

# Secure Shell

---

## Installation and Administration

**September 2001**

**Product Version:** Secure Shell, Version 1.0

**Operating System and Version:** Tru64 UNIX Version 5.1A or higher

This manual describes how to install, configure, and use the Secure Shell software.

---

© 2001 Compaq Computer Corporation

Compaq, the Compaq logo, Tru64, and TruCluster are trademarks of Compaq Information Technologies Group, L.P.

UNIX is a trademark of The Open Group.

Contains SSH Secure Shell technology, SSH is a registered trademark of SSH Communications Security Corp. (<http://www.ssh.com>)

All other product names mentioned herein may be trademarks of their respective companies.

Confidential computer software. Valid license from Compaq required for possession, use, or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Compaq shall not be liable for technical or editorial errors or omissions contained herein. The information in this document is provided "as is" without warranty of any kind and is subject to change without notice. The warranties for Compaq products are set forth in the express limited warranty statements accompanying such products. Nothing herein should be construed as constituting an additional warranty.

---

# Contents

## About This Manual

### 1 Secure Shell Overview

1.1	The Secure Shell Server .....	1-2
1.2	The Secure Shell Client .....	1-2
1.3	Server and Client Communication .....	1-2

### 2 Installing the Secure Shell Software

2.1	Downloading the Secure Shell Software .....	2-1
2.2	Installing the Secure Shell Software .....	2-2
2.3	Installing and Viewing the Secure Shell Documentation .....	2-2
2.4	Starting the Server Software .....	2-2
2.5	Uninstalling the Secure Shell Software .....	2-3
2.6	Removing the Secure Shell Documentation .....	2-3
2.7	Secure Shell Directories and Files .....	2-3

### 3 Configuring and Managing the Secure Shell Software

3.1	Configuring the Server .....	3-1
3.2	Configuring the Client .....	3-7
3.2.1	Configuring r* Commands to Use Secure Shell .....	3-10
3.3	User Authentication Methods .....	3-11
3.3.1	Password Authentication Method .....	3-11
3.3.2	Public Key Authentication Method .....	3-12
3.3.2.1	Configuring Public Key Authentication on the Client ..	3-12
3.3.2.2	Configuring Public Key Authentication on the Server ..	3-13
3.3.2.3	Logging In to the Server Using Public Key Authentication .....	3-14
3.3.2.4	Managing Passphrases .....	3-14
3.3.3	Host-Based Authentication Method .....	3-15
3.3.3.1	Configuring Host-Based Authentication .....	3-15
3.3.3.2	Connecting Using Host-Based Authentication .....	3-17
3.4	Managing the Server .....	3-17
3.4.1	Starting, Stopping, and Resetting the sshd2 Daemon .....	3-17
3.4.2	Generating A Host Key .....	3-18

3.4.3	Forwarding TCP/IP and X11 Data Through a Secure Shell Connection .....	3-18
3.4.3.1	TCP/IP Port Forwarding .....	3-18
3.4.3.2	X11 Forwarding .....	3-19

## 4 Using the Secure Shell Commands

4.1	Copying Files .....	4-1
4.1.1	Using the scp2 Command .....	4-1
4.1.2	Using the sftp2 Command .....	4-2
4.2	Logging In and Executing Commands on a Server .....	4-2

## Index

### Examples

3-1	Sample sshd2_config File .....	3-4
3-2	Sample ssh2_config File .....	3-8
3-3	Public Key Authentication Login Output .....	3-14

### Tables

1-1	Secure Shell Commands .....	1-1
3-1	sshd2_config Sections .....	3-2
3-2	ssh2_config Sections .....	3-7

---

## About This Manual

This manual describes how to install, configure, and use the Secure Shell software on a system running the Compaq Tru64™ UNIX operating system software.

### Audience

This manual is intended for anyone who is responsible for installing and configuring the Secure Shell sever and client software on a system running the Tru64 UNIX operating system software.

### Organization

The manual is organized as follows:

- Chapter 1* Provides an overview of Secure Shell client and server software.
- Chapter 2* Describes how to install the Secure Shell server software.
- Chapter 3* Describes how to configure and manage the Secure Shell server and client software.
- Chapter 4* Describes how to use Secure Shell commands.

### Reader's Comments

Compaq welcomes any comments and suggestions you have on this and other Tru64 UNIX manuals.

You can send your comments in the following ways:

- Fax: 603-884-0120 Attn: UBPG Publications, ZKO3-3/Y32
- Internet electronic mail: [readers\\_comment@zk3.dec.com](mailto:readers_comment@zk3.dec.com)

A Reader's Comment form is located on your system in the following location:

```
/usr/doc/readers_comment.txt
```

Please include the following information along with your comments:

- The full title of the manual and the order number. (The order number appears on the title page of printed and PDF versions of a manual.)

- The section numbers and page numbers of the information on which you are commenting.
- The version of Tru64 UNIX that you are using.
- If known, the type of processor that is running the Tru64 UNIX software.

The Tru64 UNIX Publications group cannot respond to system problems or technical support inquiries. Please address technical questions to your local system vendor or to the appropriate Compaq technical support office. Information provided with the software media explains how to send problem reports to Compaq.

## Conventions

This manual uses the following typographical conventions:

\	A backslash at the end of a line in an example indicates continuation.
#	A number sign represents the system prompt when you are logged in to a Tru64 UNIX system using the root user account.
<b>net stop</b>	Bold courier type indicates user input.
>>>	The console subsystem prompt is three right angle brackets.
<i>file</i>	Italic (slanted) type indicates variable values, placeholders, and function argument names.
[   ]	In syntax definitions, brackets indicate items that are optional and braces indicate items that are required. Vertical bars separating items inside brackets or braces indicate that you choose one item from among those listed.
{   }	
. . .	In syntax definitions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.
cat(1)	A cross-reference to a reference page includes the appropriate section number in parentheses. For example, cat(1) indicates that you can find information on the cat command in Section 1 of the reference pages.
[Ctrl/x]	This symbol indicates that you hold down the first named key while pressing the key or mouse button that follows the slash. In examples, this key combination is enclosed in a box (for example, [Ctrl/C] ).

