote system.
  6. Load the *authorized\_keys* file to a text editor. If the file does not exist, create it with a text editor.
  7. Paste the RSA public key from the clipboard to the end of the *authorized\_keys* file. Make sure your text editor has automatic word wrapping, because the key must not be wrapped over several lines. The file has exactly one line per public key entry.
  8. Save the *authorized\_keys* file, and exit the text editor.
- ☞ **The account is now ready to be accessed using RSA authentication from a client machine that contains the correct RSA identity file (the private and the public key pair).**

## Connecting Using RSA Authentication

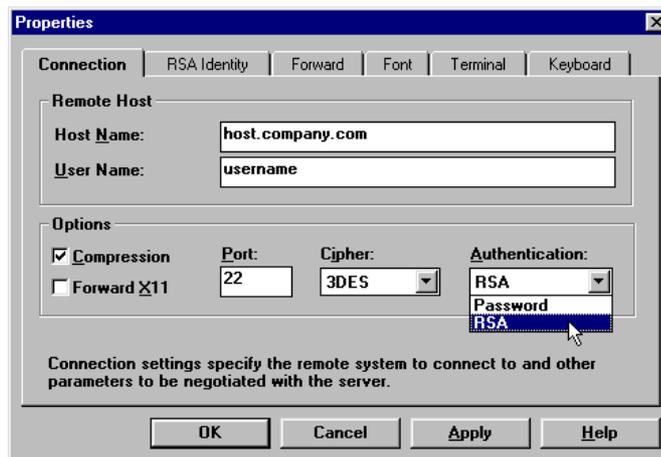
RSA authentication is based on public-key cryptography. RSA is a crypto system that uses an encryption key and a separate decryption key that cannot be derived from the encryption key.

During the installation of F-Secure SSH, a public and private key pair is generated for your RSA identity user authentication. The server knows the public part of your RSA identity, but only you know the private key.

When you log in, the program tells the server which key pair to use for authentication. If the server verifies the key, it sends the F-Secure SSH Client a challenge in the form of a random number encrypted by your public key. The challenge can only be decrypted with the correct private key.

To respond to the challenge, the client prompts you to enter your passphrase to unlock the RSA Identity. The client then decrypts the challenge using the private key, verifying your identity without disclosing the private key to the server.

- ☞ The file `$HOME/.ssh/authorized_keys` on the server lists the public keys of the users that are permitted to log into the system.

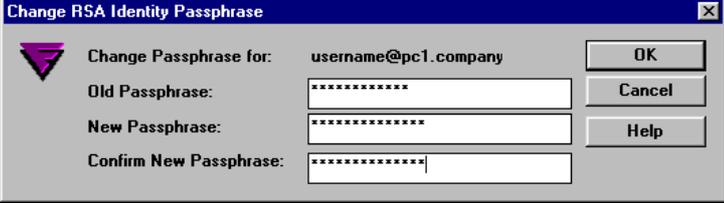


To connect to a remote computer using RSA authentication, follow these steps:

1. From the Edit menu, choose Properties.
  2. In the Connection dialog box, enter the name of the remote system to connect to, and the user name for your account.
  3. In the Options group, select RSA authentication. If RSA authentication is not available, see Section 5.4, "Setting Up Your Account for RSA Authentication," on page 164.
  4. Click the **OK** button.
  5. From the File menu, choose Connect. F-Secure SSH will establish the connection and ask for the passphrase you provided to protect the RSA identity file.
  6. In the dialog box, type your passphrase.
  7. Click the **OK** button to connect.
-  **If RSA authentication fails, F-Secure SSH reverts to password authentication and prompts you for a password. The password is sent to the remote host for checking. However, since all communication is encrypted, the password cannot be seen by anyone spying on network traffic.**

## Changing Your RSA Identity Passphrase

To prevent unauthorized persons from using your RSA Identity to gain access to your connections, you should use a passphrase to protect the Identity file. Change your passphrase periodically.



To change your RSA Identity passphrase, do the following:

1. Choose Edit > Properties > RSA Identity.
2. In the RSA Identity property page, click the **Passphrase** button.
3. Your RSA Identity Comment will be displayed in the Change RSA Identity Passphrase dialog box.
4. In the Change RSA Identity Passphrase dialog box, type your current passphrase, a new passphrase, and confirm the new passphrase.
5. Click the **OK** button. Your new RSA Identity passphrase takes effect immediately.
6. In the RSA Identity property page, click the **OK** button.

## 5.5 Forwarding

### Securing X11 Connections

F-Secure SSH can secure X11 connections to an X Window System Server running on your computer.

If the Forward X11 option in the Connection dialog box is selected, all connections to the X11 display are automatically forwarded through the encrypted channel.

Do not manually set the DISPLAY environment variable. The DISPLAY value set by F-Secure SSH will point to the server machine, but with a display number greater than zero. This is normal. It occurs because the program creates a proxy X server on the server machine for forwarding the connections over the encrypted channel.

F-Secure SSH will automatically set up Xauthority data on the server machine. For this purpose, it will generate a random authorization cookie and store it in Xauthority on the server. It will verify that any forwarded connections carry this cookie, and it will replace the cookie with the real cookie when the connection is opened. The real authentication cookie is never sent to the server machine (and no cookies are sent in the clear).

To establish a secure X11 connection from a remote host to an X Window System Server running on your computer, follow these steps:

1. Start your PC X Window System Server.
2. Make sure that connections to the PC X Window System Server are allowed from the IP address 127.0.0.1 (the localhost).
3. From the Edit menu, choose Properties.
4. In the Connection dialog box, select Forward X11.
5. Click the **OK** button.
6. From the File menu, choose Connect. Make the connections as specified in Connecting Using Password Authentication, or Connecting Using RSA Authentication.
7. Type a command in the terminal window to start any X11 application (for example, xterm &).

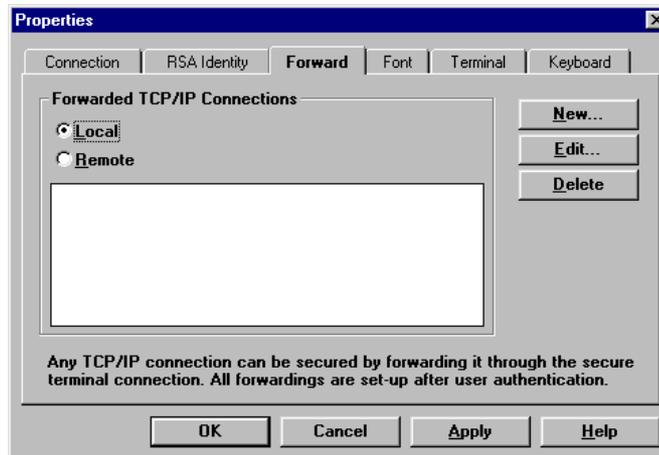
 **Use a fast cipher (Blowfish) for best performance with forwarded X11 connections.**

## Securing Arbitrary TCP/IP Connections

F-Secure SSH secures arbitrary TCP/IP connections using technology called TCP/IP port forwarding. TCP/IP port forwarding works by creating a proxy server for a source port used by a TCP/IP service. The proxy server waits on the local machine for a connection from a client program to the source port. The program then forwards the request and the data over the secure channel to the remote system. Then the F-Secure SSH server on the remote systems makes the connection to the destination host and to the destination port.

Most remote services that use TCP/IP can be secured. These include custom client-server applications, database systems, and services like HTTP, TELNET, POP, and SMTP.

F-Secure SSH provides automatic forwarding for the X11 Windowing System commonly used on UNIX machines. You must configure other TCP/IP connections manually.



Do the following steps to connect securely to a remote TCP/IP service. (In this example, a secure HTTP connection is used.)

1. Determine the port numbers used by the service. (For example, port 80 is used by the HTTP protocol and the Web servers.) See the */etc/services* file on your remote system for the default UNIX service and port configurations.
2. From the Edit menu, choose Properties.
3. In the Forward dialog box, select Local and click **New** to define the TCP/IP connections to be secure.
4. Specify a descriptive name for the forwarded connection (for example, 'Secure Intranet Connection').
5. Specify the source port, which is the port to be used for the client connection (for example, 80 for the secure Intranet connection).
6. Specify the destination host, which is the host that runs the service. If this is your connection host for your terminal session, use 127.0.0.1 (the localhost) to specify that the connection should be forwarded through the secure terminal connection. Then connect immediately to the service. If the service is running on another host, you must specify the forwarding to that host. Connection from the terminal session host onward will not be encrypted.
7. Specify the destination port, which is the port used by the server to wait for connections (for example, 80 for Web servers).
8. If you specify a local forwarding, make sure that "Allow Local Connection Only" is checked to prevent your forwarded connections from being used by any other computer except your own. See the Security Notice at the end of this section.
9. Click the **OK** button.
10. From the File menu, choose connect and make the connections as specified in Connecting Using Password Authentication, or in Connecting Using RSA Authentication.
11. If you want another application (such as a WWW browser) to start automatically when you double-click an *.ssh* file, define the application in the Application to Start edit box.

**Edit Local Forwarding**

**Forwarded Connection**

**Name**  
Secure Intranet Connection

**Source Port**  
80

**Destination Host**  
i-www.company.com

**Destination Port**  
80

Allow local connections only

**Application to Start**

**File**  
C:\PROGRA~1\NETSCAPE\COMMUN~1\PROGRAM\

Browse

OK  
Cancel  
Help

- Forwarding a connection to any host other than the terminal session host will only encrypt the forwarded connection up to the terminal session host. The connection from there on will not be encrypted to the destination host. The destination host should always be in a trusted network, or it should be the host for the terminal session.

**Edit Local Forwarding**

**Forwarded Connection**

**Name**  
Secure Intranet Connection

**Source Port**  
80

**Destination Host**  
127.0.0.1

**Destination Port**  
80

Allow local connections only

**Application to Start**

**File**  
C:\PROGRA~1\NETSCAPE\COMMUN~1\PROGRAM\

Browse

OK  
Cancel  
Help

- ✦ **SECURITY NOTICE:** If you do not check the **Allow Local Connection Only** option in the **Edit Local Forwarding** dialog box, then any host can connect to the forwarded source port on your computer. And that host will connect to the remote machine through your secured connection.

## 5.6 F-SECURE.INI File

*f-secure.ini* is a shared initialization file for all Data Fellows' F-Secure Windows applications. It contains several settings, including default language and directories.

When F-Secure SSH is installed, the setup program checks for the existence of the *f-secure.ini* file in the Windows directory. If the file does not exist, it is created. If the file exists and contains the settings from a previous installation, Setup will check for a previous installation and use its settings as the basis for the new installation.

During the installation process, the following entries are written to the *f-secure.ini* file.

Setting	Description
Language	The preferred language. If the language is not available for one of the F-Secure products, US English (ENU) is used by default.
RandomSeed	The seed value for the cryptographically strong random number generator. It is updated each time an F-Secure application closes.
InstallDirectory	The location for the <i>readme.txt</i> file and other files, such as connection template files ( <i>.ssh</i> files), RSA identity files ( <i>identity</i> and <i>identity.pub</i> ), and the <i>ssh.ini</i> file. Install Directory is also the preferred working directory that will be set to <i>ssh.exe</i> when a program manager icon is created.
ProgramDirectory	Program Directory is the directory that contains the F-Secure SSH executables and libraries (for example, <i>ssh.exe</i> , <i>gmp.dll</i> , <i>keygen.exe</i> , <i>fssenu.dll</i> , etc).

**Sample F-SECURE.INI file:**

```
[Settings]
RandomSeed=f69f64...

[F-Secure SSH]
Language=ENU
DefaultUser=root
InstallDirectory=C:\F-SECURE\SSH
ProgramDirectory=C:\F-SECURE\SSH\PROGRAM

[F-Secure SSH Recent Host List]
DefaultHost=ssh.company.com
NumHosts=3
Host1=ssh.company.com
Host2=www.company.com
Host3=ftp.company.com

[F-Secure SSH Recent File List]
File1=C:\F-SECURE\SSH\CONNECT.SSH

[Web Connection Settings]
BrowserDefault=C:\NETSCAPE\PROGRAMS\NETSCAPE.EXE
BrowserArea=Europe
BrowserConfigured=Yes
```

- ☞ **If the *f-secure.ini* file is not in the Windows directory, it is searched for in the directory containing the *ssh.exe* file. If it is not found there, a new *f-secure.ini* file is created in the Windows directory.**

## 5.7 DEFAULTS.SSH File

The *defaults.ssh* file is a connection template file used to load default settings each time SSH is run.

