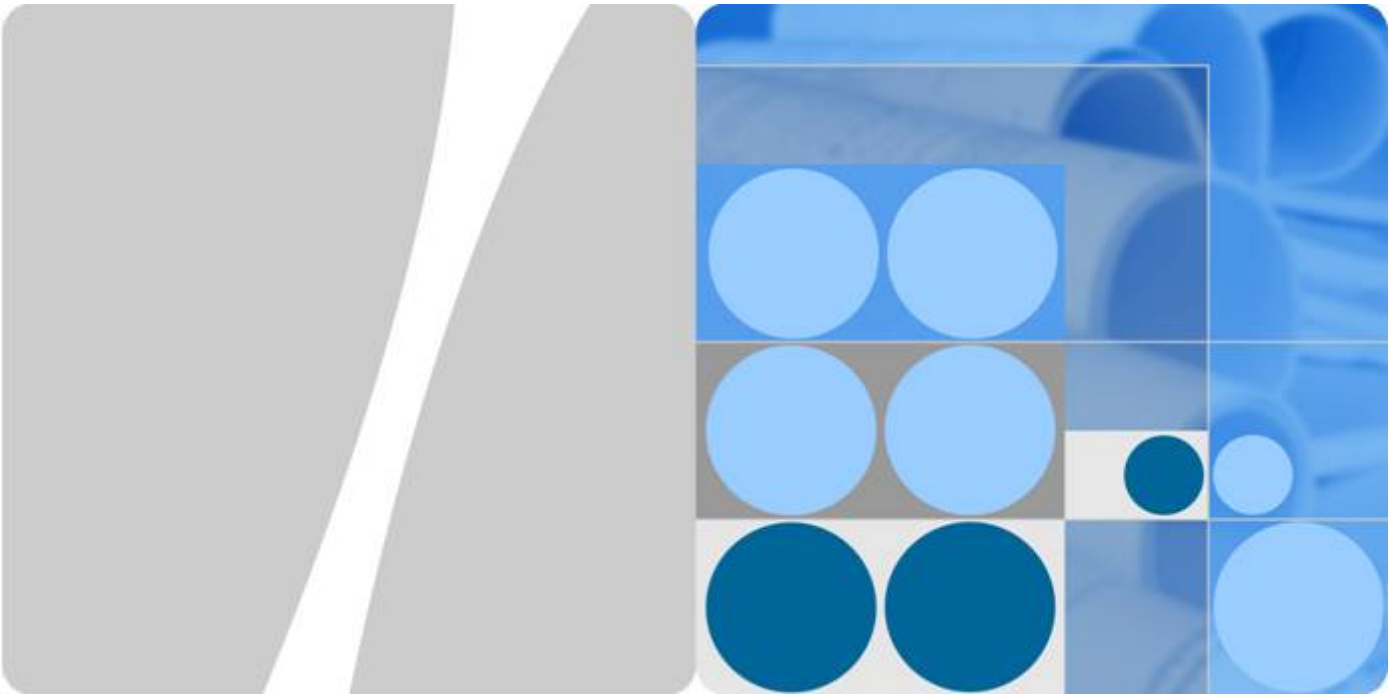


Enterprise Data Communication Products

Feature Description - Basic Configuration

Issue 01

Date 2012-09-30

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://enterprise.huawei.com

About This Document

Intended Audience

This document describes the definition, purpose, and implementation of features on enterprise datacom products including the campus network switch, enterprise router, data center switch, and WLAN. For features supported by the device, see *Configuration Guide*.

This document describes how to configure the Basic configuration.

This document is intended for:

- Data configuration engineers
- Commissioning engineers
- Network monitoring engineers
- System maintenance engineers

Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
 DANGER	Indicates a hazard with a high level of risk, which if not avoided, will result in death or serious injury.
 WARNING	Indicates a hazard with a medium or low level of risk, which if not avoided, could result in minor or moderate injury.
 CAUTION	Indicates a potentially hazardous situation, which if not avoided, could result in equipment damage, data loss, performance degradation, or unexpected results.
TIP	Indicates a tip that may help you solve a problem or save time.
 NOTE	Provides additional information to emphasize or supplement important points of the main text.

Command Conventions

The command conventions that may be found in this document are defined as follows.

Convention	Description
Boldface	The keywords of a command line are in **boldface**.
Italic	Command arguments are in *italics*.
[ ]	Items (keywords or arguments) in brackets [ ] are optional.
{ x | y | ... }	Optional items are grouped in braces and separated by vertical bars. One item is selected.
[x | y | ...]	Optional items are grouped in brackets and separated by vertical bars. One item is selected or no item is selected.
{ x | y | ... }*	Optional items are grouped in braces and separated by vertical bars. A minimum of one item or a maximum of all items can be selected.
[x | y | ...]*	Optional items are grouped in brackets and separated by vertical bars. Several items or no item can be selected.
&<1-n>	The parameter before the & sign can be repeated 1 to n times.
#	A line starting with the # sign is comments.

Interface Numbering Conventions

Interface numbers used in this manual are examples. In device configuration, use the existing interface numbers on devices.

Change History

Changes between document issues are cumulative. Therefore, the latest document version contains all updates made to previous versions.

Changes in Issue 01 (2012-09-30)

Initial commercial release.

Contents

1 Basic Configuration

About This Chapter

1.1 Introduction to Basic Configuration

Definition

In configuration management, the terminal service provides the access interface and human-machine interfaces (HMIs) for users to configure devices.

The login mode includes:

- Login through the console port
- Telnet server/client
- Login through Secure Shell (SSH), with password, Revest-Shamir-Adleman Algorithm (RSA) authentication, and Digital Signature Algorithm (DSA)
- Login through customized user interfaces providing multiple user authentications and authorization modes

The file transfer provides transmission control for system files and configuration files, and simple remote management for the file system.

The file transfer mode includes:

- FTP client/server
- TFTP client
- SSH FTP (SFTP) client/server
- SSL FTP (FTPS) client/server

This document describes the principles of every protocol feature according to the protocol type.

It includes the following parts:

- FTP
- TFTP
- Telnet
- SSH
- SSL
- User management
- Virtual file system
- Daylight saving time
- Timing restart

Purpose

The terminal service provides the access interface and HMIs for users to configure devices. The file transfer provides transmission control for system files and configuration files, and simple remote management for the file system.

1.2 Principles

1.2.1 TFTP

Overview

The Trivial File Transfer Protocol (TFTP) is a simple protocol for file transfer.

The TFTP client supports file upload and download by using TFTP. To ensure simple implementation, TFTP uses the User Datagram Protocol (UDP) as its transport protocol.

Compared with FTP, TFTP does not require complicated interaction interfaces and authentication control. Thus, TFTP is applicable in a networking environment without complicated interactions between the client and the server. For example, you can obtain the memory image of the system through TFTP when the system is started up. To retain the small size of TFTP packets, TFTP is realized based on UDP.

Presently, the device implements the TFTP client rather than the TFTP server. The TFTP client can upload and download files.

Basic Concepts of TFTP

- Operation code

  TFTP packet header contains a two-byte operation code, with values defined as follows:
  - 1: Read request (RRQ): indicates a read request (RRQ).
  - 2: Write request (WRQ): indicates a write request (WRQ).
  - 3: Data (DATA): indicates data packets.
  - 4: Acknowledgment (ACK): indicates a positive reply packet.
  - 5: Error (ERROR): indicates error packets.
- File type

  TFTP supports the following file types:
  - Binary type: is used to transfer program files.
  - ASCII type: is used to transfer text files.

Currently, the device can act only as the TFTP client and only the binary transfer type is available.

Basic Principle of TFTP

- The user name and password are not required.

  This is because TFTP is designed for the bootstrap process.
- TFTP transfer

  The client initiates the TFTP transfer.
  - To download files, the client sends an RRQ to the server. The server then accepts the request and sends a data packet to the client. After receiving the data packet, the client sends an ACK packet to the server.
  - To upload files, the client sends an WRQ to the server. After the server accepts the request, the client sends a data packet to the server and waits for an ACK packet from the server.

1.2.2 FTP

Overview

As a standard of file transfer, the File Transfer Protocol (FTP) runs at the application layer in the TCP/IP protocol suite. It is used for transferring files between local hosts and remote hosts, especially in version upgrade, log download, file transfer, and configuration saving. FTP is implemented based on the file system.

FTP adopts the client/server architecture, as shown in **Figure 1-1**.

Figure 1-1 Networking diagram of FTP client/server architecture

- FTP server: indicates that the device functions as an FTP server. It provides access and operation for the remote client. Users can log in to the device and access the files on the device using the FTP client program.
- FTP client: provides commands on the local device to perform operation on the files on the remote server. After setting up a connection with the device by running the terminal emulation program or Telnet program on the PC, the user can set up a connection with the remote FTP server by using the FTP command and access files on the remote server.