---

## Secure Shell Overview

The Secure Shell software is a client/server software application that provides a suite of secure network commands that can be used in addition to or in place of traditional nonsecure network commands (sometimes referred to as the *r\** commands). Table 1–1 describes the traditional nonsecure network commands and the equivalent Secure Shell commands.

**Table 1–1: Secure Shell Commands**

To	Traditional Command	Secure Shell Command
Remotely execute commands on a computer	rsh	ssh2
Remotely log in to a computer	rlogin or telnet	ssh2
Transfer files	rcp or ftp	scp2 or sftp2

See Chapter 4 for more information on using the Secure Shell commands.

Using Secure Shell commands provides secure network communications between systems running the Secure Shell client software and systems running the Secure Shell server software through the use of:

- **Authentication**  
Secure Shell servers and clients use an authentication method to reliably determine each other's identity, then the user's identity.
- **Data encryption**  
Secure Shell servers and clients exchange encrypted data. Data encryption is transparent to users.
- **Data integrity**  
Secure Shell servers and clients detect whether or not data was intercepted and modified while in transit.

This chapter describes:

- Secure Shell servers
- Secure Shell clients
- Secure Shell server and client communication

## 1.1 The Secure Shell Server

A Secure Shell server is a system on which the Secure Shell server software is installed and the Secure Shell `sshd2` daemon is started. The Compaq Secure Shell software includes the Secure Shell server software that runs on a system running the Tru64 UNIX Version 5.1A or higher operating system software. The Compaq Secure Shell Version 1.0 software is based on SSH Version 2.4.1. software.

The remainder of this guide refers to a Secure Shell server as server.

## 1.2 The Secure Shell Client

A Secure Shell client is a system on which the Secure Shell client software is installed. The Compaq Secure Shell software includes the Secure Shell client software that runs on a system running the Tru64 UNIX Version 5.1A or higher operating system software.

The Secure Shell client software provides:

- The `scp2` and `sftp2` commands to copy files to and from a server.
- The `ssh2` command to log in and execute commands on a server.
- Other Secure Shell commands to manage the Secure Shell client software.

You can configure the Secure Shell client to automatically and transparently use a Secure Shell connection to secure `r*` commands. For example, when a user enters an `r*` command, such as the `rcp` command, the command will automatically use a Secure Shell connection transparent to the user. See the Secure `r`-utilities section in Table 3–2 for more information on securing `r*` commands.

The remainder of this guide refers to a Secure Shell client as client.

## 1.3 Server and Client Communication

When the server is started, the `sshd2` daemon listens on port 22 (by default) for a client to initiate a socket connection. Once connected, the `sshd2` daemon starts a child process that initiates a public host key exchange with the client. The public host key exchange is a process in which the server proves its identity to the client.

Each server has a public and private host key pair that was created when the Secure Shell software was installed. The key pair is a cryptographic hash which uniquely identifies the server. The server's private host key is in a file called `/etc/ssh2/hostkey`. The server's public host key is in a file called `/etc/ssh2/hostkey.pub`.

When a user connects to a new server, the user is (by default) prompted to accept a copy of the server's public host key. If the user accepts the key, a copy of the server's public host key is copied to the user's `hostkeys` directory. The client uses this public key to authenticate the server on subsequent connections.

After the child process has proven the server's identity, the actual connection with the client occurs. This includes cipher negotiation, user authentication, command execution, encrypted data transfer, and termination of the connection. Once the connection has been terminated, the child process started by the `sshd2` daemon terminates.



# 2

---

## Installing the Secure Shell Software

The Tru64 UNIX Secure Shell software provides the software to run the server and client on a system running the Tru64 UNIX software.

This chapter describes:

- How to download the Secure Shell software
- How to install the Secure Shell software
- How to install and view the Secure Shell documentation
- How to start the server
- How to uninstall the Secure Shell software
- How to remove the Secure Shell documentation
- Secure Shell directories and files

### 2.1 Downloading the Secure Shell Software

The Secure Shell software is provided in a compressed tar file that contains:

- A subset called `SSHBASE100` that contains the Secure Shell software.
- A subset called `SSHDOC100` that contains Secure Shell reference pages.
- A tar file called `ssh_ssb_docs.tar.gz` that contains the Secure Shell HTML documentation library.

The Secure Shell software is available from the Compaq Web site at <http://tru64unix.compaq.com/internet>.

To download the Secure Shell software:

1. Create a directory called `/usr/share/doclib/ssh` and change to that directory.
2. Using a Web browser, go to <http://tru64unix.compaq.com/internet> and choose download software.
3. Choose to download the Secure Shell software kit and follow the instructions on the screen.
4. Uncompress the tar file:

```
# gunzip tar_file.tar.gz
```

5. Extract the contents of the tar file:

```
# tar -xvf tar_file.tar
```

## 2.2 Installing the Secure Shell Software

Installing the Secure Shell software:

- Creates the `/etc/ssh2` directory.
- Creates a public and private host key for the server.
- Modifies the `/etc/services` file and the `/etc/clua_services` file in a TruCluster Server environment to reserve port 22 for the `sshd2` daemon.
- Symbolically links the `/etc/ssh2/knownhosts` directory to the `/var/ssh2/knownhosts` directory.

Use the Tru64 UNIX `setld` command to install the subsets that contain the Secure Shell software and reference pages on a system running the Tru64 UNIX Version 5.1A or higher operating system software. To install the subsets:

1. Change to the directory where the Secure Shell software was downloaded.
2. As root, install the subsets:

```
# setld -l . SSHBASE100 SSHDOC100
```

## 2.3 Installing and Viewing the Secure Shell Documentation

To install the Secure Shell documentation:

1. If necessary, change to the directory where you downloaded the Secure Shell software.
2. Uncompress the documentation tar file:

```
# gunzip ssh_ssb_docs.tar.gz
```

3. Extract the contents from the documentation tar file:

```
# tar -xvf ssh_ssb_docs.tar
```

To view the documentation, open the `/usr/share/doc-clib/ssh/DOCS/HTML/LIBRARY.HTM` file in a web browser.

## 2.4 Starting the Server Software

Starting the `sshd2` daemon starts the Secure Shell server software. When the `sshd2` daemon starts, it uses the values assigned to keywords in the

`/etc/ssh2/ssh2_config` file to configure how the server responds to clients.

Review the keywords in the `ssh2_config` file before starting the `ssh2` daemon. See Section 3.1 and `ssh2_config(4)` for more information about the `ssh2_config` file.