You can edit the *defaults.ssh* file by opening the file as a normal connection template. After saving your changes, all subsequent SSH sessions will have the new settings loaded.

If the *defaults.ssh* file does not exist, "hard-coded" defaults are used. To create a new *defaults.ssh* file, edit a Connection template and save it in the Install Directory (for example, *c:\f-secure\ssh\defaults.ssh*).

## 6. F-Secure SSH 1 for UNIX

### 6.1 Overview

F-Secure SSH 1 for UNIX consists of two separate programs SSH1 Client (ssh) and SSH1 Server (sshd).

F-Secure SSH 1 server for UNIX provides secure login connections, file transfer, X11, and TCP/IP connections for F-Secure SSH clients over untrusted networks. The server uses cryptographic authentication, automatic session encryption, and integrity protection for all connections.

### **The F-Secure Product Family**

F-Secure products utilize the SSH protocol as a generic transport-layer encryption mechanism, providing both host authentication and user authentication, along with privacy and integrity protection.

The encryption technology has been developed in Europe and does not fall under the U.S. ITAR export regulations. F-Secure products can be used in every country where encryption is legal, including the U.S. F-Secure products are sold with pre-licensed patented encryption algorithms to provide the best possible security.

The F-Secure SSH 1 for UNIX server can be used together with F-Secure SSH 1 clients for Windows, Macintosh, and UNIX to make secure login connections to remote offices. The F-Secure SSH 1 server includes tools for secure systems administration. Tools are provided for secure file transfer and for secure backups using public-key encryption.

## F-Secure SSH 1 Server

The F-Secure SSH 1 for UNIX server provides users with secure login connections, file transfer, X11, and TCP/IP connections over untrusted networks. The server uses cryptographic authentication, automatic session encryption, and integrity protection for all transferred data. RSA is used for key exchange and authentication, and symmetric algorithms, Blowfish or three-key triple-DES (3DES), for encrypting transferred data.

System administrators can use tools provided in the server package to replace existing RSH, RLOGIN, RCP, RDIST, and TELNET protocols. This will enable administrators to perform all remote system administration tasks over secure connections. A tool is also included for making secure backups with RSA-based public-key encryption.

The F-Secure SSH 1 for UNIX server supports TCP/IP port forwarding technology to connect otherwise insecure connections over a secure channel.

## F-Secure SSH 1 Client Products

F-Secure SSH 1 clients provide the users with secure login connections over untrusted networks. The F-Secure SSH 1 client acts as a replacement for the telnet protocol, taking advantage of the cryptographic authentication, automatic session encryption, and integrity protection methods that are defined by the SSH1 protocol. F-Secure SSH 1 clients fully support VT100 terminal emulation and ANSI colors.

The F-Secure SSH 1 clients also support TCP/IP port forwarding-technology to connect arbitrary and otherwise insecure connections over a secure channel. TCP/IP port forwarding works by creating a proxy server for a source port that a TCP/IP service uses. The proxy server waits on the local machine for a connection from a client program to the source port. F-Secure SSH 1 then forwards the request and the data over the secure channel to the remote system. The F-Secure SSH 1 server on the remote system makes the final connection to the destination host and the destination port.

Most remote services that use TCP/IP can be secured, including custom client-server applications, database systems, and services like HTTP, TELNET, POP, SMTP. F-Secure SSH 1 also provides automatic forwarding for the X11 Windowing System commonly used on UNIX machines.

## 6.2 Portability

F-Secure SSH 1.3.7 has been used in the following environments:

- 386BSD 0.1; i386
- AIX 3.2.5, 4.1, 4.2; RS6000, PowerPC
- BSD 4.4; several platforms
- BSD/OS 1.1, 2.0.1; i486
- BSD/386 1.1; i386
- ConvexOS 10.1; Convex
- DGUX 5.4R2.10; DGUX
- FreeBSD 1.x, 2.x; Pentium
- HPUX 7.x, 9.x, 10.0; HPPA
- IRIX 5.2, 5.3; SGI Indy
- IRIX 6.0.1; Mips-R8000
- Linux 1.2.x, 2.0.x Slackware 2.x, 3.x, RedHat 2.1, 3.0; i486
- Linux/Mach3, Macintosh(PowerPC)
- Mach3; Mips
- Mach3/Lites; i386
- Machten 2.2VM (m68k); Macintosh
- NCR Unix 3.00; NCR S40
- NetBSD 1.0A, 1.1, 1.2; Pentium, Sparc, Mac68k, Alpha
- NextSTEP 3.3; i486
- OSF/1 3.0, 3.2, 3.2; Alpha
- Sequent Dynix/ptx 3.2.0 V2.1.0; i386
- SCO Unix; i386 (client only)
- SINIX 5.42; Mips R4000
- Solaris 2.3, 2.4, 2.5; Sparc, i386
- Sony NEWS-OS 3.3 (BSD 4.3); m68k
- SunOS 4.1.1, 4.1.2, 4.1.3, 4.1.4; Sparc, Sun3
- SysV 4.x; several platforms
- Ultrix 4.1; Mips
- Unicos 8.0.3; Cray C90

Please tell us about other environments where you have used ssh. And please send us any patches you had to use, so that they can be integrated into the distribution. Send them to us at [F-Secure-SSH-BUGS@DataFellows.com](mailto:F-Secure-SSH-BUGS@DataFellows.com).

In your bug reports, include the SSH version number and machine type. If applicable, include the output of the `-v` option from the client side, and the output of the `-d` option from the server.

## Hints and Tips

**Linux note:** Some linux systems have a bug which causes an error about `libc.so.4` when compiling `ssh`. This can be solved in any of the following ways:

- Do `ln -s libc.so /usr/lib/libg.so` as root.
- Install `gcc-2.7.0`.
- Configure SSH with `CFLAGS=-O ./configure` (that is, without debug info).

More information on this problem is available in <ftp://ftp.netcom.com/pub/ze/zenon/linux>.

**BSDI BSD/OS note:** Apparently the `gcc` that comes with BSD/OS is broken. Use `CC=cc ./configure` or `setenv CC cc; ./configure` when configuring, in order to force it to use `cc` instead of `gcc`.

**ConvexOS note:** Convex `make` is broken. Install GNU `make` first, if you have trouble compiling `ssh`.

**Ultrix note:** Ultrix `/bin/sh` is broken. Run `configure` with `"/bin/sh5 configure [options]"` if you are on Ultrix.

On alpha, you should edit `rsaref2/source/global.h`, and make `UINT4` "unsigned int" instead of "unsigned long int".

## 6.3 Installing F-Secure SSH 1 for UNIX

F-Secure SSH 1 should always be installed as root. For more information about installation, see Section 6.8, "Advanced Installation," on page 201.

For most machines and configurations, you only need to do the following steps:

1. Unpack the distribution file with GZIP and TAR.
2. Compile the source code using the following commands in the distribution directory:

```
./configure
make
make install
```

In order to generate host keys and to install configuration files in a networked environment with a shared binary directory, it is enough to "make install" on one machine, and then "make hostinstall" on the other machines.

Examine (and edit, if needed) the following files:

- /etc/sshd\_config (server configuration file)
- /etc/ssh\_config (client configuration file - defaults for users).

You may want to create the /etc/ssh\_known\_hosts for your site and update it periodically. (See the manual page for make-ssh-known-hosts on how to do this easily.) The file format is documented on the sshd manual page.

You should edit /etc/rc.local or equivalent to start sshd at boot.

The source is written in ANSI C, and requires an ANSI C compiler or GCC.

A copy of GCC is available on all major FTP sites (for example, in <ftp://prep.ai.mit.edu:/pub/gnu>).

## 6.4 Using F-Secure SSH 1 for UNIX

ssh is used for starting a secure terminal connection to a remote host, executing commands on a remote host and creating tunnels for the secure transferring of TCP packets and X11 connections. This utility is intended to be used as a secure substitute for rlogin, rsh and telnet. It provides secure, encrypted communications between two hosts over an unsecure network.

SSH1 connects and logs into the specified hostname. The user must prove his identity to the remote machine by using one of several methods. One method is RSA-based authentication, a scheme based on public-key cryptography.

The format for starting an SSH session from the UNIX command prompt is:

```
ssh [options] hostname [command]
```

If the command contains spaces, enclose the whole command inside quotation marks.

In the following example, a connection is made to second.host.com using F-Secure ssh. In this case, the username on second.host.com is the same as the username on the system being connected from.

```
my.company.com% ssh second.host.com
andrew's password:
Last login: Wed Jun 16 08:48:59 1999
second.host.com%
```

In the following example, a connection is made to a host on which the user name is different from the one on the system the connection is made from.

```
my.company.com% ssh -l joseph third.host.com
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)? yes
Host 'third.host.com' added to the list of known hosts.
Creating random seed file ~/.ssh/random_seed. This may take a
while.
joseph@third.host.com's password:
Last login: Wed Jun 16 10:22:07 1999 from my.company.com
Linux 2.0.35.
third.host.com:~$
```

## Command Line Options

- a** Disables forwarding of the authentication agent connection. This may also be specified on a host-by-host basis in the configuration file.
- c 3des|blowfish|des|none** Selects the cipher to use for encrypting the session. **3des** (pronounced "triple-des") is used by default. It is believed to be very secure. **3des** is encrypt-decrypt-encrypt triple with three different keys. It is presumably more secure than **des**. **des** is the data encryption standard, but it is breakable by government agencies, large corporations, and major criminal organizations. **blowfish** is a very fast block cipher algorithm invented by Bruce Schneier. It uses 128-bit keys. **none** disables encryption entirely; it is used for debugging and renders the connection insecure.
- e ch|^ch|none** Sets the escape character for sessions with a pty (default: ~). The escape character is only recognized at the beginning of a line. The escape character, when followed by a dot (.), closes the connection; when followed by control-Z, suspends the connection; and when followed by itself, sends the escape character once. Setting the character to `none' disables any escapes and makes the session fully transparent.
- f** Requests ssh to operate in the background after authentication is completed and forwardings have been established. This is useful if ssh asks for passwords or passphrases, but the user wants it to work in the background. This may also be useful in scripts. **-f** implies **-n**. The recommended way to start X11 programs at a remote site is with a command like *"ssh -f hostname application"*.

- i** *identity\_file*                      Selects the file from which the identity (private key) for **RSA** authentication is read. Default is *.ssh/identity* in the user's home directory. Identity files may also be specified on a host-by-host basis in the configuration file. It is possible to have multiple **-i** options (and multiple identities specified in configuration files).
- l** *login\_name*                        Specifies the user to log in as on the remote machine. This may be specified on a host-by-host basis in the configuration file.
- n**                                        Redirects stdin from */dev/null* (in effect, prevents reading from stdin). This must be used when **ssh** is run in the background. A common trick is to use this to run X11 programs in a remote machine. For example, "ssh -n shadows.cs.hut.fi emacs &" will start an emacs on shadows.cs.hut.fi, and the X11 connection will be automatically forwarded over an encrypted channel. The **ssh** program will be put in the background. (This does not work if **ssh** needs to ask for a password or passphrase; see also the **-f** option.)
- o** *'option'*                            Can be used to give options in the format used in the config file. This is useful for specifying options for which there is no separate command-line flag. The option has the same format as a line in the configuration file.
- p** *port*                                Port to connect to on the remote host. This can be specified on a per-host basis in the configuration file.
- q**                                        Quiet mode. Suppresses all warning and diagnostic messages. Only fatal errors are displayed.
- t**                                        Force pseudo-tty allocation. This can be used to execute arbitrary screen-based programs on a remote machine. This can be very useful when implementing menu services.