Establishment of FTP Connections

FTP uses a control connection and a data connection to transmit files. A control connection connects to the control port to transmit control commands, while data connection connects to the data port. After a control connection is set up, a data connection is established by running commands on the control port to transmit data.

An FTP connection can be set up in active mode and passive mode. In active mode, the data connection is initiated by the server; in passive mode, the data connection is initiated by the client. By default, the active mode is used. The mode can be switched by commands.

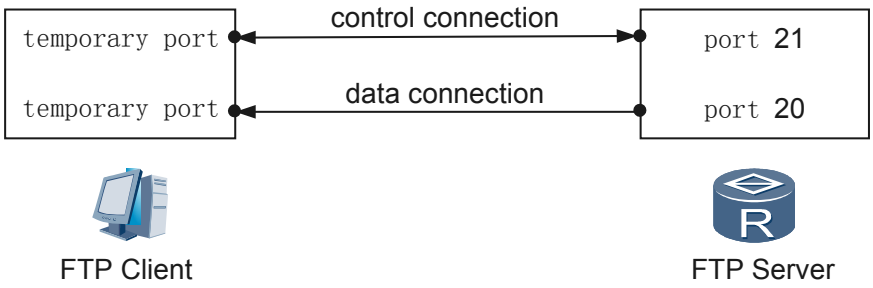
In active mode, when a firewall is configured on the client, the data connection may fail because the connection is initiated by the server. In passive mode, this problem does not happen. The active mode facilitates management on the FTP server but not on the client, while the passive mode facilitates management on the client but not on the FTP server.

By default, port 21 on the server is used to transmit control commands and port 20 is used to transmit data.

Establishing an FTP Connection in Active Mode

Figure 1-2 shows the process of setting up an FTP connection in active mode.

Figure 1-2 Process of setting up an FTP connection in active mode

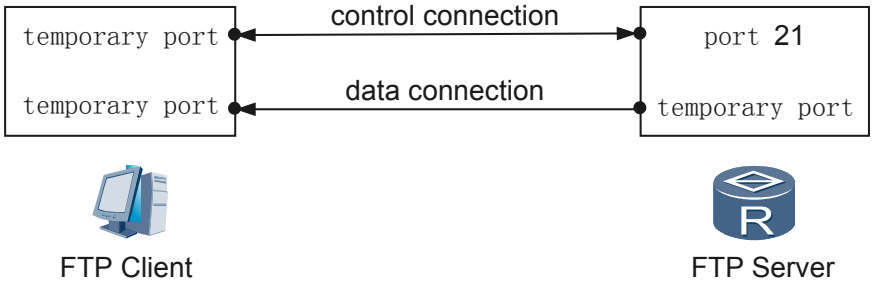


1. The server enables port 21 to wait to set up a connection with the client.
2. The client initiates a request to set up a control connection. Then the server responds to the request and a control connection is set up.
3. The client sends the **PORT** command through the control connection to inform the server of the temporary port number of a data connection.
4. Port 20 on the server sets up a data connection with the client.

Establishing an FTP Connection in Passive Mode

Figure 1-3 shows the process of setting up an FTP connection in passive mode.

Figure 1-3 Process of setting up an FTP connection in passive mode



1. The server enables port 21 to wait to set up a connection with the client.
2. The client initiates a request to set up a control connection. Then the server responds to the request and a control connection is set up.
3. The client sends the **PASV** command through the control connection to inform the server that the client is in passive mode.
4. The server replies and informs the client of the temporary port number of a data connection.
5. The client initiates a connection with the temporary port on the server.

NOTE

The temporary port is generated at random.

1.2.3 Telnet

Overview

The Telecommunication Network Protocol (Telnet) is derived from APPANET released in 1969. It is the earliest Internet application. Telnet enables a terminal to remotely log in to a server and presents an interactive operation interface. Users can first log in to one host and then log in to other hosts through Telnet to configure and manage hosts without need of connecting each host to a hardware terminal.

Basic Concepts of Telnet

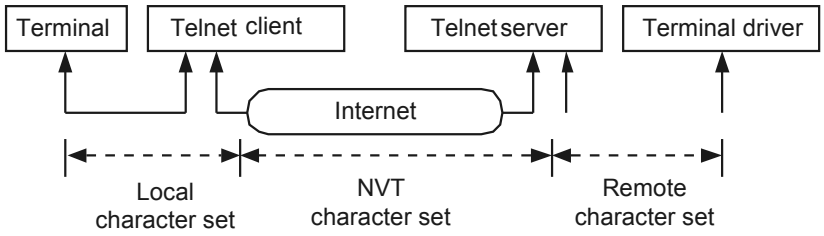
- NVT

  The Network Virtual Terminal (NVT) is a bidirectional virtual device, to and from which both ends of the connection, the client and the server, map their physical terminals. Because of the use of uniformed NVT, Telnet can operate between any two hosts (any operating system) or terminals.

  NVT is a virtual device, to and from which both ends of the connection, the client and the server, map their physical terminals. The client operating system maps whatever type of a user terminal to the NVT type; the server operating system maps the NVT type to the terminal type supported by the server.

  The mapping model between the physical terminal and NVT is shown in **Figure 1-4**.

  Figure 1-4 Networking diagram of the mapping model between the physical terminal and NVT

- NVT ASCII

  NVT ASCII refers to a 7-bit ASCII character set. Each 7-bit character is added with 0 following the highest-order bit to be sent as an 8-bit byte. The Internet protocol suite including FTP and the Simple Mail Transfer Protocol (SMTP) uses NVT ASCII.

- IAC

  Telnet uses in-band signaling bidirectionally. Byte 0xff is called IAC, short for interpret as command. The byte following 0xff indicates a command.

  The following lists the commands involved in the divice and their meanings:

  - SE: suboption end
  - SB: suboption begin
  - WILL: option negotiation
  - WONT: option negotiation
  - DO: option negotiation

- DONT: option negotiation
- IAC: indicating that the following byte is interpreted as a command

Table 1-1 Telnet Command Set Defined in the RFC

Name	Code (Decimal Notation)	Description
EOF	236	End of file
SUSP	237	Suspend the execution of the current process.
ABORT	238	Abort the process.
EOR	239	End of record
SE	240	End of subnegotiation parameters
NOP	241	No operation
DM	242	Data mark
BRK	243	Break
IP	244	Interrupt process
AO	245	Abort output
AYT	246	Are you there
EC	247	Erase character
EL	248	Erase line
GA	249	Go ahead
SB	250	Suboption begin
WILL	251	Option negotiation
WONT	252	Option negotiation
DO	253	Option negotiation
DONT	254	Option negotiation
IAC	255	Data byte 255

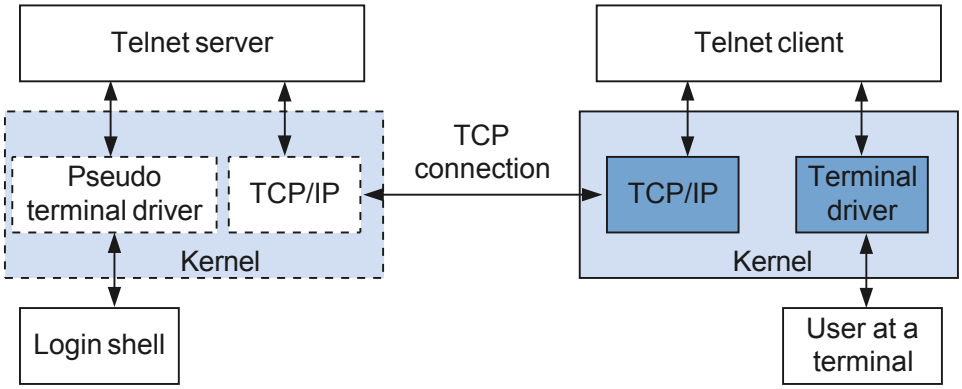
- Telnet connection

  A Telnet connection is a TCP connection used to transmit data with Telnet control information.

- Telnet client/server mode

  Telnet adopts the client/server mode. **Figure 1-5** shows the schematic diagram of the Telnet client/server mode.

Figure 1-5 Schematic diagram of the client/server mode adopted by Telnet



The preceding diagram shows that:

- Telnet uses the Transmission Control Protocol (TCP).
- All echo messages of the Telnet connection are output to the terminal.
- The server directly interacts with the pseudo terminal.
- Commands and data are transmitted between the server and the client through the TCP connection.
- The client logs in to the server.

Principle of Telnet

Telnet is designed to operate between any two hosts or terminals. The client operating system maps the terminal to the NVT regardless of the terminal type. The server must also map the NVT into whatever terminal type the server supports. This shields the specific client and terminal types. Communication ends are simply assumed as being connected to the NVTs.

NOTE

Telnet adopts the symmetric mode. In principle, there must be an NVT at both ends of a Telnet connection.