To start the `ssh2` daemon, enter:

```
# /sbin/init.d/ssh2 start
```

In a TruCluster Server environment, you must start the `ssh2` daemon on each cluster member on which you want the `ssh2` daemon to run.

The `ssh2` daemon will automatically start when the system boots.

## 2.5 Uninstalling the Secure Shell Software

To uninstall the Secure Shell software, enter:

```
# setld -d SSHBASE100 SSHDOC100
```

The Secure Shell directories and software subsets are left in place.

## 2.6 Removing the Secure Shell Documentation

To remove the Secure Shell documentation, enter:

```
# rm -rf /usr/share/doc/lib/ssh/DOCS
```

## 2.7 Secure Shell Directories and Files

The following files and subdirectories are created in the `/etc/ssh2` directory on the server when you install the Secure Shell software:

- The client configuration file (`ssh2_config`).
- The server configuration file (`ssh2_config`).
- The server private host key file (`hostkey`) and public host key file (`hostkey.pub`).
- A random seed file (`random_seed`). This file contains random data that is used to generate pseudorandom numbers for cryptographic operations.
- Public host key files for the clients with which the server communicates when using host-based authentication are in the `knownhosts` subdirectory.

The `ssh2` daemon is in the `/usr/sbin` directory.

The Secure Shell commands are in the `/usr/bin` directory.



# 3

---

## Configuring and Managing the Secure Shell Software

You configure how servers and clients communicate by setting values to keywords in the `/etc/ssh2/sshd2_config` file for the server and in the `/etc/ssh2/ssh2_config` file for the client.

The `/etc/ssh2/sshd2_config` file and the `/etc/ssh2/ssh2_config` file contain keyword-argument pairs, one per line. Keywords are assigned default values, which you can change. Keywords are case insensitive. Empty lines and lines starting with a number sign ( # ) are ignored as comments.

This chapter describes:

- How to configure the server
- How to configure the client
- User authentication methods
- Managing the server

### 3.1 Configuring the Server

The `/etc/ssh2/sshd2_config` file (or the file specified with `sshd2 -f` command) contains configuration keywords and values that the `sshd2` daemon reads when it starts. You can override the values for a keyword in the `sshd2_config` file by entering the keyword and value on the command line when you start the `sshd2` daemon; however, values set this way are effective only until the `sshd2` daemon restarts.

To modify the `sshd2_config` file while the `sshd2` daemon is running, you must reset the `sshd2` daemon to implement the change. Changes made this way are applicable only to new client connections. See Section 3.4.1 for information on resetting the `sshd2` daemon.

Table 3–1 describes the sections and some of the keywords in the `sshd2_config` file. See `sshd2_config(4)` for a complete list of keywords in the `sshd2_config` file.

**Table 3–1: sshd2\_config Sections**

Section:	Keywords
General	The <code>VerboseMode</code> keyword specifies whether or not debugging messages are displayed.
Network	<ul style="list-style-type: none"><li>• The <code>Port</code> keyword specifies which port the server listens on.</li><li>• The <code>ListenAddress</code> keyword specifies the TCP/IP address of the interface where the <code>sshd2</code> server socket is bound.</li><li>• The <code>RequireReverseMapping</code> keyword specifies whether host name DNS lookup must succeed when checking connections from hosts that are defined by the <code>AllowHosts</code> and <code>DenyHosts</code> keywords.</li><li>• The <code>MaxBroadcastsPerSecond</code> keyword specifies how many UDP broadcasts that the server handles per second.</li></ul>
Crypto	<ul style="list-style-type: none"><li>• The <code>Ciphers</code> keyword specifies the Secure Shell encryption algorithms (ciphers) that the server uses to encrypt data. Supported ciphers are <code>des</code>, <code>3des</code>, <code>blowfish</code>, <code>arcfour</code>, <code>twofish</code>, and <code>cast</code>.</li><li>• The <code>MACs</code> keyword that specifies the Secure Shell Message Authentication Code (MAC) algorithm to use for data integrity verification. Supported MAC algorithms are <code>hmac-sha1</code>, <code>hmac-sha1-96</code>, <code>hmac-md5</code>, <code>hmac-md5-96</code>, <code>hmac-ripemd160</code>, and <code>hmac-ripemd160-96</code>, of which <code>hmac-sha1</code>, <code>hmac-sha1-96</code>, <code>hmac-md5</code> and <code>hmac-md5-96</code> are included.</li></ul>
User	<ul style="list-style-type: none"><li>• The <code>PrintMotd</code> keyword specifies if the <code>/etc/motd</code> file is displayed when a user logs in.</li><li>• The <code>CheckMail</code> keyword specifies if information is displayed when there is new mail when a user logs in.</li><li>• The <code>UserConfigDirectory</code> keyword specifies where to locate user-specific configuration data.</li></ul>
Tunneling	<ul style="list-style-type: none"><li>• The <code>AllowX11Forwarding</code> keyword specifies if X11 forwarding is permitted.</li><li>• The <code>AllowTcpForwarding</code> keyword specifies if TCP/IP forwarding is permitted.</li></ul> <p>See Section 3.4.3 for more information on configuring TCP/IP and X11 forwarding.</p>