- v Verbose mode. Causes **ssh** to print debugging messages about its progress. This is helpful in debugging connection, authentication, and configuration problems.
- x Disables X11 forwarding. This can also be specified on a per-host basis in a configuration file.
- C Requests compression of all data (including **stdin**, **stdout**, **stderr**, and data for forwarded X11 and TCP/IP connections). The compression algorithm is the same as used by **gzip**, and the "level" can be controlled by the **CompressionLevel** option. Compression is desirable on modem lines and other slow connections, but will only slow down things on fast networks. The default value can be set on a host-by-host basis in the configuration files, using the **Compress** option.
- L *source port: destination host:destination port*  
Specifies that the given **source port** on the local (client) host is to be forwarded to the given **destination host** and **destination port** on the remote side. This works by allocating a socket to listen to the **source port** on the local side. When a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to the **destination host:destination port** from the remote machine. Port forwardings can also be specified in the configuration file. Only root can forward privileged ports.

**-R** *source port: destination host:destination port*

Specifies that the given **source port** on the remote (server) host is to be forwarded to the given **destination host** and **destination port** on the local side. This works by allocating a socket to listen to the **source port** on the remote side. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to the **destination host:destination port** from the local machine. Port forwardings can also be specified in the configuration file. Privileged ports can be forwarded only when logging in as root on the remote machine.

## Technical Description

**ssh** is a program for logging into a remote machine and for executing commands from the remote machine. Ssh is intended to replace rlogin and rsh, and provide secure encrypted communications between two untrusted hosts over an insecure network. In addition, X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel.

Ssh connects and logs into the specified hostname. The user must prove his identity to the remote machine using one of several methods.

First, if the user's machine is listed in `/etc/hosts.equiv` or `/etc/shosts.equiv` on the remote machine, and the user names are the same on both sides, the user is immediately permitted to log in. Second, if `.rhosts` or `.shosts` exist in the user's home directory on the remote machine and contains a line with the name of the client machine and the name of the user on that machine, the user is permitted to log in. This form of authentication by itself is normally not allowed by the server because it is not secure.

The second (and primary) authentication method is the `rhosts` or `hosts.equiv` method combined with RSA-based host authentication. In this method, login is permitted only if both the login would be permitted by `.rhosts`, `.shosts`, `/etc/hosts.equiv`, or `/etc/shosts.equiv`, AND the client's host key can be verified. (See `$HOME/.ssh/known_hosts` and `/etc/ssh_known_hosts` in the FILES section.) This authentication method closes security holes presented by

IP spoofing, DNS spoofing and routing spoofing. [Note to the administrator: /etc/hosts.equiv, .rhosts, and the rlogin/rsh protocol in general are inherently insecure and should be disabled if security is desired.]

As a third authentication method, ssh supports RSA-based authentication. This scheme is based on public-key cryptography. RSA is one cryptosystem in which encryption and decryption are accomplished using separate keys, and in which the decryption key cannot be derived from the encryption key. In this system, each user creates a public/private key pair for authentication purposes. The server knows the public key, and only the user knows the private key. The file \$HOME/.ssh/authorized\_keys lists the public keys that are permitted to log in. When the user logs in, the ssh program tells the server which key pair to use for authentication. If the server verifies the key, it sends F-Secure SSH 1 Client a challenge in the form of a random number encrypted by the user's public key. The challenge can only be decrypted using the proper private key. The user's client then decrypts the challenge using the private key, proving that he/she knows the private key but without disclosing it to the server.

SSH1 implements the RSA authentication protocol automatically. The user creates his RSA key pair by running `ssh-keygen(1)`. This stores the private key in `.ssh/identity` and the public key in `.ssh/identity.pub` in the user's home directory. The user should then copy the `identity.pub` to `.ssh/authorized_keys` in his home directory on the remote machine. (The `authorized_keys` file corresponds to the conventional `.rhosts` file, and has one key per line, although the lines can be very long.) Afterwards, the user can log in without giving the password. RSA authentication is much more secure than `rhosts` authentication.

The most convenient way to use RSA authentication may be with an authentication agent.

If other authentication methods fail, ssh prompts the user for a password. The password is sent to the remote host for checking. However, because all communications are encrypted, the password cannot be seen by someone listening in on the network.

When the user's identity has been accepted by the server, the server either executes the given command, or logs into the machine and gives the user a normal shell on the remote machine. All communication with the remote command or shell will be automatically encrypted.

If a pseudo-terminal has been allocated (normal login session), the user can disconnect with `~.`, and suspend ssh with `~^Z`. All forwarded connections can be listed with `~#`, and if the session blocks waiting for forwarded X11 or TCP/IP connections to terminate, it can be backgrounded with `~&`. (This

should not be used while the user shell is active, because it can cause the shell to hang.) All available escapes can be listed with “~?”. A single tilde character can be sent as “~~”. Or it can be sent by following the tilde with a character other than those described above.) To be interpreted as special, the escape character must always be the first character of a new line. The escape character can be changed in configuration files or on the command line.

If no pseudo terminal has been allocated, the session is transparent and binary data can be reliably transferred. On most systems, setting the escape character to “none” will make the session transparent even if a pseudo terminal is used.

The session terminates when the command or shell on the remote machine exists and all X11 and TCP/IP connections have been closed. The exit status of the remote program is returned as the exit status of ssh.

If the user is using X11 (the DISPLAY environment variable is set), the connection to the X11 display is automatically forwarded to the remote side in such a way that any X11 programs started from the shell (or command line) will go through the encrypted channel, and the connection to the real X server will be made from the local machine. The user should not manually set DISPLAY. Forwarding of X11 connections can be configured on the command line or in configuration files.

The DISPLAY value set by ssh will point to the server machine, but with a display number greater than zero. This is normal, and happens because ssh creates a “proxy” X server on the server machine for forwarding the connections over the encrypted channel.

Ssh will also automatically set up Xauthority data on the server machine. For this purpose, it will generate a random authorization cookie, store it in Xauthority on the server, and verify that any forwarded connections carry this cookie and replace it by the real cookie when the connection is opened. The real authentication cookie is never sent to the server machine (and no cookies are sent in plain text).

If the user is using an authentication agent, the connection to the agent is automatically forwarded to the remote side unless disabled in a command line or in a configuration file.

Forwarding of arbitrary TCP/IP connections over the secure channel can be specified either on a command line or in a configuration file. One possible application of TCP/IP forwarding is to connect securely to an electronic purse; another application is to go through firewalls.

Ssh automatically maintains and checks a database containing RSA-based identifications for all previously used hosts. The database is stored in `.ssh/known_hosts` in the user's home directory. Additionally, the file `/etc/ssh_known_hosts` is automatically checked for known hosts. Any new hosts are automatically added to the user's file. If a host's identification changes, ssh gives a warning and disables password authentication to prevent a Trojan horse from retrieving the user's password. This mechanism also prevents man-in-the-middle attacks. The **StrictHostKeyChecking** option can be used to prevent logins to machines whose host key is not known or has changed.

## Installation

**Ssh** is normally installed as `suid root`. It needs root privileges only for rhosts authentication (rhosts authentication requires that the connection must come from a privileged port, and allocating such a port requires root privileges). It also needs to be able to read `/etc/ssh_host_key` to perform RSA host authentication. It is possible to use ssh without root privileges, but rhosts authentication will then be disabled. Ssh drops any extra privileges immediately after the connection to the remote host has been made.

Considerable work has been put into making **ssh** secure. However, if you find a security problem, please report it immediately to `<F-Secure-SSH-Bugs@DataFellows.com>`.

## 6.5 Using F-Secure sshd 1.3.7 for UNIX

**sshd** (F-Secure SSH1 Daemon) is the daemon program for F-Secure SSH 1. Together these programs can be used to replace **rlogin** and **rsh** programs, and provide secure encrypted communications between two hosts over an insecure network. The programs are intended to be as easy to install and use as possible.

### Starting the Server

The server should be started at boot from `/etc/rc` or equivalent. It need not be given any arguments; however, an optional “-b bits” flag may be used to specify RSA key size (default is 768 bits). Key sizes less than 512 bits can be broken; larger key sizes generally mean more security but require more time to generate and use. 1024 bits is secure for any practical time with current technology.

The server is not started using `inetd`, because it needs to generate the RSA key before serving the connection, and this can take about a minute on slower machines. On a fast machine, and small (breakable) key size (< 512 bits), it may be feasible to start the server from `inetd` on every connection. The server must be given “-i” flag if started from `inetd`.

## Options

Option	Description
<b>-b</b> <i>bits</i>	Specifies the number of bits in the server key (default 768).
<b>-d</b>	Debug mode. The server sends verbose debug output to the system log, and does not put itself in the background. The server also will not fork and will only process one connection. This option is only intended for debugging for the server.
<b>-f</b> <i>configuration_file</i>	Specifies the name of the configuration file. The default is <i>/etc/sshd_config</i> .
<b>-g</b> <i>login_grace_time</i>	Gives the grace time for clients to authenticate themselves (default 600 seconds). If the client fails to authenticate the user within this many seconds, the server disconnects and exits. A value of zero indicates no limit.
<b>-h</b> <i>host_key_file</i>	Specifies the file from which the host key is read (default <i>/etc/ssh_host_key</i> ). This option must be given if sshd is not run as root (as the normal host file is normally not readable by anyone but root).
<b>-i</b>	Specifies that sshd is being run from inetd. Sshd is normally not run from inetd because it needs to generate the server key before it can respond to the client, and this may take tens of seconds. Clients would have to wait too long if the key was regenerated every time. However, with small key sizes (e.g. 512) using sshd from inetd may be feasible.

- k** *key\_gen\_time* Specifies how often the server key is regenerated (default 3600 seconds, or one hour). The motivation for regenerating the key fairly often is that the key is not stored anywhere, and after about an hour, it becomes impossible to recover the key for decrypting intercepted communications even if the machine is cracked into or physically seized. A value of zero indicates that the key will never be regenerated.
- p** *port* Specifies the port on which the server listens for connections (default 22).
- q** Quiet mode. Nothing is sent to the system log. Normally the beginning, authentication, and termination of each connection is logged.

## Login Process

When a user successfully logs in, **sshd** does the following:

1. If the login is on a tty, and no command has been specified, prints the last login time and `/etc/motd` (unless prevented in the configuration file or by `$HOME/.hushlogin`; see the FILES section).
2. If the login is on a tty, records the login time.
3. Checks `/etc/nologin`; if it exists, prints contents and quits (unless root).
4. Changes to run with normal user privileges.
5. Sets up basic environment.
6. Reads `/etc/environment`, if it exists.
7. Reads `$HOME/.ssh/environment`, if it exists.
8. Changes to user's home directory.
9. If `$HOME/.ssh/rc` exists, runs it (with the user's shell); else if `/etc/sshr` exists, runs it (with `/bin/sh`); otherwise runs `xauth`. The 'rc' files are given the X11 authentication protocol and cookie in standard input.
10. Runs user's shell or command.

## Technical Description

**Sshd** is the daemon that listens for connections from clients. It is normally started at boot from **/etc/rc.local** or equivalent. It forks a new daemon for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange.

**Sshd** works as follows. Each host has a host-specific RSA key (normally 1024 bits) used to identify the host. Additionally, when the daemon starts, it generates a server RSA key (normally 768 bits). This key is normally regenerated every hour if it has been used, and is never stored on disk.

Whenever a client connects the daemon, the daemon sends its host and server public keys to the client. The client compares the host key against its own database to verify that it has not changed. The client then generates a 256-bit random number. It encrypts this random number using both the host key and the server key, and sends the encrypted number to the server. Both sides then start to use this random number as a session key which is used to encrypt all further communications in the session. The rest of the session is encrypted using a conventional cipher. Currently, 3DES, Blowfish, DES, and IDEA as an option. 3DES is used by default. The client selects the encryption algorithm to use from those offered by the server.