Both ends of a Telnet connection can send a WILL, WONT, DO, or DONT request for option negotiation. The options include echo, character set of command change, and linemode.

This section describes the operating principle of Telnet from the following aspects:

- Requests in a Telnet connection

  Either end of a Telnet connection can initiate a request to the other end. **Table 1-2** shows different requests and their meanings.

Table 1-2 Meanings of requests in a Telnet connection

Request from the Sender	Description	Response from the Receiver			
		WILL	WONT	DO	DONT

Request from the Sender	Description	Response from the Receiver			
WILL	The sender wants to enable the option.	-	-	The receiver accepts the request.	The receiver denies the request.
WONT	The sender wants to disable the option.	-	-	-	The receiver must accept the request (1).
DO	The sender wants the receiver to enable the option.	The receiver accepts the request.	The receiver denies the request.	-	-
DONT	The sender wants the receiver to disable the option.	-	The receiver must accept the request(1).	-	-

NOTE

When the sender sends an "option disable" request, such as WONT and DONT, the receiver must accept the request.

When the sender sends the "option enable" request, such as WILL and DO, the receiver can either grant or reject the request.

- If the receiver accepts the request, the option is enabled immediately.
- If the receiver rejects the request, the option remains disabled, but the sender can still retain the features as the NVT.

- Option negotiation

  Option negotiation requires three bytes:

  The IAC type, the byte for WILL, DO, WONT or DONT, and the option ID.

  The following example illustrates the process of option negotiation.

  The server sends a request for enabling "remote traffic control" with option ID 33, and the client accepts the request. The commands exchanged between the server and client are as follows:

  - On the server: <IAC,WILL,33>
  - On the client: <IAC,DO,33>

- Suboption negotiation

Certain options require more information than the option ID. For example, if the sender requires the receiver to specify the terminal type, the receiver must respond with an ASCII string for specifying the terminal type.

The format of the commands for suboption negotiation is as follows:

< IAC, SB, option code, contents of suboption, IAC, SE >

A complete process of suboption negotiation is as follows:

- The sender sends a DO or WILL command carrying an option ID to request that the option be enabled.
- The receiver returns a WILL or DO command carrying the option ID to accept the request.

  Through the preceding two steps, both ends agree to enable the option.

  One end of the connection starts suboption negotiation by sending a request composed of SB, suboption ID, and SE in sequence.
- The opposite end responds to the request for suboption negotiation by sending a command composed of SB, suboption ID, related negotiation information, and SE in sequence.
- The receiver returns a DO or WILL command to accept the negotiation information about the suboption.

If there is no other suboption to be negotiated, the current negotiation is complete.

NOTE

In the preceding process, the receiver is assumed to accept the requests from the sender. In actual situations, the receiver can reject requests from the sender at any time as required.

The following example illustrates the process of terminal type negotiation.

The client needs to enable "terminal type" with option ID 24. The server accepts the request and returns a request for querying the terminal type of the client. The client then sends a response carrying its terminal type "DELL PC" to the server. The commands exchanged between the server and client are as follows:

- From the client: < IAC, WILL, 24 >
- From the server: < IAC, DO, 24 >
- On the server: <IAC, SB, 24, 1, IAC, SE>
- From the client: < IAC, SB, 24, 0, 'D', 'E', 'L', 'L', 'P', 'C', IAC, SE >

NOTE

- Only the sender that sends the DO command can request terminal type information.
- Only the sender that sends the WILL command can provide terminal type information.

Terminal type information cannot be sent automatically but only in request-response mode.

The terminal type is an NVT ASCII string of case insensitive characters.

- Operating modes

  Telnet has the following operating modes:

  - Half-duplex mode
  - A character at a time
  - A line at a time
  - Linemode

Telnet in the device

The device covers the following Telnet services.

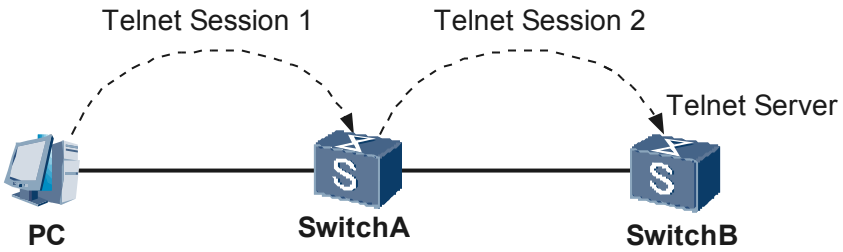
- Telnet server

  A user runs the Telnet client application on the PC so that it can log in to the device through Telnet and then manage the device.

- Telnet client

  After running the emulation terminal program or Telnet client application on the PC to set up a connection with the device, a user can run the **telnet** command to log in to and manage other devices. In the scenario shown in **Figure 1-6**, Switch A acts as a Telnet server. It, however, can act as a Telnet client simultaneously.

Figure 1-6 Networking diagram of Switch A acting as the Telnet client



1.2.4 SSH

Overview

SSH is short for Secure Shell. Its standard port number is 22.

Data transmission in Telnet mode is prone attacks, because it does not have a secure authentication mode and use TCP to transmit data in plain text. Simple Telnet access is vulnerable to Denial of Service (DoS) attacks, IP address spoofing, and route spoofing.

With the increasing emphasis on network security, data transmission in plain text used by traditional Telnet and FTP becomes unacceptable. SSH is a network security protocol. It provides the secure remote access and other secure network services on an insecure network by encrypting network data.

SSH uses TCP to exchange data and builds a secure channel based on TCP. In addition to standard port 22, SSH supports access through other service ports to prevent attacks.

SSH supports password authentication, Digital-Signature Algorithm (DSA) and Revest-Shamir-Adleman Algorithm (RSA) authentication. It uses DES, 3DES, and AES encryption to prevent password interception, thus ensuring the integrity and reliability of the data and guarantee the secure data transmission. In particular, RSA and DSA authentication supports the combined use of symmetric encryption and asymmetric encryption. This implements secure key exchange and finally secures the session process.

By virtue of data encryption in transmission and more secure authentication, SSH is widely used and has become one of the important network protocols.

SSH has two versions: SSH1 (SSH 1.5) and SSH2 (SSH 2.0). Both are different and incompatible. SSH2.0 is superior to SSH 1.5 in security, functions, and performance.

Devices that can function as the STelnet client and server and SFTP client and server support both SSH1 (SSH 1.5) and SSH2 (SSH 2.0).

Secure Telnet (STelnet) enables users to remotely and securely log in to the device, and provides the interactive configuration interface. All data exchanges based on STelnet are encrypted. This ensures the security of sessions.

The SSH File Transfer Protocol (SFTP) enables users to log in to the device securely for file management from a remote device. This improves the security of data transmission for the remote system update. Meanwhile, the client function provided by SFTP enables users to log in to the remote device for the secure file transmission.

Basic Concepts of SSH

- SFTP

  SFTP guarantees secure file transfer over an insecure network by authenticating the client and encrypting data in bidirectional mode.

- SCP

  SCP guarantees secure file transfer over an insecure network by authenticating the client and encrypting data in bidirectional mode.

- STelnet

  STelnet ensures secure Telnet services. It guarantees secure file transfer on a traditional insecure network by authenticating the client and encrypting data in bidirectional mode.

- RSA authentication

  RSAauthentication is based on the private key of the client. It is a public key encryption architecture and an asymmetric encryption algorithm. Based on the problem of factoring large numbers, RSA is mainly used to transmit the keys of the symmetric encryption algorithm, which can improve encryption efficiency and simplify key management.

  The server checks whether the SSH user, public key, and digital user signature are valid. If all of them are valid, the user is permitted to access the server; if any of them is invalid, the authentication fails and the user is denied to access the server.

- DSA authentication

  The digital signature algorithm (DSA) is an asymmetric encryption algorithm used for the authenticating clients. DSA algorithm consists of a public key and a private key.

  Like RSA, the server checks whether the SSH user, public key, and digital user signature are valid. If all of them are valid, the user is permitted to access the server; if any of them is invalid, the authentication fails and the user access is denied.

  Compared with RSA authentication, DSA authentication adopts the DSA encryption mode and is widely used.

  - In many cases, SSH only supports DSA to authenticate the server and the client.
  - In SSH, DSA authentication takes precedence over RSA authentication.

- Password authentication

  Password authentication is based on the user name and password.

  On the server, the AAA module assigns a login password to each authorized user . The server has the mappings between user names and passwords. When a user requests to access

the server, the server authenticates the user name and password of the user. If either of them fails to pass the authentication, the access request of the user is denied.

- RSA-password authentication and DSA-Password authentication

  The server can authenticate the client by checking both the public key and the password. It allows user to access only when both public key and password are consistent with those configured on the server.

- ALL authentication

  The server can authenticate the client by checking both the public key and the password. It allows user to access when either the public key or the password is consistent with those configured on the server.