**Table 3–1: sshd2\_config Sections (cont.)**

Section:	Keywords
Authenticat- ion	<ul style="list-style-type: none"><li>• The <code>AllowedAuthentications</code> keyword specifies the authentication methods that the client and server use to authenticate users. Supported authentication methods are <code>password</code>, <code>publickey</code>, and <code>hostbased</code>. See Section 3.3 for more information on these authentication methods and how to configure the server and client to use them.</li><li>• The <code>PasswordGuesses</code> keyword specifies the number of failed password tries that the user is permitted when using <code>password</code> authentication.</li></ul>
User public key authen- tication	<ul style="list-style-type: none"><li>• The <code>HostKeyFile</code> keyword specifies the file containing the private host key.</li><li>• The <code>PublicHostKeyFile</code> keyword specifies the file containing the public host key.</li><li>• The <code>IdentityFile</code> keyword specifies the file containing the user public key when using <code>publickey</code> authentication.</li><li>• The <code>AuthorizationFile</code> keyword specifies the name of the user’s authorization file when using <code>publickey</code> authentication.</li></ul> <p>See Section 3.3 for more information on <code>publickey</code> authentication.</p>
Host restrictions	<ul style="list-style-type: none"><li>• The <code>AllowHosts</code> keyword followed by host names specifies the names of hosts from which user login is allowed.</li><li>• The <code>DenyHosts</code> keyword followed by host names specifies the names of hosts from which user login is not allowed.</li><li>• The <code>AllowSHosts</code> keyword followed by host names specifies matching host names to honor in the <code>.shosts</code> file and in the <code>.rhosts</code> files when using host-based authentication.</li><li>• The <code>DenySHosts</code> keyword followed by host names specifies matching host names to ignore in the <code>.shosts</code> file and in the <code>.rhosts</code> files when using host-based authentication.</li></ul> <p>See Section 3.3 for more information on host-based authentication.</p>
User restrictions	<ul style="list-style-type: none"><li>• The <code>AllowUsers</code> keyword followed by user names specifies the user names who are allowed to log in.</li><li>• The <code>DenyUsers</code> keyword followed by user names specifies the user names who are not allowed to log in.</li><li>• The <code>AllowGroups</code> keyword followed by group names specifies that only users who belong to those groups can log in.</li><li>• The <code>DenyGroups</code> keyword followed by group names specifies that users who belong to those groups can not log in.</li></ul>

**Table 3–1: sshd2\_config Sections (cont.)**

Section:	Keywords
SSH1 compatibility	The <code>Ssh1Compatibility</code> keyword specifies whether or not to use SSH1 compatibility mode.
Subsystem definitions	<p>The <code>Subsystem-sftp</code> keyword specifies which subsystems to execute. A subsystem is a set of remote predefined commands on the server. For example, if the server is running on a system called <code>server.example.com</code>, you can enter the following command to backup a file system:</p> <pre># ssh2 server.example.com /bin/tar c /home</pre> <p>Alternatively, you can define a subsystem called <code>backup</code> as <code>Subsystem-backup /bin/tar c /home</code> and achieve the same result as the previous command line by entering:</p> <pre># ssh2 -s backup server.example.com</pre>

Example 3–1 is a sample `/etc/ssh2/sshd2_config` file.

**Example 3–1: Sample sshd2\_config File**

```
## sshd2_config
## SSH 2.4 Server Configuration File
##

## General

  VerboseMode    no
# QuietMode     yes
  AllowCshrcSourcingWithSubsystems no
  ForcePTYAllocation no
  SyslogFacility AUTH
# SyslogFacility LOCAL7

## Network

  Port          22
  ListenAddress 0.0.0.0
  RequireReverseMapping no
  MaxBroadcastsPerSecond 0
# MaxBroadcastsPerSecond 1
# NoDelay      yes
# KeepAlive    yes
# MaxConnections 50
# MaxConnections 0
# 0 == number of connections not limited
```

### Example 3–1: Sample sshd2\_config File (cont.)

---

```
## Crypto

Ciphers      AnyCipher
# Ciphers    AnyStd
# Ciphers    AnyStdCipher
# Ciphers    3des
MACs         AnyMAC
# MACs      AnyStd
# MACs      AnyStdMAC
# RekeyIntervalSeconds 3600

## User

PrintMotd    yes
CheckMail    yes
UserConfigDirectory "%D/.ssh2"
# UserConfigDirectory "/etc/ssh2/auth/%U"
UserKnownHosts yes
# LoginGraceTime 600
# PermitEmptyPasswords no
# StrictModes yes

## User public key authentication

HostKeyFile  hostkey
PublicHostKeyFile hostkey.pub
RandomSeedFile random_seed
IdentityFile identification
AuthorizationFile authorization
AllowAgentForwarding yes

## Tunneling

AllowX11Forwarding yes
AllowTcpForwarding yes
# AllowTcpForwardingForUsers sjl, cowboyneal@slashdot.org
# DenyTcpForwardingForUsers "2[:isdigit:]*4, peelo"
# AllowTcpForwardingForGroups privileged_tcp_forwarders
# DenyTcpForwardingForGroups coming_from_outside

## Authentication
## Hostbased and PAM are not enabled by default.

# BannerMessageFile      /etc/ssh2/ssh_banner_message
```

### Example 3–1: Sample sshd2\_config File (cont.)

---

```
# BannerMessageFile          /etc/issue.net
# PasswordGuesses            3
# AllowedAuthentications     hostbased,publickey,password
# AllowedAuthentications     publickey,pam-1@ssh.com
# AllowedAuthentications     publickey,password
# RequiredAuthentications    publickey,password
# SshPAMClientPath           ssh-pam-client

## Host restrictions

# AllowHosts                  localhost, foobar.com, friendly.org
# DenyHosts                   evil.org, aol.com
# AllowSHosts                  trusted.host.org
# DenySHosts                  not.quite.trusted.org
# IgnoreRhosts                no
# IgnoreRootRHosts           no
# (the above, if not set, is defaulted to the value of IgnoreRHosts)

## User restrictions

# AllowUsers                  "sj*,s[:isdigit:]##,s(jl|amza)"
# DenyUsers                   skuuppa,warezdude,31373
# DenyUsers                   don@untrusted.org
# AllowGroups                  staff,users
# DenyGroups                   guest
# PermitRootLogin              nopwd
# PermitRootLogin              yes

## SSH1 compatibility

# Ssh1Compatibility
# Sshd1Path

## Chrooted environment

# ChRootUsers                  ftp, guest
# ChRootGroups                 guest

## subsystem definitions

subsystem-sftp                 sftp-server
```

---

## 3.2 Configuring the Client

The `/etc/ssh2/ssh2_config` file contains configuration keywords and values that the client software reads when it starts. Each user can also have their own personal `ssh2_config` file in a `$HOME/.ssh2` directory, where `$HOME` is the name of the user's home directory. The `/etc/ssh2/ssh2_config` file is read first, then the user's version is read. The last obtained value for a keyword is used.

Table 3–2 describes the sections and some of the keywords in the `ssh2_config` file. See `ssh2_config(4)` for a complete list of keywords in the `ssh2_config` file.