Next, the server and the client enter an authentication dialog. The client tries to authenticate itself using `.rhosts` authentication, `.rhosts` authentication combined with RSA host authentication, RSA challenge-response authentication, or password based authentication.

Rhosts authentication is normally disabled because it is fundamentally insecure, but can be enabled in the server configuration file if desired. System security is not improved unless **rshd(8)**, **rlogind(8)**, **rexecd(8)**, and **rexed(8)** are disabled. (This completely disables **rlogin(1)** and **rsh(1)** into that machine.)

If the client successfully authenticates itself, a dialog for preparing the session is entered. At this time, the client may make request things such as allocating a pseudo-tty, forwarding X11 connections, forwarding TCP/IP connections, or forwarding the authentication agent connection over the secure channel.

Finally, the client either requests a shell or execution of a command. The sides then enter session mode. In this mode, either side may send data at any time, and such data is forwarded to or from the shell or command on the server side, and the user terminal in the client side.

When the user program terminates, and all forwarded X11 and other connections have been closed, the server sends command exit status to the client, and both sides exit.

**Sshd** can be configured using command-line options or a configuration file. Command-line options override values specified in the configuration file.

**Sshd** rereads its configuration file if it is sent the hangup signal, SIGHUP.

## 6.6 Using scp

SCP copies files between hosts on a network. It uses SSH 1 for data transfer. And it uses the same authentication and provides the same security as F-Secure SSH. SCP asks for passwords or passphrases if they are needed for authentication.

Any file name may contain a host and user specification to indicate that the file is to be copied to or from that host. Copies between two remote hosts are permitted.

The format for copying files between two hosts is:

```
scp [options] [[user@]host1:]filename1... [[user@]host2:]filename2
```

For example, to copy the file `program.exe` from your local hard drive to `second.host.com`, where your username is *andrews*, you would enter:

```
scp program.exe andrews@second.host.com:program.exe
```

## Command Line Options

<b>-c</b> <i>cipher</i>	Selects the cipher to use for encrypting the data transfer. This option is directly passed to <b>ssh</b> .
<b>-i</b> <i>identity_file</i>	Selects the file from which the identity (private key) for RSA authentication is read. This option is directly passed to <b>ssh</b> .
<b>-p</b>	Preserves modification times, access times, and modes from the original file.
<b>-r</b>	Copies entire directories recursively.
<b>-v</b>	Verbose mode. Causes <b>scp</b> and <b>ssh</b> to print debugging messages about their progress. This is helpful in debugging connection, authentication, and configuration problems.
<b>-B</b>	Selects batch mode (prevents requests for passwords or passphrases).
<b>-C</b>	Enable compression. Passes the <b>-C</b> flag to <b>ssh</b> to enable compression.
<b>-P</b> <i>port</i>	Specifies the port to connect to on the remote host. This option is written with a capital P because <b>-p</b> is reserved for preserving the times and modes of the file in <b>rcp</b> .

## Derivation

**Scp** is based on the **rcp** program in BSD source code from the Regents of the University of California.

## 6.7 Public-key authentication

### ssh-keygen

ssh-keygen generates and manages authentication keys for SSH. Normally, each user who wants to use SSH1 with RSA authentication runs it one time to create the authentication key in `$HOME/.ssh/identity`. Additionally, the system administrator may use this to generate host keys.

**Ssh-keygen** generates and manages authentication keys for **ssh** (1). Normally each user who wants to use **ssh** with RSA authentication runs this once to create the authentication key in `$HOME/.ssh/identity`. Additionally, the system administrator may use this to generate host keys.

Normally this program generates the key and asks for a file in which to store the private key. The public key is stored in a file with the same name but ".pub" appended. The program also asks for a passphrase. The passphrase may be empty to indicate no passphrase (host keys must have empty passphrase), or it may be a string of arbitrary length. Good passphrases are 10-30 characters long and are not simple sentences or otherwise easily guessed. (English prose has only one to two bits of entropy per word, and offers very bad passphrases). The passphrase can be changed later by using the `-p` option.

There is no way to recover a lost passphrase. If the passphrase is lost or forgotten, you will have to generate a new key and copy the corresponding public key to other machines.

The comment field in the key file is for the user's convenience to identify the key. The comment could include the key's purpose or any other useful information. The comment is initialized to `user@host` when the key is created, but can be changed using the `-c` option.

The cipher to be used when encrypting keys with a passphrase is by default 3des. Using the `-u` option, keys encrypted in any supported cipher can be updated to use this default cipher.

The format for creating an authentication key pair from the UNIX command prompt is:

```
ssh-keygen [options]
```

All options start with "-".

By default, `ssh-keygen` creates a 1024-bit DSA key pair, prompts you to enter and verify your passphrase, saves the private key to `$HOME/.ssh2/id_dsa_1024_a` and saves the public key to `$HOME/.ssh2/id_dsa_1024_a.pub`.

Example:

```
$ ssh-keygen1
Initializing random number generator...
Generating p: .....++ (distance 370)
Generating q: .....++ (distance 160)
Computing the keys...
Testing the keys...
Key generation complete.
Enter file in which to save the key (/home/andrews/.ssh/identity):
Enter passphrase:
Enter the same passphrase again:
Your identification has been saved in identity.
Your public key is:
1024 33
    158914471207145365759164611484925141075661866465928852400952329628
    250851002444552897771490312379434957528529939542774297207722673860
    046441416337788968590393436472981163197805307680496049655648787247
    470187896324092427871178147580479579902052511554942791297531008911
    578532864862823346458804176124698378516451493 andrews@my.host.com
Your public key has been saved in identity.pub
$
```

## Command Line Options

Option	Description
<b>-b</b> <i>bits</i>	Specifies the number of bits in the key to create. Minimum is 512 bits. Generally 1024 bits is considered sufficient; larger key sizes slow down the system and do not improve security. The default is 1024 bits.
<b>-c</b>	Requests changing the comment in the private and public key files. The program will prompt for the file containing the private keys, for passphrase if the key has one, and for the new comment.
<b>-f</b>	Specifies the file name in which to load and store the key.
<b>-p</b>	Requests changing the passphrase of a private key file instead of creating a new private key. The program will prompt for the file containing the private key, for the old passphrase, and twice for the new passphrase.
<b>-u</b>	Requests that the key's cipher is changed to the current default cipher (determined at compile-time; currently 3DES).
<b>-C</b>	Provides the new comment.
<b>-N</b>	Provides the new passphrase.
<b>-P</b>	Provides the (old) passphrase.

### Files

***\$HOME/.ssh/random\_seed***      Used for seeding the random number generator. This file should not be readable by anyone but the user. This file is created the first time the program is run, and is updated every time.

- `~/.ssh/identity`** Contains the RSA authentication identity of the user. This file should not be readable by anyone but the user. It is possible to specify a passphrase when generating the key; that passphrase will be used to encrypt the private part of this file using 3DES. This file is not automatically accessed by **`ssh-keygen`**, but it is offered as the default file for the private key.
- `~/.ssh/identity.pub`** Contains the public key for authentication. The contents of this file should be added to **`~/.ssh/authorized_keys`** on all machines where you want to log in using RSA authentication. There is no need to keep the contents of this file secret.

## ssh-agent

`ssh-agent` is a program used to hold authentication private keys. `ssh-agent` is started at the beginning of an X-session or a login session, and all other windows or programs are started as children of the `ssh-agent` program. (The command normally starts X or the user shell.) Programs started under the agent inherit a connection to it. The agent is automatically used for RSA authentication when logging to other machines which are using SSH.

The format for starting the `ssh-agent` service from the UNIX command prompt is:

```
ssh-agent [command]
```

If you have logged into a UNIX system from another non-native UNIX platform, you will not be able to successfully fork the `ssh-agent` to the background. The way to use `ssh-agent` in that situation is to use the following command:

```
ssh-agent $SHELL
```

This will start another shell on top of the original shell, with `ssh-agent` running in the background. When you exit the shell with a standard UNIX command, such as `exit`, `logout`, `quit`, you will return to the original shell from which `ssh-agent` was started.

Initially, the agent does not have any private keys. Keys are added using `SSH-ADD`. When executed without arguments, `SSH-ADD` adds the `~/.ssh/identity` file. If the identity has a passphrase, `SSH-ADD` asks for the passphrase (using a small X11 application if running under X11, or from the terminal if running without X). `SSH-ADD` then sends the identity to the agent.

Several identities can be stored in the agent. The agent can automatically use any of these identities.

The agent is run on the user's local PC, laptop, or terminal. Authentication data need not be stored on any other machine, and authentication passphrases never go over the network. However, the connection to the agent is forwarded over SSH remote logins. Thus the user can securely use the privileges given by the identities anywhere in the network .

A connection to the agent is inherited by child programs. There are two alternative methods for inheriting the agent. The preferred method is to have an open file descriptor which is inherited, and have an environment variable (SSH\_AUTHENTICATION\_FD) contain the number of this descriptor. This restricts access to the authentication agent to only those programs that are siblings of the agent, and it is fairly difficult even for root to get unauthorized access to the agent.

On some machines, an alternative method is used. A unixdomain socket is created (/tmp/ssh\_agent.\*), and the name of this socket is stored in the SSH\_AUTHENTICATION\_SOCKET environment variable. The socket is made accessible only to the current user. This method is easily abused by root or by another instance of the same user. The socket is only used if ssh is unable to find a file descriptor that would not be closed by shells.

The agent exits automatically when the command given on the command line terminates.

## Files

- |  |  |
|--|--|
| <b><i>\$HOME/.ssh/identity</i></b>       | Contains the RSA authentication identity of the user. This file should not be readable by anyone but the user. It is possible to specify a passphrase when generating the key; that passphrase will be used to encrypt the private part of this file. This file is not used by <b>ssh-agent</b> , but is normally added to the agent using <b>ssh-add</b> at login time. |
| <b><i>/tmp/ssh_agent.&lt;pid&gt;</i></b> | Unix-domain sockets used to contain the connection to the authentication agent. These sockets should only be readable by the owner. The sockets should get automatically removed when the agent exits.   |

## ssh-add

ssh-add adds identities to the authentication agent, **ssh-agent**. When run without arguments, it adds the file ***\$HOME/.ssh/identity***. Alternative file names can be given on the command line. If any file requires a passphrase, **ssh-add** requests the passphrase from the user. If the user is using X11, the passphrase is requested using a small X11 program; otherwise, it is read from the user's tty.

The authentication agent must be running and must be an ancestor of the current process for **ssh-add** to work.

If ssh-add needs a passphrase, it will read the passphrase from the current terminal if it was run from a terminal. If ssh-add does not have a terminal associated with it but DISPLAY is set, it will open an X11 window to read the passphrase. This is particularly useful when calling ssh-add from a .Xsession or related script. (Note that on some machines it may be necessary to redirect the input from /dev/null to make this work.)

The format for adding new keys to the ssh-agent using *ssh-add* from the UNIX command prompt is:

```
ssh-add [options] [files...]
```

### Options

- l Lists all identities currently represented by the agent.
- d Instead of adding the identity, removes the identity from the agent.
- D Deletes all identities from the agent.

## Exit Status

Ssh-add returns one of the following exit statuses. These may be useful in scripts.

0	The requested operation was performed successfully.
1	No connection could be made to the authentication agent. Presumably there is no authentication agent active in the execution environment of ssh-add.
2	The user did not supply a required passphrase.
3	An identify file could not be found, was not readable, or was in bad format.
4	The agent does not have the requested identity.
5	An unspecified error has occurred; this is a catchall for errors not listed above.