SSH Features Supported by the Device

- Basic SSH functions
  - Different encryption algorithms for incoming and outgoing data
  - Different MAC algorithms for incoming and outgoing data
  - Encryption algorithms of 3DES-cbc, DES, and Advanced Encryption Standard (AES128)
  - HMAC-sha1 authentication algorithm
  - diffie-hellman-group1-sha1 algorithm for key exchange
  - Public key format of SSH-RSA
  - Public key format of SSH-DSA
  - Key re-exchange (It indicates renegotiation of the key. During this process, the algorithm and the key used for the algorithm are negotiated.)
  - Public key authentication and password authentication
- SSH client function

  The SSH client function allows users to establish SSH connections with a UNIX host or the device supporting the SSH server. **Figure 1-7** and **Figure 1-8** show the establishment of an SSH connection in the Local Area Network (LAN) and in the Wide Area Network (WAN) respectively.

Figure 1-7 Establishing an SSH connection in a LAN

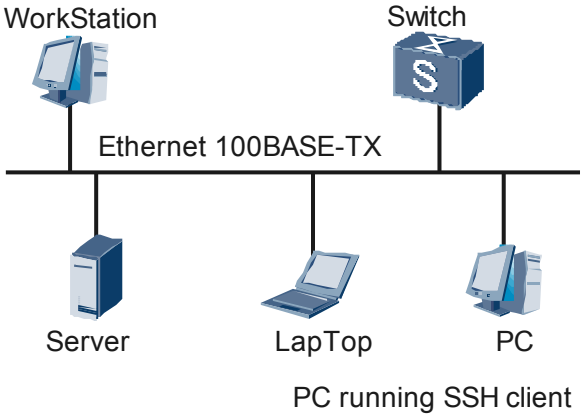
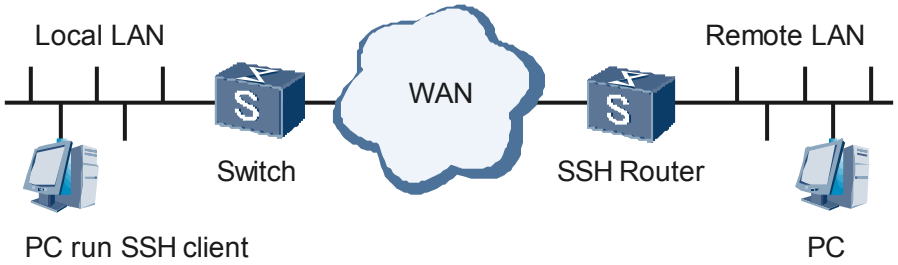


Figure 1-8 Establishing an SSH connection in a WAN



- SSH for SFTP

  SFTP is based on SSH2.0. It guarantees secure file transfer on a traditional insecure network by authenticating the client and encrypting data in bidirectional mode.

  An SFTP-enabled device can provide the following functions:

  - Acting as the SFTP client or the SFTP server
  - Being enabled with or disabled from SFTP services (By default, SFTP services are disabled.)
  - Setting the default directory that the SFTP client is allowed to access

- SSH for SCP

  SCP is based on SSH2.0. It guarantees secure file transfer on a traditional insecure network by authenticating the client and encrypting data in bidirectional mode.

  An SCP-enabled device can provide the following functions:

  - Acting as the SCP client or the SCP server
  - Being enabled with or disabled from SCP services (By default, SCP services are disabled.)

- SSH for STelnet

  An STelnet-enabled device can provide the following functions:

  - Acting as the STelnet client or the STelnet server
  - Being enabled with or disabled from STelnet services. (By default, STelnet services are disabled.)

- SSH for non-standard ports

  The standard SSH listening port number is 22. When attackers continuously access the port, the bandwidth and performance of the server is reduced and authorized users are prevented from accessing this port. This is known as a DoS attack.

  To address the problem, you can change the listening port to another port on the SSH server so that attackers cannot know the actual listening port. This prevents attackers from consuming bandwidth and system resources during their continuous accesses to the standard port. Authorized users can access the SSH server through non-standard ports to decrease DoS attacks.

  Applications of this function are as follows:

  - The STelnet client can access the server using a non-standard port.
  - The listening port can be set on the SSH server.

Principles of SSH

SSH uses the traditional client/server (C/S) application model. Its security is guaranteed by using the following modes:

Data encryption: Through the negotiation between the client and the server, an encryption key is generated and used in data symmetric encryption. This ensures the confidentiality during data transmission.

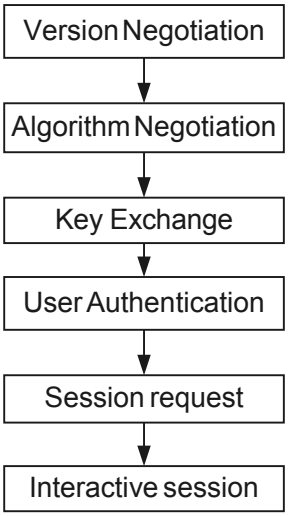
Data integrity: Through the negotiation between the client and the server, an integrity key is generated and used to uniquely identify a session link. All session packets are identified by the integrity key. Any modifications made by the third party during transmission can be discovered by the receiver based on the integrity key. The receiver can thus discard these modified packets to ensure the data integrity.

Authority authentication: There are multiple authentication modes. Authority authentication allows only valid users to have session with the server, thus improving system security and safeguarding the benefits of valid users.

Establishment of an SSH Connection

The SSH connection goes through six phases in the whole communication process, as shown in **Figure 1-9**. The SSH connection is established through negotiation. The following is the whole SSH negotiation procedure.

Figure 1-9 Establishment of an SSH connection



1. Version negotiation

   In the version negotiation phase, the SSH client sends a request for setting up a TCP connection to the SSH server. After the TCP connection is set up, the SSH server and SSH client negotiate the SSH version. After a matched version protocol is obtained, different version protocols correspond to different state machine processes. If the version of the client matches that of the server, the key negotiation starts; otherwise, the SSH server tears down the TCP connection.

2. Algorithm negotiation

In the algorithm negotiation phase, the sender sends algorithm negotiation messages to the receiver, together with their parameters, such as the random cookie, key exchange algorithm, host key algorithm, Message Authentication Code (MAC) method, and supported language.

After receiving algorithm negotiation messages, the receiver compares the received algorithm list set with the local algorithm list set. If the key exchange algorithm, public key encryption algorithm, or MAC algorithm is not found, the receiver tears down the connection with the sender and the algorithm negotiation fails.

3. Key exchange

   After algorithm negotiation is completed, key exchange begins. The client and the server begin to calculate the session ID. The client randomly generates a 32-byte session key. The first 16 bytes of the session key are used to perform the Exclusive-OR (XOR) operation with the 16 bytes of the session ID with the last 16 bytes of the session key unchanged. The result is arrayed into an MP integer by MSB. The public key with the smaller analog is selected from host public keys and server public keys to perform encryption. The encryption result is arrayed into an MP integer in the sequence of MSB first. Then, the public key with the larger analog is selected to perform encryption. The encryption result along with the encryption algorithm selected by the client, the 8-byte cookie transmitted by the server, and the protocol flag of the client is sent to the server.

   During the session, massive data transmission must use the fast-speed symmetrical key algorithm. The symmetrical encryption and decryption need to share keys. The key exchange process implements key's secure transmission over an insecure channel.

   The server is in the waiting state. When receiving a key generation message from the client, the server returns a key generation message to the client, which indicates that key exchange is completed and a new key should be used for communications. If the server fails to receive a key generation message from the client, it returns a key exchange failure message and tears down the connection.

4. User authentication

   After obtaining the session key, the SSH server authenticates the SSH client. The SSH client sends the identity information to the SSH server. After a certain authentication mode is configured on the SSH server, the client sends an authentication request to the server. If the authentication succeeds or the connection with the server expires, the client is cut off from the server.

   The SSH server authenticates a user in one of the following methods:

   - In RSA authentication, the client generates an RSA key pair and sends the public key to the server. When a user initiates an authentication request, the client randomly generates a text encrypted with the private key and sends it to the server. The server decrypts it by using the public key. If decryption succeeds, the server considers this user trustable and grants this user access rights. If decryption fails, the server tears down the connection.
   - In DSA authentication, the client generates a DSA key pair and sends the public key to the server. When a user initiates an authentication request, the client randomly generates a text encrypted with the private key and sends it to the server. The server decrypts it by using the public key. If decryption succeeds, the server considers this user trustable and grants this user access rights. If decryption fails, the server tears down the connection.
   - Password authentication is implemented based on AAA. Like Telnet and FTP, SSH supports local database authentication and remote RADIUS server authentication. The

SSH server compares the user name and password of an SSH client with the pre-configured ones. If both are matched, authentication succeeds.