**Table 3–2: ssh2\_config Sections**

Section	Keywords
Secure r-utilities	<p>The <code>EnforceSecureRutils</code> keyword specifies whether or not to use Secure Shell to automatically secure the <code>r*</code> commands; for example, the <code>rsh</code>, <code>rlogin</code>, and <code>rcp</code> commands.</p> <p>See Section 3.2.1 for considerations about configuring the <code>r*</code> commands to use Secure Shell.</p>
General	<ul style="list-style-type: none"><li>• The <code>VerboseMode</code> keyword specifies whether or not debugging messages are displayed.</li><li>• The <code>PasswordPrompt</code> keyword specifies the password prompt that is displayed.</li><li>• The <code>AuthenticationSuccessMsg</code> keyword specifies whether to display the <code>Authentication</code> successful message after authentication has completed successfully.</li></ul>
Network	<ul style="list-style-type: none"><li>• The <code>Port</code> keyword specifies which port to connect on the server.</li><li>• The <code>KeepAlive</code> keyword specifies whether <code>keepalive</code> messages are sent. If they are sent, the loss of a connection or crash of a system will be noticed.</li></ul>
Crypto	<ul style="list-style-type: none"><li>• The <code>Ciphers</code> keyword specifies the Secure Shell encryption algorithms (ciphers) that the client uses to encrypt data. Supported ciphers are <code>des</code>, <code>3des</code>, <code>blowfish</code>, <code>arcfour</code>, <code>twofish</code>, and <code>cast</code>.</li><li>• The <code>MACs</code> keyword specifies the Secure Shell Message Authentication Code (MAC) algorithm to use for data integrity verification. Supported MAC algorithms are <code>hmac-sha1</code>, <code>hmac-sha1-96</code>, <code>hmac-md5</code>, <code>hmac-md5-96</code>, <code>hmac-ripemd160</code>, and <code>hmac-ripemd160-96</code>, of which <code>hmac-sha1</code>, <code>hmac-sha1-96</code>, <code>hmac-md5</code> and <code>hmac-md5-96</code> are included.</li><li>• The <code>StrictHostKeyChecking</code> keyword specifies whether the client automatically adds new server host keys to the <code>\$HOME/.ssh2/hostkeys</code> directory.</li></ul>

**Table 3–2: ssh2\_config Sections (cont.)**

Section	Keywords
Tunneling	<ul style="list-style-type: none"><li>• The <code>GatewayPorts</code> keyword specifies whether servers can connect to locally forwarded ports.</li><li>• The <code>ForwardX11</code> keyword specifies whether X11 connections are automatically redirected over the secure channel to set the <code>DISPLAY</code> environment variable.</li><li>• The <code>ForwardAgent</code> keyword specifies whether the connection to the authentication agent (if any) will be forwarded to the server when using public key authentication.</li></ul>
Authenticati- on	The <code>AllowedAuthentications</code> keyword specifies the authentication methods that the client and server use to authenticate users. Supported authentication methods are <code>password</code> , <code>publickey</code> , and <code>hostbased</code> . See Section 3.3 for more information on these authentication methods and how to configure the server and client to use them.
User public key authen- tication	<ul style="list-style-type: none"><li>• The <code>IdentityFile</code> keyword specifies the file containing the identity user host key when using <code>publickey</code> authentication.</li><li>• The <code>AuthorizationFile</code> keyword specifies the name of the user's authorization file when using <code>publickey</code> authentication.</li></ul> See Section 3.3 for more information on <code>publickey</code> authentication.
SSH1 com- patibility	The <code>Ssh1Compatibility</code> keyword specifies whether or not to use SSH1 compatibility mode.

Example 3–2 is a sample `/etc/ssh2/ssh2_config` file.

**Example 3–2: Sample ssh2\_config File**

```
## ssh2_config
## SSH 2.0 Client Configuration File
##

## The "*" is used for all hosts, but you can use other hosts as
## well.
* :

## COMPAQ Tru64 UNIX specific
# Secure the r* utilities (no, yes)
    EnforceSecureRutils          no

## General

    VerboseMode    no
# QuietMode      yes
# DontReadStdin  no
```

### Example 3–2: Sample ssh2\_config File (cont.)

---

```
# BatchMode    yes
# Compression  yes
# ForcePTYAllocation  yes
# GoBackground  yes
# EscapeChar    ~
# PasswordPrompt    "%U@%H's password: "
# PasswordPrompt    "%U's password: "
# AuthenticationSuccessMsg yes

## Network

Port    22
NoDelay  no
KeepAlive  yes
# SocksServer    socks://mylogin@socks.ssh.com:1080/203.123.0.0/16,198.74.23.0/24

## Crypto

Ciphers    AnyStdCipher
MACs    AnyMAC
StrictHostKeyChecking    ask
# RekeyIntervalSeconds    3600

## User public key authentication

IdentityFile    identification
AuthorizationFile    authorization
RandomSeedFile    random_seed

## Tunneling

# GatewayPorts    yes
# ForwardX11    yes
# ForwardAgent    yes

# Tunnels that are set up upon logging in

# LocalForward    "110:pop3.ssh.com:110"
# RemoteForward    "3000:foobar:22"

## SSH1 Compatibility
```

### Example 3–2: Sample ssh2\_config File (cont.)

---

```
    Ssh1Compatibility          yes
    Ssh1AgentCompatibility    none
# Ssh1AgentCompatibility    traditional
# Ssh1AgentCompatibility    ssh2
# Ssh1Path    /usr/local/bin/ssh1

## Authentication
## Hostbased is not enabled by default.

# AllowedAuthentications    hostbased,publickey,password
    AllowedAuthentications    publickey,password

# For ssh-signer2 (only effective if set in the global configuration
# file, usually /etc/ssh2/ssh2_config)

# DefaultDomain    foobar.com
# SshSignerPath    ssh-signer2

## Examples of per host configurations

#alpha*:
# Host    alpha.oof.fi
# User    user
# PasswordPrompt    "%U:s password at %H: "
# Ciphers    idea

#foobar:
# Host    foo.bar
# User    foo_user
```

---

#### 3.2.1 Configuring r\* Commands to Use Secure Shell

By default, the `EnforceSecureRutils` keyword is disabled. The `EnforceSecureRutils` keyword requires that the `AllowTcpForwarding` keyword be enabled on the server, which it is by default. If you enable the `EnforceSecureRutils` keyword, do not disable the `AllowTcpForwarding` keyword on the server.