## Files

<code>\$HOME/.ssh/identity</code>	Contains the RSA authentication identity of the user. This file should not be readable by anyone but the user. It is possible to specify a passphrase when generating the key; that passphrase will be used to encrypt the private part of this file. This is the default file added by ssh-add when no other files have been specified.
-----------------------------------	--

## 6.8 Advanced Installation

### Configuration Options

The package comes with an Autoconf-generated configure script. The script accepts several options. All standard options, including:

<code>--prefix=PREFIX</code>	where to install files (default: subdirs of /usr/local)
<code>--exec_prefix=PREFIX</code>	where to install executables (default: same as prefix)
<code>--srcdir=DIR</code>	find sources in DIR (default: where configure is)
	Specific options:
<code>--with-rsh=PATH</code>	Use rsh specified by PATH when needed
<code>--with-etcdir=PATH</code>	Store system files in the given dir (default: /etc)
<code>--with-path=PATH</code>	Default path to pass to user shell.
<code>--with-rsaref</code>	Use rsaref2 from rsaref2 subdirectory.
<code>--with-libwrap[=PATH]</code>	Use libwrap (tcp_wrappers) and inetd.
<code>--with-socks4[=PATH]</code>	Include SOCKS (firewall traversal) support.
<code>--with-socks5[=PATH]</code>	Include SOCKS5 (firewall traversal) support.
<code>--with-securid[=PATH]</code>	Support for the SecurID card.
<code>--enable-warnings</code>	Adds -Wall to CFLAGS, if using gcc.

You may also configure the following variables:

<code>CC=compiler</code>	specify name of the C compiler (default: gcc or cc)
<code>CFLAGS=flags</code>	specify flags to C compiler (default: -O -g, or -O)
<code>LDFLAGS=flags</code>	specify flags to linker (default: none)

Alternate values can be given to configure in the environment, for example:

```
CC=xcc CFLAGS="-O2" LDFLAGS="-L/lib/zzz" ./configure
```

(If you have already configured, and later want to give some values on the command line, you may need to say. "make distclean" before reconfiguring.)

## Configuration Files

The server has a configuration file `/etc/sshd_config`, which specifies the permitted authentication methods, hosts, port number, etc. The defaults are acceptable for most sites, but you may want to check this file. Its format is documented in the manual page covering `sshd`.

The client reads the configuration file `/etc/ssh_config`, which gives site-wide defaults for various options. Options in this file can be overridden by user-specific configuration files. The file is documented on the manual page covering `ssh`.

## Makefile

The Makefile is generated from `Makefile.in` by running `configure`. It supports the following targets:

<code>all:</code>	compile everything
<code>install:</code>	install in <code>\$exec_prefix/bin</code> and <code>\$prefix/man/man1</code> .
<code>hostinstall:</code>	generate host key and install config files
<code>uninstall:</code>	remove installed files
<code>clean:</code>	remove object files and executables
<code>distclean:</code>	remove everything not in the distribution

## Libwrap And Inetd

`Sshd` normally does not use `inetd` or `tcp-wrappers`. However, it can be compiled to use these by adding `--with-libwrap` on the `configure` command line. This requires that the `tcp_wrappers` `libwrap.a` library and the associated `tcpd.h` have been installed somewhere where the compiler can find them. With `libwrap` support, `ssh` will process the `/etc/hosts.allow` and `/etc/hosts.deny` files, and use `inetd` if required by them. The name of the user on the client side (as returned by `inetd`) will be logged if requested by the configuration files. See `tcp_wrappers` documentation for more information.

## Replacing Rlogin And Rsh

The software has been designed so that it can be installed with the names `rlogin`, `rsh`, and `rcp`. And it will use the SSH protocol whenever the remote machine supports it. If the remote host does not support SSH, the software will automatically execute `rlogin/rsh` (after displaying a warning that there is no encryption).

Rlogin/rsh replacement is done as follows:

```
./configure --with-rsh=RSH-PATH --program-transform-  
name='s/^s/r/'  
make install,
```

where `RSH-PATH` is the complete pathname of the real `rsh` program. The `rlogin` program is searched in the same directory, with the name “`rlogin`”. This means that you need to copy both `rsh` and `rlogin` to the same directory, using these names. It is not sufficient to simply rename them, for example, to “`rsh.old`”.

This will create links for `rlogin`, `rsh`, and `rcp`. If you are installing in the same directory where `rlogin`, `rsh`, and `rcp` are normally (for example, `/usr/bin`), you must first move the original programs to another directory (for example, `/usr/lib/rsh`).

When you do this, build a file containing the host keys of all machines in your organization. Then copy this file to `/etc/ssh_known_hosts` on every machine. This will make `.rhosts` and `/etc/hosts.equiv` authentication work for users without any changes to the user configuration, but will be much more secure than conventional `.rhosts` and `/etc/hosts.equiv` authentication. This will also protect the users against router attacks where someone, perhaps remotely, reconfigures the routers to direct connections to a particular host to a different machine, which can then grab any passwords which the user types thinking she/he is connected to the real machine.

## 6.9 Configuring F-Secure SSH 1 for UNIX

### Configuration Files

SSH obtains configuration data from the following sources (in this order): system-wide configuration file (*/etc/ssh\_config*), the user's configuration file (*(\$HOME/.ssh/config)*), and command line options. For each parameter, the last value retrieved will be used. The configuration files contain sections bracketed by "Host" specifications, and that section is only applied for hosts that match one of the patterns given in the specification. The matched host name is the one given on the command line.

Since the first value retrieved for each parameter is used, more host-specific declarations should be given near the beginning of the file. General defaults should be given at the end of the file.

The configuration file has the following format:

Empty lines and lines starting with "#" are comments.

Otherwise a line takes the format "keyword arguments". The possible keywords and their meanings are as follows (note that the configuration files are case-sensitive):

**Host** restricts the following declarations (up to the next **Host** keyword) to those hosts that match one of the patterns given after the keyword. "\*" and "?" can be used as wildcards in the patterns. A single "\*" can be used as the entire pattern to provide global defaults for all hosts. The host is the **hostname** argument given on the command line (that is, the name is not converted to a canonized host name before matching).

<b>BatchMode</b>	If set to "yes", password querying will be disabled. This option is useful in scripts and other batch jobs where there are no users to supply a password. The argument must be "yes" or "no".
<b>Cipher</b>	Specifies the cipher to use for encrypting the session. , Ciphers currently supported are <b>3des</b> , <b>blowfish</b> , <b>des</b> , and <b>none</b> . The default is <b>3des</b> . Using <b>none</b> (no encryption) is intended only for debugging, and will make the connection insecure.

<b>Compression</b>	Specifies whether to use compression. The argument must be " <b>yes</b> " or " <b>no</b> ".
<b>CompressionLevel</b>	Specifies the compression level if compression is enabled. The argument must be an integer from 1 (fast) to 9 (slow, best). The default level is 6, which is good for most applications. The values used are the same as in in GNU GZIP.
<b>ConnectionAttempts</b>	Specifies the number of attempts to connect (at the rate of one per second) before falling back to <b>rsh</b> or exiting. The argument must be an integer. This may be useful in scripts if the connection sometimes fails.
<b>EscapeChar</b>	Sets the escape character (default: ~). The escape character can also be set on the command line. The argument should be a single character, '^', followed by a letter (or "none") to disable the escape character entirely (making the connection transparent for binary data).
<b>FallBackToRsh</b>	Specifies that when a connection via <b>ssh</b> fails due to a connectionrefused error (there is no <b>sshd</b> listening on the remote host), <b>rsh</b> should automatically be used instead. (A message warns that the session is unencrypted.) The argument must be " <b>yes</b> " or " <b>no</b> ".
<b>ForwardAgent</b>	Specifies whether a connection to the authentication agent will be forwarded to the remote machine. The argument must be " <b>yes</b> " or " <b>no</b> ".
<b>ForwardX11</b>	Specifies whether X11 connections will be automatically redirected over the secure channel and <b>DISPLAY</b> set. The argument must be " <b>yes</b> " or " <b>no</b> ".
<b>GlobalKnownHostsFile</b>	Specifies a file to use instead of <i>/etc/ssh_known_hosts</i> .

<b>HostName</b>	Specifies the real host name to log into. This can be used to specify nicknames or abbreviations for hosts. The default is the name given on the command line. Numeric IP addresses are also permitted (both on the command line and in <b>HostName</b> specifications).
<b>IdentityFile</b>	Specifies the file from which the user's RSA authentication identity is read (default <b>.ssh/identity</b> in the user's home directory). Additionally, any identities represented by the authentication agent will be used for authentication. The file name may use the tilde syntax to refer to a user's home directory. It is possible to have multiple identity files specified in configuration files; all these identities will be tried in sequence.
<b>KeepAlive</b>	<p>Specifies whether the system should send keepalive messages to the other side. If they are sent, death of the connection or a machine crash will be properly noticed. However, this means that connections will die if the route is down temporarily. Some people find that annoying.</p> <p>The default is <b>"yes"</b> (to send keepalives), and the client will notice if the network goes down or the remote host dies. This is important in scripts, and many users find this useful.</p> <p>To disable keepalives, the value should be set to <b>"no"</b> in both the server and the client configuration files.</p>
<b>LocalForward</b>	Specifies that a TCP/IP port on the local machine be forwarded over the secure channel to given <b>destination host:destination port</b> from the remote machine. The first argument must be a port number, and the second must be <b>destination host:destination port</b> . Multiple forwardings may be specified, and additional forwardings can be given on the command line. Only the root can forward privileged ports.

<b>PasswordAuthentication</b>	Specifies whether to use password authentication. The argument to this keyword must be “ <b>yes</b> ” or “ <b>no</b> ”.
<b>Port</b>	Specifies the port number to connect to on the remote host. Default is 22.
<b>ProxyCommand</b>	<p>Specifies the command to connect to the server. The command string extends to the end of the line, and is executed with <code>/bin/sh</code>. In the command string, <code>%h</code> will be substituted by the host name to connect, and <code>%p</code> will be substituted by the port. The command can basically be anything. It should read from its <b>stdin</b> and write to its <b>stdout</b>. It should eventually connect an <b>sshd</b> server running on some machine, or should execute “<code>sshd -l</code>” somewhere. Host key management will be done using the <code>HostName</code> of the host being connected (defaulting to the name typed by the user).</p> <p>Note that <code>ssh</code> can also be configured to support the SOCKS system using the <code>--with-socks4</code> or <code>--with-socks5</code> compile-time configuration option.</p>
<b>RemoteForward</b>	Specifies that a TCP/IP port on the remote machine will be forwarded over the secure channel to a given <b>destination host:destination port</b> from the local machine. The first argument must be a port number. The second argument must be <b>destination host:destination port</b> . Multiple forwardings may be specified. Additional forwardings can be given on the command line. Only the root can forward privileged ports.
<b>RhostsAuthentication</b>	Specifies whether to try <code>rhosts</code> -based authentication. This declaration affects only the client side and has no effect on security. Disabling <code>rhosts</code> authentication may reduce authentication time on slow connections. Most servers do not permit <code>RhostsAuthentication</code> because it is not secure. The argument to this keyword must be “ <b>yes</b> ” or “ <b>no</b> ”.

<b>RhostsRSAAuthentication</b>	Specifies whether to try rhosts-based authentication with RSA host authentication. This is the primary authentication method for most sites. The argument must be “ <b>yes</b> ” or “ <b>no</b> ”.
<b>RSAAuthentication</b>	Specifies whether to try RSA authentication. The argument to this keyword must be “ <b>yes</b> ” or “ <b>no</b> ”. RSA authentication will be attempted only if the identity file exists, or an authentication agent is running.
<b>StrictHostKeyChecking</b>	If this flag is set to “yes”, <b>ssh</b> will never automatically add host keys to the <b><i>\$HOME/.ssh/known_hosts</i></b> file, and it will not connect hosts whose host key has changed. This provides maximum protection against trojan horse attacks. However, it can be somewhat annoying if you do not have good <b><i>/etc/ssh_known_hosts</i></b> files installed and frequently connect new hosts. Basically, this option forces the user to manually add any new hosts. This option is normally disabled, and new hosts will automatically be added to the known host files. In either case, the host keys of known hosts will be verified automatically. The argument must be “ <b>yes</b> ” or “ <b>no</b> ”.
<b>User</b>	Specifies the user to log in as. This is useful if you have a different user name on different machines. You will not have to remember your user name to log in.
<b>UserKnownHostsFile</b>	Specifies a file to use instead of <b><i>\$HOME/.ssh/known_hosts</i></b> .