5. Session request

   After user authentication is completed, the client sends a session request to the server. The session requests include the running of Shell and commands. At the same time, the server waits to process the request from the client. In this phase, the server responds to the client with an SSH_SMSG_SUCCESS message after successfully processing a request from the client. If the server fails to process or identify the request, it responds with an SSH_SMSG_FAILURE message.

   Possible causes for the authentication failure are as follows:

   - The server fails to process the request.
   - The server cannot identify the request.

6. Interactive session

   After the session request is accepted, the SSH connection enters the interactive session mode. In this phase, data is transmitted bidirectionally.

   a. The client sends a packet with the encrypted command to the server.

   b. After receiving the packet, the server decrypts the packet and runs the command.. Then, the server packages encrypted command execution results and sends the packet to the client.

   c. Upon receiving the packet, the client decrypts it and displays command execution results on the terminal.

1.2.5 SSL

Overview

The Secure Sockets Layer (SSL) protocol is a cryptographic protocol that provides communication security over the Internet. It allows a client and a server to communicate in a way designed to prevent eavesdropping by authenticating the server or the client.

SSL and application layer protocols work independently. Connections of application layer protocols such as HTTP and FTP can be established based on SSL handshakes. Before a client and a server use an application layer protocol to communicate, SSL is used to determine cryptography, negotiate a secret key, and authenticate the server. Data that is then transmitted using the application layer protocol between the client and the server will be encrypted, protecting privacy.

SSL has the following advantages:

- Provides secure network transmission. It uses data encryption, authentication, and a message integrity check to ensure secure data transmission over the network.
- Supports various application layer protocols. SSL is originally designed for securing World Wide Web traffic. As SSL functions between the application layer and the transport layer, it secures data transmission based on TCP connections for any application layer protocol.
- Is easy to deploy. Currently, SSL has become a world-wide communications standard for authenticating Web site and Web page users and encrypting data transmitted between browser users and Web servers.

Security Mechanism

- Connection privacy

  SSL uses symmetric cryptography to encrypt data to be transmitted and the key exchange algorithm Rivest Shamir and Adleman (RSA), which is one of asymmetric algorithms, to encrypt the key used by the symmetric cryptography.

- Identity authentication

  Digital-signed certificates are used to authenticate a server and a client that attempt to communicate with each other. Authenticating the client identity is optional. The SSL server and client use the mechanism provided by the Public Key Infrastructure (PKI) to apply to a CA for a certificate.

- Message integrity

  A keyed message authentication code (MAC) is used to verify message integrity during transmission.

  A MAC algorithm computes a key and arbitrary-length data to output a MAC.

  - A message sender uses a MAC algorithm and a key to compute a MAC and adds it to the end of the message before sending the message to the receiver.
  - The receiver uses the same key and MAC algorithm to compute a MAC and compares the computed MAC with the MAC in the received message.

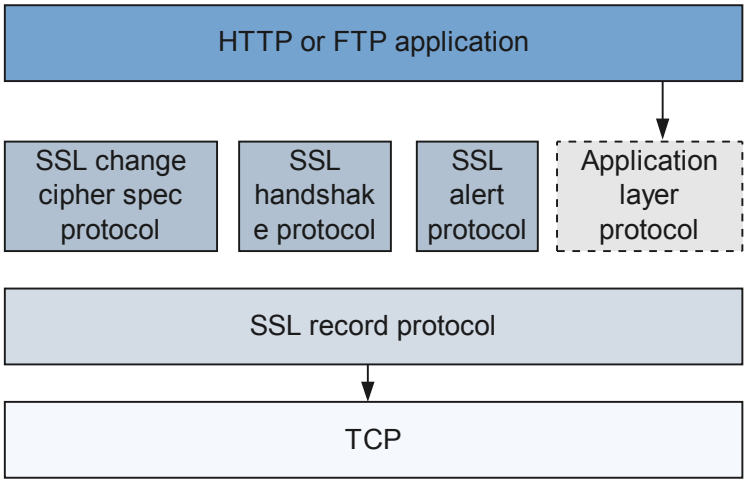
  If the two MACs are the same, the message has not been tampered during transmission. If the two MACs are different, the message has been tampered during transmission, and the receiver will discard this message.

Working Process

- SSL protocol structure

  As shown in **Figure 1-10**, SSL functions between the application layer and the transport layer. It secures data transmission based on TCP connections for any application layer protocol. SSL can be divided into two layers: lower layer with the SSL record protocol and upper layer with the SSL handshake protocol, SSL change cipher spec protocol, and SSL alert protocol.

Figure 1-10 SSL protocol stack



- SSL record protocol: divides upper-layer information blocks into records, computes and adds MACs, encrypts records, and sends them to the receiver.
- SSL handshake protocol: negotiates a cipher suite including a symmetric encryption algorithm, a key exchange algorithm, and a MAC algorithm, exchanges a shared secret key securely between a server and a client, and authenticates the server and client. The client and server establish a session using the SSL handshake protocol to negotiate session parameters including the session identifier, peer certificate, cipher suite, and master secret.
- SSL change cipher spec protocol: Is used by both the client and server to send a ChangeCipherSpec message to notify the receiver that subsequent records will be protected under the newly negotiated cipher suite and key.
- SSL alert protocol: allows one end to report alerts to the other. An alert message conveys the severity of the message and a description of the alert.

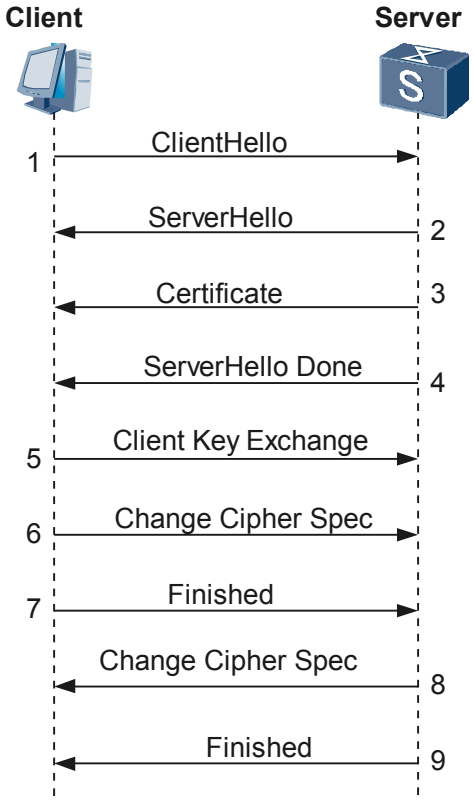
- SSL handshake process

  The client and server negotiate session parameters during the SSL handshake process to establish a session. Session parameters mainly include the session identifier, peer certificate, cipher suite, and master secret. The master secret and cipher suite are used to compute a MAC and encrypt data to be transmitted in this session.

  The SSL handshake process varies according to the real-world situations. Handshake processes in three situations are described as follows:

  - SSL handshake process in which only the server is authenticated

    Figure 1-11 SSL handshake process in which only the server is authenticated

    As shown in **Figure 1-11**, only the SSL server but not the SSL client needs to be authenticated. The SSL handshake process is as follows:

1. The SSL client sends a ClientHello message specifying supported SSL protocol versions and cipher suites to the SSL server.
2. The server responds with a ServerHello message, containing the protocol version and cipher suite chosen from the choices offered by the client. If the server allows this session to be resumed, the server sends the ServerHello message carrying a session ID to the client.
3. The server sends a Certificate message carrying its digital certificate with its public key to the client.
4. The server sends a ServerHelloDone message, indicating that the SSL protocol version and cipher suite negotiation finishes and key information exchange starts.
5. After verifying the validity of the digital certificate of the server, the client responds with a ClientKeyExchange message carrying a randomly generated key (called the master secret), which is encrypted using the public key of the server certificate.
6. The client sends a ChangeCipherSpec message to notify the server that every subsequent message will be encrypted and a MAC will be computed based on the negotiated key and cipher suite.
7. The client computes a hash for all the previous handshake messages except the ChangeCipherSpec message, uses the negotiated key and cipher suite to process the hash, and sends a Finished message containing the hash and MAC to the server. The server computes a hash in the same way, decrypts the received Finished message, and verifies the hash and MAC. If the verification succeeds, the key and cipher suite negotiation is successful.
8. The server sends a ChangeCipherSpec message to notify the client that every subsequent message will be encrypted and a MAC will be computed based on the negotiated key and cipher suite.
9. The server computes a hash for all the previous handshake messages, uses the negotiated key and cipher suite to process the hash, and sends a Finished message containing the hash and MAC to the client. The client computes a hash in the same way, decrypts the received Finished message, and verifies the hash and MAC. If the verification succeeds, the key and cipher suite negotiation is successful.