If the `EnforceSecureRutils` keyword is enabled in the `/etc/ssh2/ssh2_config` file, it is enabled for all users regardless

if it is disabled in the `ssh2_config` file in a user's `$HOME/.ssh2` directory. If the `EnforceSecureRutils` keyword is disabled in the `/etc/ssh2/ssh2_config` file, a user can enable it in the `ssh2_config` file in their `$HOME/.ssh2` directory.

If the `EnforceSecureRutils` keyword is enabled, the password prompt will be displayed on `stderr` when using an `r*` command. If `stderr` is redirected, it may appear that the `r*` command has stopped responding while waiting for the password to be entered. In the case of host-based authentication, the `$HOME/.rhosts` file is ignored. Users using host-based authentication must use the `$HOME/.shosts` file. See Section 3.3.3 for more information about host-based authentication.

## 3.3 User Authentication Methods

After the client and server establish a secure connection, the server must authenticate the user. The user must have a valid user account and home directory. The server can authenticate users by using any or all of the following methods:

- Password (allowed by default)
- Public key (allowed by default)
- Host-based

The order in which authentication methods are listed for the `AllowedAuthentications` keyword is the order in which they are used. For example, if `hostbased` is listed first, the server will try `hostbased` authentication before trying the next listed authentication. The first successful authentication is the one used.

### 3.3.1 Password Authentication Method

Password authentication is a method in which the server uses the configured Tru64 UNIX password authentication method to authenticate a user. Tru64 UNIX password authentication methods include:

- BSD
- Kerberos
- Network Information Service (NIS)
- Enhanced security

Password authentication requires that a user have a password-protected user account that can be authenticated by a Tru64 UNIX password authentication method. When a user enters a Secure Shell command to connect to a server that requires password authentication, the user is prompted for a password. The client securely transmits the user name and

password to the server. Using the configured password authentication method; for example BSD, the server looks in the `/etc/passwd` file for a matching user name and password entry. If there is no matching entry, the user is not authenticated. If there is a matching entry, the user is authenticated and proceeds with the connection.

To use password authentication, the value of the `AllowedAuthentications` keyword must include `password`, which is set by default, in the `/etc/ssh2/sshd2_config` file and in the `/etc/ssh2/ssh2_config` file; for example:

```
AllowedAuthentications    password
```

If you change the value of the `AllowedAuthentications` keyword, you must enter the following command to reset the `sshd2` daemon:

```
# /sbin/init.d/sshd reset
```

### 3.3.2 Public Key Authentication Method

Public key authentication is a method in which the user creates a cryptographic public and private key pair and assigns a passphrase (similar to a password) to the keys. A key is a unique string of binary data that creates a digital identity for a user. The private key is stored on the client. The public key is stored on servers to which the user will connect.

When a user enters a Secure Shell command to connect to a server that requires public key authentication, the user is prompted for the passphrase. The client and server use the user's passphrase and public and private keys to exchange encrypted information about the user. If the server validates that the user's private key stored on the client matches a public key stored on the server, the user is authenticated and proceeds with the connection.

Public key authentication requires configuration on the client and on the server.

#### 3.3.2.1 Configuring Public Key Authentication on the Client

To configure public key authentication on a client, either enter the `ssh-pubkeymgr` command and follow the instructions on the screen or follow these steps:

1. Create a key pair by entering the `ssh-keygen2` command. The `ssh-keygen2` command prompts for a passphrase for the key pair.

```
$> ssh-keygen2
Generating 1024-bit dsa key pair
 1 Ooo.oOo.oOo.
Key generated.
1024-bit dsa, user@Local, Wed July 25 2001 17:13:43
```

```
Passphrase :
Again :
Private key saved to
    /home/user/.ssh2/id_dsa_1024_a
Public key saved to
    /home/user/.ssh2/id_dsa_1024_a.pub
```

The `ssh-keygen2` command creates an `$HOME/.ssh2` directory for the user on the client (`$HOME` is the name of the user's home directory) and stores the authentication key pair in two files:

- The `id_dsa_1024_a` file contains the user's private key. Only the user for which the key was created should have access this file.
  - The `id_dsa_1024_a.pub` file contains the user's public key. This file will be copied to servers that use public key authentication and to which the user will connect.
2. In the user's `$HOME/.ssh2` directory, create a file called `identification` that contains the following line that identifies the name of the user's private key file:

```
IdKey id_dsa_1024_a
```

### 3.3.2.2 Configuring Public Key Authentication on the Server

To configure public key authentication on the server:

1. Make sure that the value of the `AllowedAuthentications` keyword includes `publickey`, which is set by default, in the `/etc/ssh2/sshd2_config` file and in the `/etc/ssh2/ssh2_config` file; for example:

```
AllowedAuthentications    publickey
```

If you change the value of the `AllowedAuthentications` keyword, you must enter the following command to reset the `sshd2` daemon:

```
# /sbin/init.d/sshd reset
```

2. If necessary, create a `$HOME/.ssh2` directory on the server (`$HOME` is the name of the user's home directory for which public key authentication is being configured.)
3. Copy the user's public key file (`id_dsa_1024_a.pub`) on the client to the user's `$HOME/.ssh2` directory on the server and rename it; for example, `client_id_dsa_1024_a.pub`.
4. Create or edit a file called `authorization` in the user's `$HOME/.ssh2` directory and insert the following line that identifies the name of the user's client public key file that was copied in step 3:

```
Key    client_id_dsa_1024_a.pub
```

This line directs the server to use the public key in the authorization file when authenticating the user's login.

### 3.3.2.3 Logging In to the Server Using Public Key Authentication

Example 3–3 shows sample output when logging into a server that is using public key authentication.

#### Example 3–3: Public Key Authentication Login Output

---

```
$>ssh server_name
Passphrase for key "/home/user/.ssh2/id_dsa_1024_a
with comment "1024-bit dsa, created by user@Local
Wed July 26 2001 00:13:43 +0200":
```

---

### 3.3.2.4 Managing Passphrases

The passphrase is not the user's Tru64 UNIX user account password. The passphrase is the secret text that the user entered with the `ssh-keygen2` or the `ssh-pubkeymgr` command to create a key pair. The passphrase is used only by the client and server to exchange information about the user.