**UseRsh** Specifies that rlogin/rsh should be used for this host. If the host does not support the **ssh** protocol, then **ssh** will immediately execute **rsh**. If this option is specified, all other options (except **HostName**) are ignored. The argument must be "**yes**" or "**no**".

## Environment

**DISPLAY** The **DISPLAY** variable indicates the location of the X11 server. It is automatically set by **ssh** to point to a value of the form "hostname:n", where *hostname* indicates the host in which the shell runs; and *n* is an integer  $\geq 1$ . **Ssh** uses this special value to forward X11 connections over the secure channel. Normally, the user should not explicitly set **DISPLAY**, because that will render the X11 connection insecure. And this will require the user to manually copy any required authorization cookies.

**HOME** Set to the path of the user's home directory.

**LOGNAME** Same as **USER**; set for compatibility with systems that use this variable.

**MAIL** Set to point the user's mailbox.

**PATH** Set to the default **PATH**, as specified when compiling **ssh**. On some systems, set to */etc/environment* or */etc/default/login*.

**SSH\_AUTHENTICATION\_FD**

This is set to an integer value if you are using the authentication agent and a connection to it has been forwarded. The value indicates a file descriptor number used for communicating with the agent. On some systems, **SSH\_AUTHENTICATION\_SOCKET** may be used instead to indicate the path of a unix domain socket used to communicate with the agent. (This method is less secure, and it is only used on systems that do not support the first method.)

**SSH\_CLIENT**

Identifies the client-end of the connection. The variable contains three space-separated values: client ip-address, client port number, and server port number.

**SSH\_TTY**

This is set to the name of the tty (path to the device) associated with the current shell or command. If the current session has no tty, this variable is not set.

**TZ**

The time zone variable is set to indicate the present timezone if it was set when the daemon was started. (That is, the daemon passes the value on to new connections)

**USER**

Set to the name of the user logging in.

Additionally, **ssh** reads */etc/environment* and *\$HOME/.ssh/environment*, and adds lines to the environment, of the format *VAR\_NAME=value*. Some systems may have additional mechanisms for setting up the environment (such as */etc/default/login* on Solaris).

## Files

### **\$HOME/.ssh/known\_hosts**

Records host keys for all hosts the user has logged into that are not in */etc/ssh\_known\_hosts*.

### **\$HOME/.ssh/random\_seed**

Used for seeding the random number generator. This file contains sensitive data. Only the user should have read/write access. This file is created the first time the program is run and updated automatically. The user should never need to read or modify this file.

### **\$HOME/.ssh/identity**

Contains the RSA authentication identity of the user. This file contains sensitive data and should be readable by the user but nobody else. It is possible to specify a passphrase when generating the key. The passphrase will be used to encrypt the sensitive part of this file, using **3des**.

### **\$HOME/.ssh/identity.pub**

Contains the public key for authentication. (The public part of the identity file in human-readable form.) The contents of this file should be added to ***\$HOME/.ssh/authorized\_keys*** on all machines you want to log into using RSA authentication. This file is not sensitive and can be readable by anyone. This file is never used automatically and is not required. It is provided for the user's convenience.

### **\$HOME/.ssh/config**

This is the per-user configuration file. Its format is described above. This file is used by the **ssh** client. It usually does not contain any sensitive information. However, the recommended permissions are Read/Write for the user and Not Accessible by others.

**\$HOME/.ssh/authorized\_keys**

Lists the RSA keys for logging in as this user. The format is described in the **sshd** section of the manual. In simplest form, the format is the same as the .pub identity files. Each line contains the number of bits in modulus, public exponent, modulus, and comment fields, separated by spaces. This file is not highly sensitive, but the recommended permissions are Read/Write for the user, and Not Accessible by others.

**/etc/ssh\_known\_hosts**

System-wide list of known host keys. This file should be prepared by the system administrator to contain the public host keys of all machines in the organization. This file should be world-readable. This file contains public keys (one per line), in the following format (fields separated by spaces): system name, number of bits in modulus, public exponent, modulus, and optional comment field. When different names are used for the same machine, all names should be listed, separated by commas. The format is described in the **sshd** section of the manual.

The canonical system name (as returned by name servers) is used by **sshd** to verify the client host when logging in; other names are needed because **ssh** does not convert the user-supplied name to a canonical name before checking the key, because somebody with access to the name servers would be able to fool host authentication.

**/etc/ssh\_config**

System-wide configuration file. This file provides defaults for those values that are not specified in the user's configuration file, and for those users who do not have a configuration file. This file must be world-readable.

**\$HOME/.rhosts**

This file is used in **.rhosts** authentication to list the host/user pairs that are permitted to log in. (This file is also used by **rlogin** and **rsh**, making use of this file insecure.) Each line of the file contains a host name (in the canonical form returned by name

servers), and then a user-name on that host, separated by a space. This file must be owned by the user, and must not have write permissions for anyone else. The recommended permission is rRead/Write for the user, and Not Accessible by others.

Note that by default, `sshd` will be installed so that it requires successful RSA host-authentication before permitting `.rhosts` authentication. If your server machine does not have the client's host key in `/etc/ssh_known_hosts`, you can store it in `$HOME/.ssh/known_hosts`. You can do this easily by connecting back to the client from the server machine using `ssh`. This will automatically add the host key in `$HOME/.ssh/known_hosts`.

<code>\$HOME/.shosts</code>	This file is used exactly the same way as <code>.rhosts</code> . This file enables you to use <code>rhosts</code> authentication with <code>ssh</code> without permitting login with <code>rlogin</code> or <code>rsh</code> .
<code>/etc/hosts.equiv</code>	This file is used during <code>.rhosts</code> authentication. It contains canonical hosts names, one per line. (The full format is described in the <code>sshd</code> section of the manual. If the client host is found in this file, login is automatically permitted (provided client and server user-names identical). Successful RSA host authentication is normally required. This file should be writable only by root.
<code>/etc/shosts.equiv</code>	This file is processed exactly as <code>/etc/hosts.equiv</code> . This file may be useful to permit logins using <code>ssh</code> while preventing logins using <code>rsh</code> or <code>rlogin</code> .
<code>/etc/sshrd</code>	Commands in this file are executed by <code>ssh</code> after the user logs in, but before the user's shell (or command) is started.
<code>\$HOME/.ssh/rc</code>	Commands in this file are executed by <code>ssh</code> after the user logs in, but before the user's shell (or command) is started.

## 6.10 Configuring F-Secure sshd for UNIX

### Configuration File

**Sshd** reads configuration data from `/etc/sshd_config` (or the file specified with `-f` on the command line). The file contains keyword-value pairs, one per line. Lines starting with `#` and empty lines are interpreted as comments.

- AllowHosts** This keyword can be followed by any number of host name patterns, separated by spaces. If specified, login is allowed only from hosts whose name matches one of the patterns. `*` and `?` can be used as wildcards in the patterns. Normal name servers are used to map the client's host into a canonical host name. If the name cannot be mapped, its IP-address is used as the host name. By default all hosts are allowed to connect. Note that sshd can also be configured to use `tcp_wrappers` using the `--with-libwrap` compile-time configuration option.
- AllowForwardingPort** This keyword can be followed by any number of port numbers, separated by spaces. If specified remote forwardings are only allowed to for those ports whose number matches one of the patterns. `*` can be used as wildcard entry for all ports. `>x`, `<x`, and `x..x` formats can be used to specify greater than, less than or inclusive port range. By default, all port forwardings are allowed.
- AllowForwardingTo** This keyword can be followed by any number of hostname and port number patterns, separated by spaces. The port number pattern is separated from the hostname by a colon. If specified, forwardings from client are only allowed to those hosts and port pairs whose name and port number matches one of the patterns. `*` and `?` can be used as wildcards in the patterns for host names. Normal name servers are used to map the client's host into a canonical host name. If the name

cannot be mapped, its IP-address is used as the host name. '\*' can be used as wildcard entry for all ports. '>x', '<x', and 'x..x' formats can be used to specify greater than, less than, or inclusive port range. By default, all port forwardings are allowed.

**AllowUsers**

This keyword can be followed by any number of user name patterns, separated by spaces. If specified, login is allowed only as users whose name matches one of the patterns. '\*' and '?' can be used as wildcards in the patterns. By default, logins as all users are allowed. Note that the all other login authentication steps must still be successfully completed. AllowUsers and DenyUsers are additional restrictions.

**DenyHosts**

This keyword can be followed by any number of host name patterns, separated by spaces. If specified, login is disallowed from the hosts whose name matches any of the patterns.

**DenyUsers**

This keyword can be followed by any number of user name patterns, separated by spaces. If specified, login is disallowed as users whose name matches any of the patterns.

**FascistLogging**

Specifies whether to use verbose logging. Verbose logging violates the privacy of users and is not recommended. The argument must be *yes* or *no*. The default is *no*.

**HostKey**

Specifies the file containing the private host key (default */etc/ssh\_host\_key*).

**IdleTimeOut**

Sets idle timeout limit to time in seconds (*s* or nothing after the number), in minutes (*m*), in hours (*h*), in days (*d*), or in weeks (*w*). If the connection has been idle (all channels) for that length of time, the child process is killed with SIGHUP, and the connection is closed down.

**IgnoreRhosts**

Specifies that rhosts and shosts files will not be used in authentication. */etc/hosts.equiv* and */etc/shosts.equiv* are still used. The default is *no*.

<b>KeepAlive</b>	<p>Specifies whether the system should send keepalive messages to the other side. If they are sent, death of the connection or crash of one of the machines will be properly noticed. However, this means that connections will die if the route is down temporarily (which may be annoying). On the other hand, if keepalives are not sent, sessions may hang indefinitely on the server, leaving 'ghost' users and consuming server resources.</p> <p>The default is 'yes' (to send keepalives), and the server will notice if the network goes down or the client host reboots. This avoids infinitely hanging sessions.</p> <p>To disable keepalives, the value should be set to 'no' in both the server and the client configuration files.</p>
<b>KeyRegenerationInterval</b>	<p>The server key is automatically regenerated after this many seconds (if it has been used). The purpose of regeneration is to prevent decrypting captured sessions by later breaking into the machine and stealing the keys. The key is never stored anywhere. If the value is 0, the key is never regenerated. The default is 3600 (seconds).</p>
<b>ListenAddress</b>	<p>Specifies the ip address of the interface where the sshd server socket is bind.</p>
<b>LoginGraceTime</b>	<p>The server disconnects after this time if the user has not successfully logged in. If the value is 0, there is no time limit. The default is 600 (seconds).</p>
<b>PasswordAuthentication</b>	<p>Specifies whether password authentication is allowed. The default is 'yes'.</p>
<b>PermitEmptyPasswords</b>	<p>When password authentication is allowed, it specifies whether the server allows login to accounts with empty password strings. The default is 'yes'.</p>
<b>PermitRootLogin</b>	<p>Specifies whether the root can log in using <b>ssh</b>. May be set to 'yes', 'nopwd', or 'no'. The default is 'yes', allowing root logins through any of the</p>

authentication types allowed for other users. The 'nopwd' value disables password-authenticated root logins. The 'no' value disables root logins through any of the authentication methods. ('nopwd' and 'no' are equivalent unless you have a .rhosts, .shosts, or .ssh/authorized\_keys file in the root home directory.)

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).

<b>PidFile</b>	Specifies the location of the file containing the process ID of the master sshd daemon (default: /etc/sshd.pid or /var/run/sshd.pid, depending on the system).
<b>Port</b>	Specifies the port number that sshd listens on. The default is 22.
<b>PrintMotd</b>	Specifies whether <b>sshd</b> should print <i>/etc/motd</i> when a user logs in interactively. (On some systems it is also printed by the shell, <i>/etc/profile</i> , or equivalent.) The default is 'yes'.
<b>QuietMode</b>	Specifies whether the system runs in quiet mode. In quiet mode, nothing is logged in the system log, except fatal errors. The default is 'no'.
<b>RandomSeed</b>	Specifies the file containing the random seed for the server; this file is created automatically and updated regularly. The default is <i>/etc/ssh_ran_dom_seed</i> .
<b>RhostsAuthentication</b>	Specifies whether authentication using rhosts or <i>/etc/hosts.equiv</i> files is sufficient. Normally, this method should not be permitted because it is insecure. RhostsRSAAuthentication should be used instead, because it performs RSA-based host authentication in addition to normal rhosts or <i>/etc/hosts.equiv</i> authentication. The default is 'no'.