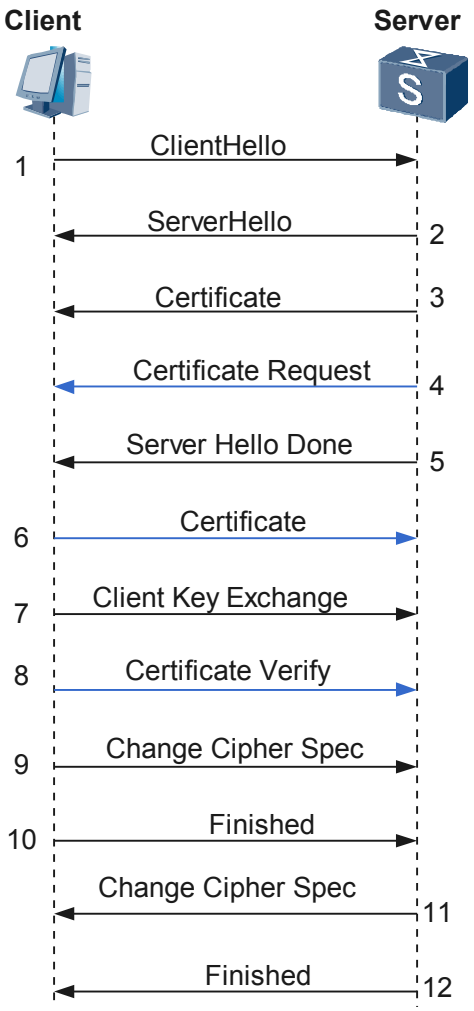
After receiving the Finished message from the server, if the client successfully decrypts the message, the client checks whether the server is the owner of the digital certificate. Only the SSL server that has a specified private key can decrypt the ClientKeyExchange message to obtain the master secret. In this process, the client authenticates the server.

NOTE

- The ChangeCipherSpec message is based on the SSL change cipher spec protocol, and other messages exchanged in the handshake process are based on the SSL handshake protocol.
- Computing a hash means that a hash algorithm (MD5 or SHA) is used to convert an arbitrary-length message into a fixed-length message.

- SSL handshake verification

Figure 1-12 SSL handshake verification

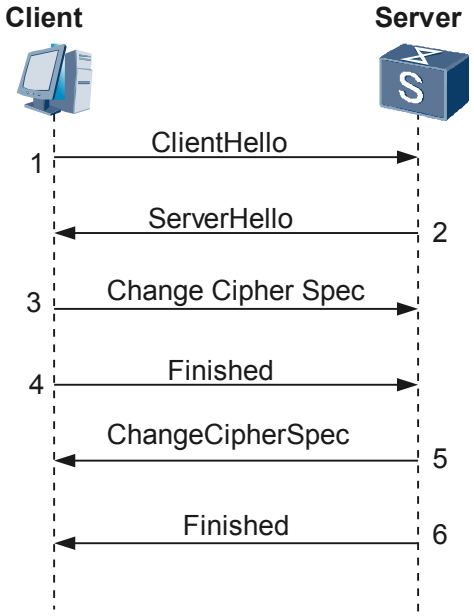


Whether to authenticate the SSL client is determined by the SSL server. As shown by blue arrows in **Figure 1-12**, if the server needs to authenticate the client, the following operations are required in addition to the SSL handshake process in which the client authenticates the server:

1. The server sends a CertificateRequest message to request the client to send its certificate to the server.
2. The client sends a Certificate message carrying its certificate and public key to the server. After receiving the message, the server verifies the validity of the certificate.
3. The client computes a hash for the master secret over handshake messages, encrypts the hash using its private key, and then sends a CertificateVerify message to the server.
4. The server computes a hash for the master secret over handshake messages, decrypts the received CertificateVerify message using the public key in the client's certificate, and compares the decrypted result with the computed hash. If the two values are the same, the client is authenticated.

- SSL handshake process for resuming a session

Figure 1-13 SSL handshake process for resuming a session



Asymmetric cryptography is used to encrypt keys and authenticate peer identities when session parameters are being negotiated and a session is being established. The computation workload is heavy, consuming a lot of system resources. To simplify the SSL handshake process, SSL allows resumed handshakes, as shown in **Figure 1-13**. The details are as follows:

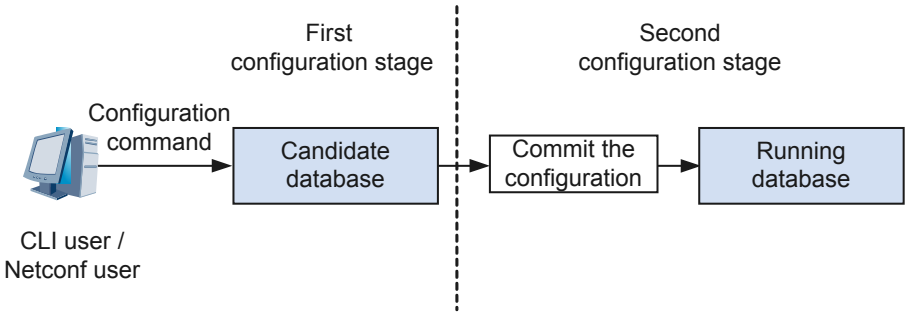
1. The client sends a ClientHello message. The session ID in this message is set to the ID of the session to be resumed.
2. If the server allows this session to be resumed, it replies with a ServerHello message with the same session ID. After that, the client and server can use the key and cipher suite of the resumed session without additional negotiation.
3. The client sends a ChangeCipherSpec message to notify the server that every subsequent message will be encrypted and a MAC will be computed based on the key and cipher suite negotiated for the original session.
4. The client computes a hash over handshake messages, uses the key and cipher suite negotiated for the original session to process the hash, and then sends a Finished message to the server so that the server can check whether the key and cipher suite are correct.
5. Similarly, the server sends a ChangeCipherSpec message to notify the client that every subsequent message will be encrypted and a MAC will be computed based on the key and cipher suite negotiated for the original session.
6. The server computes a hash over handshake messages, uses the key and cipher suite negotiated for the original session to process the hash, and then sends a Finished message to the client so that the client can check whether the key and cipher suite are correct.

1.2.6 Two-phase Validation Mode

Basic Principles

In the two-phase validation mode, the system configuration process is divided into two phases. The configuration takes effect after operations in two phases are complete. **Figure 1-14** shows the two phases of the system configuration process.

Figure 1-14 Networking diagram of two-phase validation mode



1. In the first phase, a user enters command lines and the system checks the data type, user level, and object to be configured, and checks whether there are repeated configurations. If syntax or semantics errors are found in the command line, the system displays a message on the terminal to inform the user of the error and the cause.
2. In the second phase, a user commits the configuration, and the system enters the configuration commitment phase. The system commits the configuration in the candidate database to the running database.
   - If the configuration takes effect, the system adds it to the running database.
   - If the configuration fails, the system prompts the user that the configuration is incorrect. In this situation, the user can enter the command line again or change the configuration.

The two-phase mode uses the following two databases:

- Running database:

  A configuration set that is used currently in the system.
- Candidate database:

  Mapping of the running database that the system generates for each user in the memory. Users can edit the configuration in the candidate database and then commit the edited configuration to the running database.

Validity Check

After entering the system view, the system assigns each user a candidate database. Users perform the configuration operation in their candidate databases and the system checks the validity of each user's configuration.

In two-phase validation mode, the system checks the configuration validity and prompts error messages. The system checks the validity of the following configuration items:

- Repeated configuration

The system checks whether configurations in the candidate databases are identical with that in the running databases.

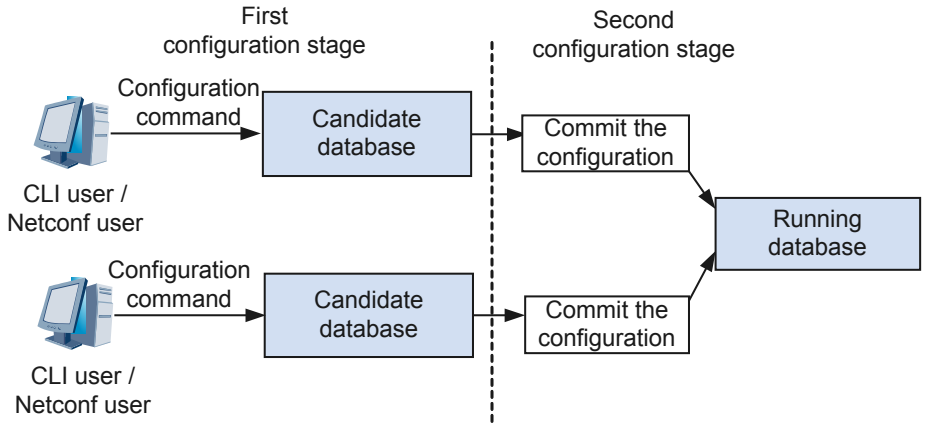
- If identical, the system does not commit the configuration to the running database and displays repeated configuration commands.
- If different, the system commits the configuration to the running database.