During a session with a server that uses public key authentication, users are prompted for their passphrase when they enter a Secure Shell command. For example, during a session the user is prompted for a passphrase each time the user enters the `scp` command or the `sftp` command.

Users can configure the server so that it does not repeatedly prompt for passphrases during a session by running an agent and loading their private keys into memory. When the agent is running, the client contacts the agent for all key-related operations. The agent terminates when the user logs out or stops the agent.

To run the agent and load private keys into memory:

1. Log in to the server.
2. Start the agent. Users can enter the following command, which is only applicable for that login session, or add the entry to their login files; for example, `.login` or `.xsession`, to automatically run the agent when they log in to the server.

```
ssh-agent2 $SHELL
```

The `$SHELL` environment variable identifies the user's login shell.

The agent invokes the specified shell as a child process, and the shell prompt appears.

3. Load the private keys into the agent:

```
$ ssh-add2
```

The `ssh-add` command prompts the user for a passphrase, then loads the private keys in the `identification` file into memory.

### 3.3.3 Host-Based Authentication Method

The host-based authentication is a method in which authentication is based on system identification, not user password or passphrase identification. Host-based authentication trusts that the user application validated the user's identity.

When a user enters a Secure Shell command to connect to a server that requires host based authentication, the server checks for an entry for the client system in the `$HOME/.rhosts` and `$HOME/.shosts` files. These files are located in a user's home directory and contain the names of remote host and user who can log in without a password. If a host is a TruCluster Server member, add the cluster alias to the `.shosts` file.

---

#### Note

---

If either of these files allow access for a particular connection, the connection is allowed, even if the other file forbids it.

The `$HOME/.shosts` file is used only by the server. The server uses the `$HOME/.rhosts` file for backward compatibility. You can ignore the `$HOME/.rhosts` file if users use only Secure Shell commands and do not use the `r*` commands.

---

The `$HOME/.rhosts` and `$HOME/.shosts` files have a similar format. See `.rhosts(4)` for more information.

#### 3.3.3.1 Configuring Host-Based Authentication

To configure host-based authentication:

1. Set the value of:
  - The `IgnoreRhosts` keyword to `no` in the `/etc/ssh2/sshd2_config` file on the server, which is the default. For example:

```
IgnoreRhosts    no
```
  - The `AllowedAuthentications` keyword to `hostbased` in the `/etc/ssh2/sshd2_config` file on the remote server and in the `/etc/ssh2/ssh2_config` file on the local server. For example:

```
AllowedAuthentications    hostbased, password
```

---

### Note

---

The `hostbased` value must be listed first if there are other authentication methods.

---

If you change the value of a keyword, you must enter the following command to reset the `sshd2` daemon:

```
# /sbin/init.d/sshd reset
```

2. Copy the client's `/etc/ssh2/hostkey.pub` file to the server as `/etc/ssh2/knownhosts/clientname.domain.ssh-dss.pub`.

The `clientname.domain` is the fully qualified domain name for the local client.

The remote server now has a copy of the local client's public key, so the remote server can verify the local client's identity based on a public key signature. By contrast, `rsh` uses only the IP address for host authentication.

3. For connections between servers in a TruCluster Server environment, create a symbolic link from the cluster alias public key file in the `knownhosts` directory to the local public key file. For example:

```
# ln -s /etc/ssh2/knownhosts/cluster_alias.company.com.ssh-dss.pub \
  /etc/ssh2/hostkey.pub
```

For a server outside a TruCluster Server environment to connect to a server inside a TruCluster Server environment, copy the public host key file for the cluster alias to the `knownhosts` directory on the server outside the cluster. This is the only public host key that needs to be copied if users always connect using the cluster alias. For example, if `clustera` is a cluster that contains members called `mem1` and `mem2`, and users need to connect with a server outside the cluster called `serverb`, copy the public key from a cluster member to `serverb` as follows:

```
# ftp /etc/ssh2/hostkey.pub \
serverb:/etc/ssh2/knownhosts/clustera.company.com.ssh-dss.pub
```

If users must connect to a specific server in a TruCluster Server environment, create a symbolic link from the specific cluster member public key file to the cluster alias public key file. For example:

```
# ln -s mem1.company.com.ssh-dss.pub \
  clustera.company.com.ssh-dss.pub
# ln -s mem2.company.com.ssh-dss.pub \
  clustera.company.com.ssh-dss.pub
```

4. Add the fully qualified domain name for the client to the `DefaultDomain` keyword in the `/etc/ssh2/ssh2_config` file. For example:

```
DefaultDomain    yourdomain.com
```

5. On the remote server, create a file called `.shosts` in the remote user's home directory and add the following entry:

```
localhostname.yourdomain.com    LocalUser
```

The remote user must be the owner of the `.shosts` file. Use the `chown` command to set the remote user as the owner of the `.shosts` file. Use the `chmod` command to set the permissions to `0400` on the `.shosts` file.

### 3.3.3.2 Connecting Using Host-Based Authentication

To connect using host-based authentication, enter the following command from local server:

```
$ ssh RemoteUser@Remote_server
```

## 3.4 Managing the Server

This section describes how to:

- Start, stop, and reset the `sshd2` daemon
- Generate a host key
- Forward TCP/IP and X11 data through an Secure Shell connection

### 3.4.1 Starting, Stopping, and Resetting the `sshd2` Daemon

By default, when the `sshd2` daemon starts, it uses the configuration information in the `/etc/ssh2/sshd2_config` file. When you start the `sshd2` daemon, you can specify configuration options on the command line. Command line configuration options override values in the `/etc/ssh2/sshd2_config` file and are effective only until the `sshd2` daemon restarts. To permanently make a configuration change, edit the `/etc/ssh2/sshd2_config` file. See Section 3.1 for more information on the `/etc/ssh2/sshd2_config` file.