<b>RhostsRSAAuthentication</b>	Specifies whether rhosts or /etc/hosts.equiv authentication together with successful RSA host authentication is allowed. The default is 'yes'.
<b>RSAAuthentication</b>	Specifies whether pure RSA authentication is allowed. The default is 'yes'.
<b>SilentDeny</b>	Specifies whether denied (or not allowed) connections are denied silently (closing the connection with no logging, etc.) or closed cleanly (sending an error message and log connection attempt).
<b>ServerKeyBits</b>	Defines the number of bits in the server key. The minimum value is 512. The default is 768.
<b>StrictModes</b>	Specifies whether ssh should check file modes and ownership of the user's home directory and rhosts files before accepting login. This is normally desirable because novices sometimes accidentally leave their directory or files world-writable. The default is 'yes'.
<b>SyslogFacility</b>	Gives the facility code that is used when logging messages from <b>sshd</b> . The possible values are: DAEMON, USER, AUTH, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7. The default is DAEMON.
<b>TISAuthentication</b>	Specifies whether authentication through TIS <b>authsrv (8)</b> is allowed. The default is 'no'.
<b>Umask</b>	Sets default umask for sshd and its childs. Remember to add 0 in front of the number to make it octal. Default is to not set umask at all.
<b>X11Forwarding</b>	Specifies whether X11 forwarding is permitted. The default is 'yes'. Note that disabling X11 forwarding does not improve security in any way, as users can always install their own forwarders.
<b>X11DisplayOffset</b>	Specifies the first display number available for sshd's X11 forwarding. This prevents sshd from interfering with real X11 servers.

<b>AllowTCPForwarding</b>	Specifies whether tcp forwarding is permitted. The default is 'yes'. Note that disabling tcp forwarding does not improve security in any way, as users can always install their own forwarders.
<b>KerberosAuthentication</b>	Specifies whether Kerberos V5 authentication is allowed. This can be in the form of a Kerberos ticket, or if PasswordAuthentication is yes, the password provided by the user will be validated through the Kerberos KDC or DCE Security Server. Default is 'yes'.
<b>KerberosOrLocalPasswd</b>	If set, then if password authentication through Kerberos fails then the password will be validated via any additional local mechanism, such as /etc/passwd or SecurID. Default is 'no'.
<b>KerberosTgtPassing</b>	Specifies whether a Kerberos V5 TGT may be forwarded to the server. Default is 'yes'.

## Authorized\_keys File Format

The *\$HOME/.ssh/authorized\_keys* file lists the RSA keys that are permitted for RSA authentication. Each line of the file contains one key (empty lines and lines starting with a '#' are ignored as comments). Each line consists of the following fields, separated by spaces: options, bits, exponent, modulus, comment. The options field is optional; its presence is determined by whether the line starts with a number or not (the option field never starts with a number). The bits, exponent, modulus and comment fields give the RSA key; the comment field is not used for anything (but may be convenient for the user to identify the key).

Note that lines in this file are usually several hundred bytes long (because of the size of the RSA key modulus). Do not type them in. Instead, copy the identity.pub file and edit it.

The options (if present) consists of comma-separated option specifications. No spaces are permitted, except within double quotes.

The following option specifications are supported:

- from='pattern-list'** Specifies that in addition to RSA authentication, the canonical name of the remote host must be present in the comma-separated list of patterns ('\*' and '?' serve as wildcards). The list may also contain patterns negated by prefixing them with '!'; if the canonical host name matches a negated pattern, the key is not accepted. The purpose of this option is to optionally increase security: RSA authentication by itself does not trust the network or name servers or anything (but the key); however, if somebody somehow steals the key, the key permits an intruder to log in from anywhere in the world. This additional option makes using a stolen key more difficult (name servers and/or routers would have to be compromised in addition to just the key).
- command='command'** Specifies that the command is executed whenever this key is used for authentication. The command supplied by the user (if any) is ignored. The command is run on a pty if the connection requests a pty; otherwise it is run without a tty. A quote may be included in the command by quoting it with a backslash. This option might be useful to restrict certain RSA keys to perform just a specific operation. An example might be a key that permits remote backups, but nothing else. Notice that the client may specify TCP/IP and/or X11 forwardings unless they are explicitly prohibited.
- environment='NAME=value'** Specifies that the string is to be added to the environment when logging in using this key. Environment variables set this way override other default environment values. Multiple options of this type are permitted.

<b>Idle-timeout=time</b>	Sets idle timeout limit to time in seconds ( <i>s</i> or nothing after number), in minutes ( <i>m</i> ), in hours ( <i>h</i> ), in days ( <i>d</i> ), or in weeks ( <i>w</i> ). If the connection have been idle (all channels) for that long time the child process is killed with SIGHUP, and connection is closed down.
<b>no-port-forwarding</b>	Forbids TCP/IP forwarding when this key is used for authentication. Any port forward requests by the client will return an error. This might be used, for example, in connection with the <b>command</b> option.
<b>no-X11-forwarding</b>	Forbids X11 forwarding when this key is used for authentication. Any X11 forward requests by the client will return an error.
<b>no-agent-forwarding</b>	Forbids authentication agent forwarding when this key is used for authentication.
<b>no-pty</b>	Prevents tty allocation (a request to allocate a pty will fail).

Examples:

```
1024 33 12121...312314325 ylo@foo.bar
from='*.niksula.hut.fi,!pc.niksula.hut.fi' 1024 35
23...2334 ylo@niksula command='dump /home',no-pty,no-
port-forwarding 1024 33 23...2323 backup.hut.fi
```

## ssh\_known\_hosts File Format

The */etc/ssh\_known\_hosts* and *\$HOME/.ssh/known\_hosts* files contain host public keys for all known hosts. The global file should be prepared by the administrator (optional), and the per-user file is maintained automatically: whenever the user connects an unknown host its key is added to the per-user file. The recommended way to create */etc/ssh\_known\_hosts* is to use the **make-ssh-known-hosts** command.

Each line in these files contains the following fields: hostnames, bits, exponent, modulus, comment. The fields are separated by spaces.

Hostnames is a comma-separated list of patterns ('\*' and '?' act as wildcards); each pattern in turn is matched against the canonical host name (when authenticating a client) or against the user-supplied name (when authenticating a server). A pattern may also be preceded by '!' to indicate negation: if the host name matches a negated pattern, it is not accepted (by that line) even if it matched another pattern on the line.

Bits, exponent, and modulus are taken directly from the host key; they can be obtained, for example, from */etc/ssh\_host\_key.pub*. The optional comment field continues to the end of the line, and is not used.

Lines starting with '#' and empty lines are ignored as comments.

When performing host authentication, authentication is accepted if any matching line has the proper key. It is thus permissible (but not recommended) to have several lines or different host keys for the same names. This will inevitably happen when short forms of host names from different domains are put in the file. It is possible that the files contain conflicting information; authentication is accepted if valid information can be found from either file.

Note that the lines in these files are typically hundreds of characters long, and you should not type in the host keys by hand. Rather, generate them by a script (see **make-ssh-known-hosts** (1)) or by taking */etc/ssh\_host\_key.pub* and adding the host names at the front.

Examples:

```
closenet,closenet.hut.fi,...,130.233.208.41 1024 37  
159...93 closenet.hut.fi
```

## Files

<b><i>/etc/sshd_config</i></b>	Contains configuration data for <b>sshd</b> . This file should be writable by root only, but it is recommended (though not necessary) that it be world-readable.
<b><i>/etc/ssh_host_key</i></b>	Contains the private part of the host key. This file is normally created automatically by 'make install', but can also be created manually using <code>ssh-keygen(1)</code> . This file should only be owned by root, readable only by root, and not accessible to others.
<b><i>/etc/ssh_host_key.pub</i></b>	Contains the public part of the host key. This file is normally created automatically by 'make install', but can also be created manually. This file should be world-readable but writable only by root. Its contents should match the private part. This file is not really used for anything; it is only provided for the convenience of the user so its contents can be copied to known hosts files.
<b><i>/etc/ssh_random_seed</i></b>	This file contains a seed for the random number generator. This file should only be accessible by root.
<b><i>/var/run/sshd.pid</i></b>	Contains the process id of the <b>sshd</b> listening for connections (if there are several daemons running concurrently for different ports, this contains the pid of the one started last). The contents of this file are not sensitive; it can be world-readable.

**\$HOME/.ssh/authorized\_keys** Lists the RSA keys that can be used to log into the user's account. This file must be readable by root (which may on some machines imply it being world-readable if the user's home directory resides on an NFS volume). It is recommended that it not be accessible by others. The format of this file is described above.

*/etc/ssh\_known\_hosts* and *\$HOME/.ssh/known\_hosts*. These files are consulted when using rhosts with RSA host authentication to check the public key of the host. The key must be listed in one of these files to be accepted. (The client uses the same files to verify that the remote host is the one we intended to connect.) These files should be writable only by root/the owner. */etc/ssh\_known\_hosts* should be world-readable, and *\$HOME/.ssh/known\_hosts* can but need not be world-readable.

***/etc/nologin*** If this file exists, **sshd** refuses to let anyone except root log in. The contents of the file are displayed to anyone trying to log in, and non-root connections are refused. The file should be world-readable.

***\$HOME/.rhosts*** This file contains host-username pairs, separated by a space, one per line. The given user on the corresponding host is permitted to log in without password. The same file is used by rlogind and rshd. **Ssh** differs from rlogind and rshd in that it requires RSA host authentication in addition to validating the host name retrieved from domain name servers (unless compiled with the `--with-rhosts` configuration option). The file must be writable only by the user; it is recommended that it not be accessible by others.

If is also possible to use netgroups in the file. Either host or user name may be of the form `+@groupname` to specify all hosts or all users in the group.

**\$HOME/.shosts**

For **ssh**, this file is exactly the same as for `.rhosts`. However, this file is not used by `rlogin` and `rshd`, so using this permits access using **ssh** only.

`/etc/hosts.equiv` This file is used during `.rhosts` authentication. In the simplest form, this file contains host names, one per line. Users on those hosts are permitted to log in without a password, provided they have the same user name on both machines. The host name may also be followed by a user name; such users are permitted to log in as **any** user on this machine (except root). Additionally, the syntax `+@group` can be used to specify netgroups. Negated entries start with `'-'`.

If the client host/user is successfully matched in this file, login is automatically permitted provided the client and server user names are the same. Additionally, successful RSA host authentication is normally required. This file must be writable only by root; it is recommended that it be world-readable.

Warning: It is almost never a good idea to use user **names in hosts.equiv**. Beware that it really means that the named user can log in as **anybody**, which includes `bin`, `daemon`, `adm`, and other accounts that own critical binaries and directories. Using a user name practically grants the user root access. Probably the only valid use for user names is in negative entries. **This warning also applies to rsh/rlogin.**

**`/etc/shosts.equiv`**

This is processed exactly as `/etc/hosts.equiv`. However, this file may be useful in environments that want to run both `rsh/rlogin` and **ssh**.