- Data type
- Commands available for each user level
- Existence of the object to be configured

Concurrent Operation of Multiple Users

As shown in **Figure 1-15**, multiple users can perform the configuration operation on the same device.

Figure 1-15 Networking diagram of concurrent configuration operations on the same device



Benefits

The two-phase mode brings the following benefits to users:

- Allowing several service configurations to take effect as a whole
- Allowing users to preview configurations in the candidate database
- Clearing configurations that do not take effect if an error occurs or the configuration does not meet the expectations
- Minimizing the impact of configuration procedures on current services

1.2.7 Configuration Rollback

Principles

After committing a configuration, the user will check the operation and impact of the configuration. If an error or a fault occurs, or the configuration procedure has unexpected impact on services, the system needs to return to the previous configuration.

Before the rollback function is worked out:

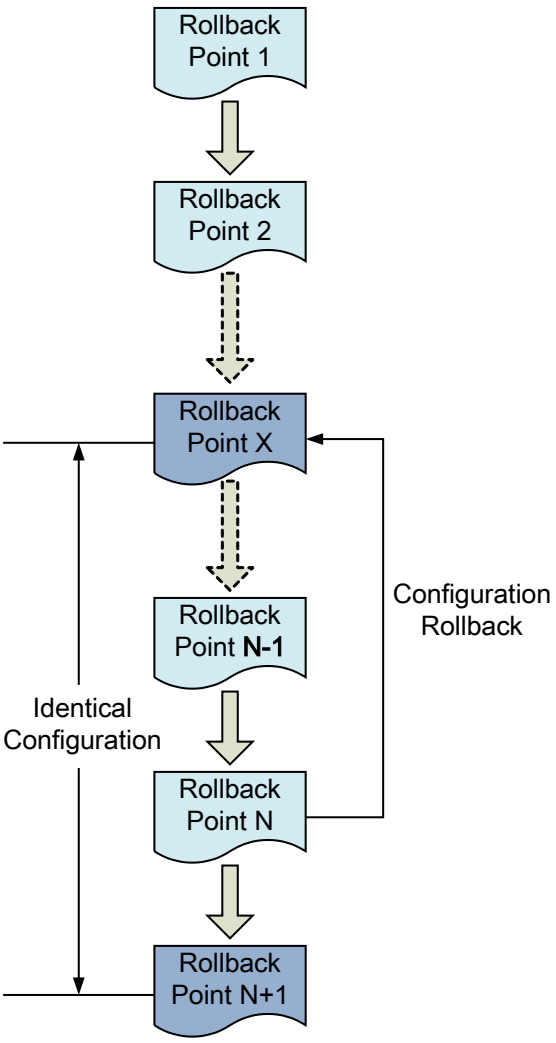
- If the system needs to returns to the previous configuration, the user cannot check the latest configurations and therefore restores the configuration based on the impression.
- If an error occurs or the impact of the configuration procedure has unexpected impact on services, the user has to delete or modify configurations one by one and is unable to perform the batch operation.

To solved theses problems, the rollback function is worked out, which enables the user to select a reasonable configuration rollback point based on the comparison between the current and target configurations. After that, the system can return to the specified configuration without system restart and service interruption.

As shown in **Figure 1-16**, a user performs N configurations and enters multiple command lines in each configuration. Each time the user commits a configuration, the system generates a related rollback point.

Rollback point N is the latest configuration that the user commits. When the user enables the configuration rollback function, the system returns to the configuration X and generates a new rollback point N+1. Configurations at rollback points N+1 and X are identical.

Figure 1-16 Networking diagram of configuration rollback



Related Concepts

- Rollback point: A historical configuration to which the system returns. Each time a user commits the configuration in the first or second phase, the system defines a rollback point at which the current configuration is saved. The rollback point can be checked using a command.
- Configuration rollback: A function enabling the system to return to the configuration at a specified rollback point in the case that an error occurs or the configuration procedure has unexpected impact on services. For example, a user has committed four configurations, generating four consecutive rollback points (a, b, c, and d). If an error is found in configuration b and the system needs to return to the previous configuration, the configuration rollback function enables the system to return to the configuration at rollback point a.

Basic Functions

Basic functions of configuration rollback include the rollback point generation, rollback point query, rollback point deletion, returning to a rollback point, and query of the configuration at a rollback point.

- Rollback point generation: Each time a user perform the configuration operation in the first phase and commit the configuration in the second phase, the system generates a related rollback point, recording the historical configuration operations. The system can generate a maximum of 50 rollback points and allows the user to add remarks to the rollback point.
- Rollback point query: A user can view rollback points generated in the system, including the label of each rollback point (a label uniquely identifies a rollback point), the user who commits the configuration that is regarded as the rollback point, type of the terminal where the configuration is performed (such as Console and VTY), and type of tools used in configuration (such as CLI, SNMP, and NETCONF), and the timestamp and remarks configured for the rollback point.
- Query of the configuration at a rollback point: A user can check the configuration at a rollback point and compares the current configuration with previous configurations. This enables the user to determine whether to perform the rollback operation after analyzing possible configuration changes and the impact on the system that the rollback operation brings.
- Rollback point configuration: A user can specify a rollback point to which the system returns. After that, all configurations at the rollback point will be restored, regardless of what changes the user has executed after the rollback point. For example, the created configuration will be deleted, the deleted configuration will be restored, and the modified configuration will return to the original configuration.
- Rollback point deletion: A user can delete the earliest rollback point in the system, clearing unnecessary information and saving system resources (such as the disk space).

Implementation

The implementation of the configuration rollback function is as follows:

1. A user sends a configuration rollback request and specifies the rollback point.
2. The system checks the validity of the specified rollback point and level of the user submitting before performing the configuration rollback operation.

3. The system compares the current and historical configurations, and saves the configuration procedures in a reverse order (from the current configuration to the historical configuration).
4. The system returns to the specified configuration.

Benefits

In terms of configuration security and system maintainability, the configuration rollback function brings the following benefits to users:

- If a user mistakenly runs certain commands (such as the **undo mpls** command) to delete service configurations in the system, all MPLS-related configurations will be deleted. In this situation, the user can enable the configuration rollback function to fast restore the configuration before the misoperation, minimizing the impact of the misoperation on the system.
- When an NMS user configures multiple network elements at the same time, some configurations may succeed and some may fail. In this situation, the NMS user can enable the configuration rollback function to restore the status of all network elements before the configuration, ensuring the configuration consistency of multiple network elements.
- Each time a feature configuration is committed, the system generates a rollback point. If there are multiple features but a user wants to test only one of them, the user needs to clear the impact of other features on the target feature. The configuration rollback function enables the system to return to the point where only the target feature is configured.

1.3 Applications

1.3.1 Applications of TFTP

Downloading or Uploading Files Through TFTP

A user can use TFTP to upload or download files to or from the server in a simple interaction environment. Currently, the device acts only as a TFTP client.

Figure 1-17 shows the networking of downloading or uploading files through TFTP.

Figure 1-17 Networking diagram of uploading or downloading files through TFTP

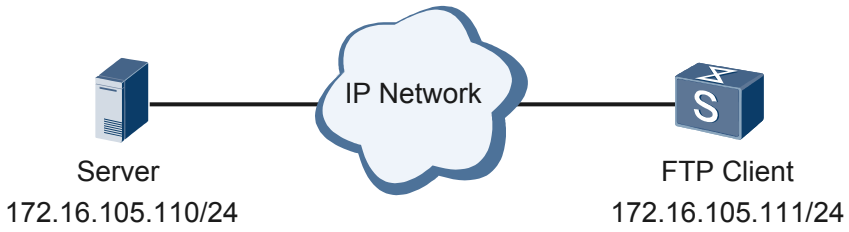
1.3.2 Applications of FTP

- Device functioning as an FTP client

  A user logs in to the FTP server from the device acting as an FTP client and then downloads files from the server to the storage device of the client.

In **Figure 1-18**, the device with the IP address of 172.16.105.111 acts as the FTP client. The user then can log in to the FTP server from the client through FTP.

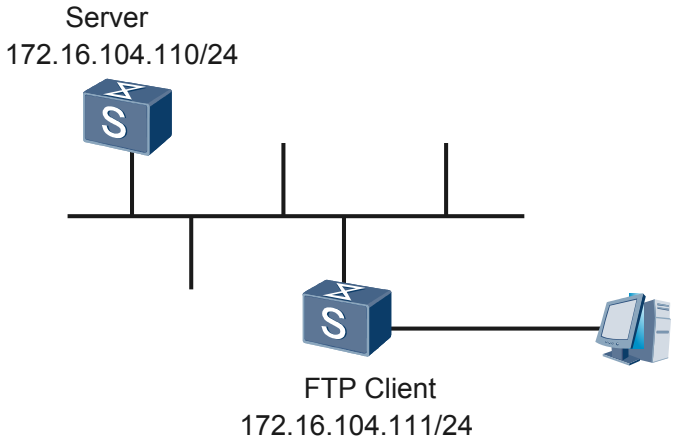
Figure 1-18 Networking diagram of the device functioning as an FTP client



- Device functioning as an FTP server

  A user logs in to the client from a HyperTerminal. The device functions as an FTP server, and downloads files from the FTP server. In **Figure 1-19**, the device with the IP address of 172.16.104.110 acts as the FTP server.