- To start the `sshd2` daemon and specify command line configuration options, enter:  

```
# /usr/sbin/sshd2 option
```
- To start the `sshd2` daemon using the configuration information in the `/etc/ssh2/sshd2_config` file, enter:  

```
# /sbin/init.d/sshd start
```
- To stop the `sshd2` daemon, enter:  

```
# /sbin/init.d/sshd stop
```
- To reset the `sshd2` daemon, enter:

```
# /sbin/init.d/sshd reset
```

## 3.4.2 Generating A Host Key

You need to generate a host key only if you want to change the host key, or if your host key was not generated during the installation procedure. To generate a host key:

1. Log in as root user.
2. Stop the `sshd2` daemon:  

```
# /sbin/init.d/sshd stop
```
3. Generate the host key:  

```
# ssh-keygen2 -P /etc/ssh2/hostkey
```
4. Start the `sshd2` daemon:  

```
# /sbin/init.d/sshd start
```

## 3.4.3 Forwarding TCP/IP and X11 Data Through a Secure Shell Connection

The Secure Shell connection protocol provides channels that can be used for a wide range of purposes. These channels are multiplexed into a single encrypted tunnel and can be used for forwarding (tunneling) arbitrary TCP/IP ports and X11 connections.

### 3.4.3.1 TCP/IP Port Forwarding

TCP/IP port forwarding, or tunneling, is a way to forward otherwise insecure TCP traffic through a Secure Shell connection. For example, you can secure POP3, SMTP, and HTTP connections that would otherwise be insecure.

There are two kinds of TCP/IP port forwarding:

- Local forwarding (also known as outgoing tunnels)  
Local port forwarding forwards traffic coming to a local port to a specified remote port. For example, if you enter the following command, all data that comes to port `local_port` on the local system will be forwarded to port `remote_port` on the remote system:  

```
ssh2 -L local_port:remote:remote_port user@remote
```
- Remote forwarding (also known as incoming tunnels)  
Remote port forwarding forwards traffic coming to a remote port to a specified local port. For example, if you enter the following command, all data that comes to port `remote_port` on the remote system will be forwarded to port `local_port` on the local system:

```
ssh2 -R remote_port:local:local_port user@remote
```

To enable TCP forwarding, the value of the `AllowTcpForwarding` keyword must be `yes` in the `/etc/ssh2/sshd2_config` file, which is the default.

### 3.4.3.2 X11 Forwarding

To enable X11 forwarding, the value of the `ForwardX11` keyword must be `yes` in the `/etc/ssh2/sshd2_config` file, which is not the default.

Log in to the remote system and enter the following command to start an X clock program that can be used for testing the forwarding connection:

```
xclock &
```

If the X clock window is displayed, X11 forwarding is working.

---

#### Note

---

Do not set the `DISPLAY` variable on the client. Doing so will disable encryption. (X connections forwarded through Secure Shell use a special local display setting.)

---



---

## Using the Secure Shell Commands

This chapter describes how to use Secure Shell commands to:

- Copy files between clients and servers
- Log in and execute commands on a server

---

### Note

---

Client-to-server communication requires that the client have a copy of the server's public key to authenticate the identity of the server. When a user connects to a new server, the user is prompted to accept a copy of the server's public host key. If the user accepts the key, a copy of the server's public host key is copied to the user's `hostkeys` directory on the client. The client uses this public key to authenticate the server on subsequent connects.

---

## 4.1 Copying Files

To use Secure Shell to securely copy files between clients and servers over the network, you can use:

- The `scp2` command
- The `sftp2` command

### 4.1.1 Using the `scp2` Command

You can use the `scp2` command on a client to securely copy files to and from a server. The `scp2` command runs with normal user privileges. The basic syntax for the `scp2` command is:

```
scp2 user@system:/directory/file user@system:/directory/file
```

Local paths can be specified without the `user@system:` prefix. Relative paths can also be used; they are interpreted relative to the user's home directory.

You can enter the `scp` command. The installation process creates a symbolic link from the `scp` command executable to the `scp2` command executable.

See `scp2(1)` for more information about the `scp` command.

### 4.1.2 Using the `sftp2` Command

You can use the `sftp2` command on a client to securely copy files to and from a server. The `sftp2` command is an FTP-command that works in a similar fashion to the `scp2` command. Even though the `sftp2` command functions like the `ftp` command, the `sftp2` command does not use the FTP daemon or the `ftp` client for its connections. The `sftp2` command runs with normal user privileges.

The basic syntax for the `sftp2` command is:

```
sftp2 [options] hostname
```

You can enter the `sftp` command. The installation process creates a symbolic link from the `sftp` command executable to the `sftp2` command executable.

See `sftp2(1)` for more information about the `sftp2` command.

## 4.2 Logging In and Executing Commands on a Server

You use the `ssh2` command on the client to securely log in and execute commands on a server. The basic syntax for the `ssh2` command is:

```
ssh2 [options] server_name [command]
```

When a user successfully logs in to a server, the `sshd2` daemon:

1. Changes to run with user privileges.
2. Sets up a basic environment.
3. Changes to the user's home directory.
4. Runs the user's shell.

You can enter the `ssh` command. The installation process creates a symbolic link from the `ssh` command executable to the `ssh2` command executable.

See `ssh2(1)` for more information about the `ssh2` command.

---

# Index

## A

---

### authentication

- host based, 3–15
- password, 3–11
- public key, 3–12

## C

---

### client

- configuring, 3–7

### command

- scp2, 4–1
- sftp2, 4–2
- ssh2, 4–2

### configuring

- client, 3–7
- server, 3–1

### copying

- scp2 command, 4–1
- sftp2 command, 4–2

## D

---

### documentation

- installing, 2–2
- removing, 2–3

### downloading

- Secure Shell, 2–1

## E

---

### executing commands

- ssh2 command, 4–2

## F

---

### forwarding, 3–18

- TCP/IP Port, 3–18
- X11, 3–19

## G

---

### generating

- host keys, 3–18

## H

---

### host keys

- generating, 3–18

## I

---

### installing

- documentation, 2–2
- subsets, 2–2

## L

---

### logging in

- ssh2 command, 4–2

## P

---

### passphrase

- managing, 3–14

## R

---

### resetting

- daemon, 3–17

## S

---

### **Secure Shell**

- client, 1–2
- commands, 1–1t, 4–1
- communication, 1–2
- configuring, 3–1
- directories, 2–3
- downloading, 2–1
- files, 2–3
- installing, 2–1
- overview, 1–1

- server, 1–2
- uninstalling, 2–3

### **server**

- configuring, 3–1
- managing, 3–17

### **starting**

- daemon, 2–2, 3–17
- server, 2–2

### **stopping**

- daemon, 3–17

### **subsets**

- installing, 2–2