- /etc/environment** This file is read into the environment at login (if it exists). It can only contain empty lines, comment lines (starting with '#'), and assignment lines of the form name=value. This file is processed in all environments. (Normal rsh/rlogin only process it on AIX and potentially some other systems.) The file should be writable only by root, and should be world-readable.
- \$HOME/.ssh/environment** This file is read into the environment after /etc/environment. It has the same format. The file should be writable only by the user; it need not be readable by anyone else.
- \$HOME/.ssh/rc** If this file exists, it is run with the user's shell after reading the environment files but before starting the user's shell or command. If X11 spoofing is in use, this will receive the 'proto cookie' pair in standard input (and DISPLAY in environment). This must call xauth in that case.
- The primary purpose of this file is to run any initialization routines which may be needed before the user's home directory becomes accessible; AFS is a particular example of such an environment.
- This file will probably contain some initialization code followed by something similar to: 'if read proto cookie; then echo add \$DISPLAY \$proto \$cookie | xauth -q -; fi'.
- If this file does not exist, /etc/sshr is run, and if that does not exist either, xauth is used to store the cookie.
- This file should be writable only by the user, and need not be readable by anyone else.
- /etc/sshr** Like \$HOME/.ssh/rc, but run with /bin/sh. This can be used to specify machine-specific login-time initializations globally. This file should be writable only by root, and should be world-readable.

## 6.11 Troubleshooting

### Common Problems

#### Shadow passwords

Many different shadow password schemes exist. SSH recognizes and supports many of them, but probably not all of them. You may not see any problem at compile time. However, you can suspect that SSH may not be recognizing the shadow password scheme on your system if password authentication does not work even when SSHD is running as root. If that happens, please contact the author. We welcome submissions of code to recognize and use the shadow password mechanism on new systems (`configure.in`, and `auth-passwd.c`, respectively).

`login.c` does not compile, or logging of logins does not work properly.

Mechanisms for updating `wtmp`, `utmp`, `lastlog`, and similar mechanisms are not standardized. SSH substitutes many of the functions of the conventional login program. These functions are implemented in `login.c`. You may need to modify this file to make it work on exotic systems. Please send any modifications and bug fixes back to the author for inclusion in the distribution. If you just want to try SSH, and cannot get this file to compile, it is safe to define all of the functions as empty. In that case, however, logins will not be logged.

SSHD does not start, or dies immediately.

The easiest solution is to use the `-d` option. SSHD will send debugging output to `stderr` and `syslog`. The `-d` option has other effects; for example, the daemon will not fork and will only serve a single connection before exiting. However, it is very useful for debugging problems.

SSHD sends debugging output to the system log.

Check your system log (and `syslogd` configuration) to see why it dies. It is possible your system does not have a proper host key in `/etc/ssh_host_key`. You can either generate a key with `ssh-keygen` (it is automatically generated by “make install”), or specify an alternative key with the `-h` option to the server. It is also possible that the port which the server tries to listen to is already reserved by some other program.

Rhosts authentication does not work.

By default, the server does not accept normal `.rhosts` or `/etc/hosts.equiv` authentication, because they are basically insecure and can be spoofed by anyone with access to the local network. Rhosts authentication can be enabled at compile time by giving the `--with-rhosts` option to configure.

The best alternative is to gather the public host keys of the entire site into a file, and copy the file to `/etc/ssh_known_hosts` on every machine in the organization running SSHD. This will prevent all IP spoofing attacks and provides improved security (provided that `rshd`, `rlogind`, and `rexecd` are disabled).

Opening connections are too slow.

On very slow machines, encrypting and decrypting the session key may be too slow. For example, on a heavily loaded Sun3 it took several minutes to log in with the default key sizes. When we changed it to use a shorter host key (512 bits) and

server key (384 bits), the login time dropped to about one second. A symptom of this problem is that "ssh -v hostname" waits for a long time after printing "Sent encrypted session key".

Shorter host keys can be generated with "ssh-keygen -b 512", giving /etc/ssh\_host\_key as the file in which to save the key (with an empty passphrase). The server key size can be specified with the -b option on the SSHD command line, typically in /etc/rc.local. The server must be restarted for changes to take effect.

The program reports "Could not set controlling tty", or something similar.

There are many different styles of pseudo ttys. SSH currently supports five different styles, plus variations. It is possible that there are more variations, some of which are not supported by existing code. Fixing the problem may require adding new code in pty.c and configure.in. You are encouraged to write the needed code and send a patch to the author, or at least report the problem.

General problem solving      The client has -v option, which sends verbose output to stdout. It is very helpful in solving problems.

The server has -d option, which sends verbose debugging output to system log and stderr. This option also causes the server to serve only a single connection and prevents forking. This helps debugging.



# 7. F-Secure SSH 1.0.2 for Macintosh

## 7.1 Overview

This section only contains installation instructions for the Macintosh client. The client is identical to the F-Secure SSH 1 client for Windows. See Chapter 5 for information about using F-Secure SSH 1 for Macintosh.

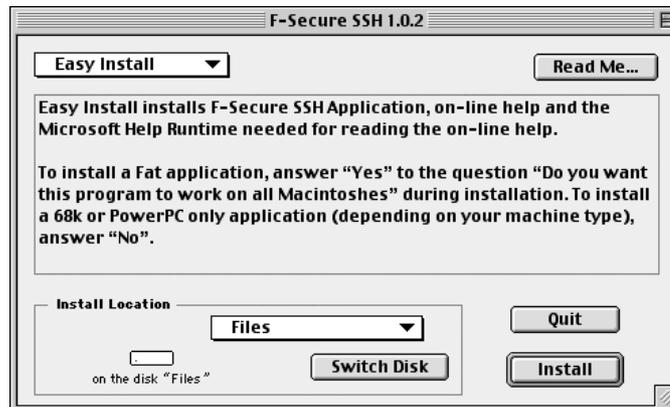
## 7.2 Installation

To install F-Secure SSH 1.0.2 for Macintosh, do the following steps:

1. Insert the F-Secure CD-ROM in the CD-ROM drive.
2. Open the F-Secure CD in a window by double-clicking on its icon.
3. Double-click on the F-Secure SSH 1.0.2 icon on the CD.
4. Click the **Continue** button in the first screen of the Installer.



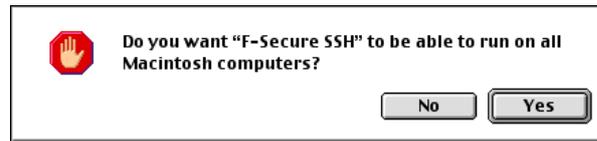
5. You will see the *Readme* document. You can print it out or save it as a file. When you are ready, click **Continue**.
6. In the next window, you can select either Easy Install or Custom Install from the list box. For Easy Install, select the Install Location and click **Install**. To read the Readme document again, click **Read Me**. To quit the installation, click **Quit**.



7. For Custom Installation, select which of the two components you want to install, select the Install Location and click **Install**. To read the Readme document again, click **Read Me**. To quit the installation, click **Quit**.



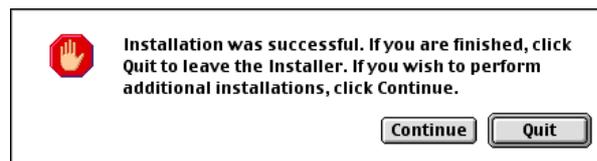
8. Next, you will be asked if you want to enable F-Secure SSH 1 to run on all Macintosh computers. To install as a Fat application, answer 'Yes'. To install as a 68k or PowerPC-only application, answer 'No'.



9. Next, the program will be installed. You can abort the installation at any time by clicking **Stop**.



10. After the installation, the success of the installation will be confirmed. To quit the Installer, click **Quit**. To return to the Installer, click **Continue**.



# Technical Support

Data Fellows Technical Support is available by e-mail or from our Web site. You can access our Web site from within F-Secure FileCrypto or from your Web browser.

## Web Club

One of the most convenient ways to reach Data Fellows for information and technical support is through the World Wide Web.

You can connect directly to our Web site at the following locations:

*<http://www.DataFellows.com>*

*<http://www.Europe.DataFellows.com>*

The F-Secure Support Center can be found at:

*<http://www.DataFellows.com/support/>*

If you are using F-Secure SSH 3.0 Client for Windows, you can connect to the F-Secure Cryptography Web Club from the program's Help menu. Follow these steps:

1. From the Help menu, choose Web Club.
2. In the Select Web Browser dialog box, enter the path and filename for the Web browser you would like to use to access the Data Fellows Web server.
3. Select the closest Data Fellows mirror site.
4. Click the **OK** button to connect.

## Electronic Mail Support

If you have any questions about F-Secure that are not covered in the manual or on-line services at *www.DataFellows.com*, you can contact your local F-Secure distributor or Data Fellows directly.

For basic technical assistance, please contact your F-Secure distributor. If there is no authorized F-Secure Business Partner in your country, you can request basic technical assistance from:

*F-Secure-SSH-Support@DataFellows.com*

Please include the following information along with your support request:

1. Name and version number of your F-Secure software program (including the build number).
2. Name and version number of your operating system (including the build number).
3. A detailed description of the problem, including any error messages displayed by the program, and any other details that might help us duplicate the problem.

When contacting Data Fellows support by telephone, please do the following so that we may help you more effectively and save time:

- Be at your computer so you can follow instructions given by the support technician, or be prepared to write down instructions.
- Have your computer turned on and (if possible) in the state it was in when the problem occurred. Or you should be ready to replicate the problem on the computer with minimum effort.

After installing the F-Secure software, you may find a README file in the following locations:

- Windows: F-Secure SSH folder in the Windows Start > Programs menu.
- Macintosh: F-Secure CD-ROM ("Documentation" folder).
- UNIX: F-Secure CD-ROM (UNIX directory).

The README file contains the latest information.

## About Data Fellows

Data Fellows is one of the world's leading developers of data security products. The company develops, markets and supports anti-virus, data security and cryptography software products for corporate computer networks. It has offices in San Jose, California, and Espoo, Finland, with corporate partners, VARs and other distributors in over 80 countries around the world. Its products have been translated into more than 20 languages.

Data Fellows software products have received numerous international awards and citations. The company was named one of the Top 100 Technology companies in the world by Red Herring magazine in its September 1998 issue. F-Secure Workstation Suite 4.0 was awarded five stars (highest rating) by SECURE Computing Magazine in its July 1999 issue. F-Secure Anti-Virus was awarded Editor's Choice by PC Professionell Magazine (German) in its July 1999 issue. Other commendations include Hot Product of the Year 1997 (Data Communications Magazine), Best Anti-Virus product (SVM Magazine, May 1997), and the 1996 European Information Technology Prize.

Data Fellows has tens of thousands of customers in more than 100 countries. These include many of the world's largest industrial corporations and best-known telecommunications companies; major international airlines; several European governments, post offices and defense forces; and several of the world's largest banks. Customers include NASA, the US Air Force, the US Department of Defense Medical branch, the US Naval Warfare Center, the San Diego Supercomputer Center, Lawrence-Livermore National Laboratory, IBM, Unisys, Siemens-Nixdorf, EDS, Cisco, Nokia, Sonera (formerly Telecom Finland), UUNet Technologies, Boeing, Bell Atlantic, and MCI.

## The F-Secure Product Family

All F-Secure products are integrated into the **F-Secure Framework**, which provides a three-tier, scalable, policy-based management infrastructure for minimizing the cost of security management.

**F-Secure Workstation Suite** consists of malicious code detection and removal, unobtrusive file and network encryption, and personal firewall functionality, all integrated into a policy-based management architecture.

**F-Secure Anti-Virus**, with multiple scanning engines (including F-PROT and AVP), is the most comprehensive, real-time virus scanning and protection system for all Windows platforms. It is a three-tiered solution aimed at the corporate market, and includes a wealth of features for network management and centralized deployment.

**F-Secure VPN+** provides a software-based, IPSec-compliant Virtual Private Network solution for large corporate networks as well as remote and small office networks. By combining F-Secure VPN+ products, companies of any size can use cost-effective public networks, or the Internet, to create secure VPNs without the need to install special hardware.

**F-Secure FileCrypto** is the first and only product to integrate strong real-time encryption directly into the Windows file system. It automatically encrypts data before being stored on the hard disk, protecting sensitive information in the most demanding situations. FileCrypto also allows users to send encrypted, self-extracting packages by e-mail to other users.

**F-Secure SSH** provides secure remote login, terminal, and other connections over unsecured networks. It is the most widely used secure remote administration tool.

**F-Secure NameSurfer** is the solution for remote Internet and Intranet DNS administration. Its easy-to-use WWW user interface automates and simplifies DNS administration.