Figure 1-19 Networking diagram of the device functioning as an FTP server



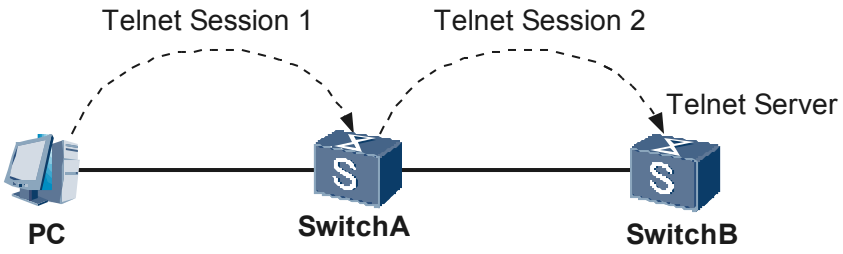
1.3.3 Telnet

- Device functioning as a Telnet client

  A user runs the Telnet client application on the PC so that it can log in to the device through Telnet and then manage the device.
- Device functioning as a Telnet server

  After running the emulation terminal program or Telnet client application on the PC to set up a connection with the device, a user can run the **telnet** command to log in to and manage other devices. In the scenario shown in **Figure 1-20**, Switch A acts as a Telnet client. It however, can act as a Telnet server simultaneously.

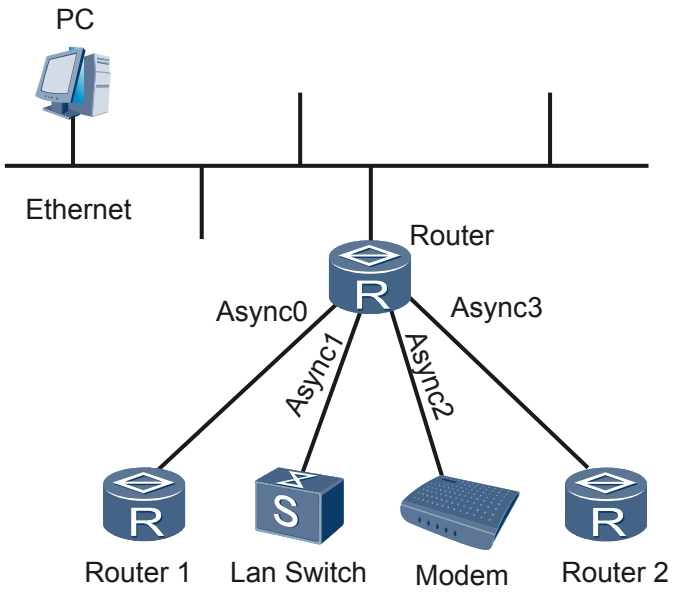
Figure 1-20 Networking diagram of the device acting as Telnet client



- Terminal redirection

  As shown in **Figure 1-21**, a user runs the Telnet client application and logs in to the device through a specified port. It then sets up a connection with the devices that are connected to the device through asynchronous serial interfaces. The typical application is that the devices directly connected with the device through asynchronous serial interfaces can be remotely configured and maintained.

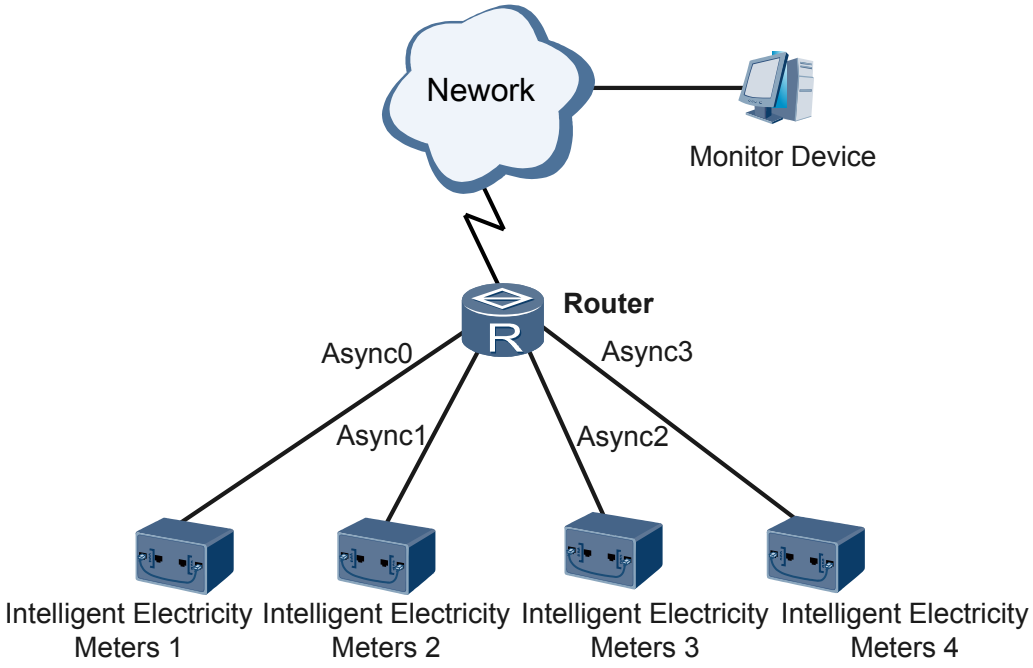
  Figure 1-21 Using redirection to connect to remote routers and switches (1)



  Managing terminals such intelligent electricity meters, intelligent water meters, and automatic teller machines

  As shown in **Figure 1-22**, the redirection function is enabled on the device. The device listens to the specified TCP port and receives data packets from the terminals through serial ports. After receiving data packets, the device encapsulates the packets into Ethernet frames so that they can be transmitted over an Ethernet network. This implements the remote data transmission and management on the terminals.

Figure 1-22 Using redirection to connect to remote intelligent terminals (2)



NOTE

Only the device providing asynchronous serial interfaces supports the terminal redirection service of Telnet.

1.3.4 Applications of SSH

- SSH for STelnet

  The STelnet client is based on SSH2 and the STelnet server is based on SSHv1.x and SSHv2. The client and the server set up a secure connection through negotiation. The client can then log in to the server using Telnet. **Figure 1-23** shows the networking of SSH for STelnet.

  Figure 1-23 Networking diagram of SSH for STelnet

  - A device can function as the STelnet server. Alternatively, it can function as the STelnet client to access other STelnet servers.
  - STelnet services can be enabled or disabled as required. By default, STelnet services are disabled. Enabling or disabling of STelnet services must be configured in global mode.
- SSH for SFTP

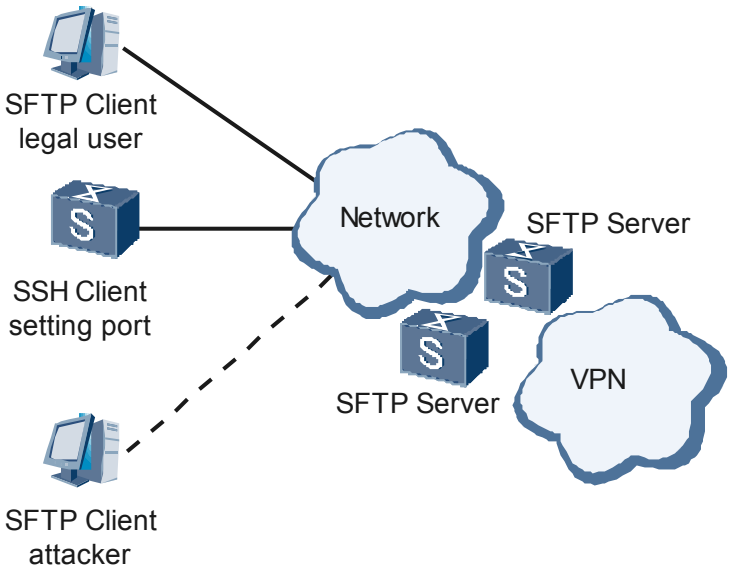
  Attackers cannot pass the authentication because they cannot provide the correct private key or password.. In addition, they cannot obtain the session key between another client

and the server. Only the server and the related client can decrypt packets exchanged between them. Even if attackers intercept packets exchanged between the server and the client, they cannot decrypt the packets. In this manner, the secure data transmission on the network is guaranteed.

SFTP is based on SSH2.0, which supports two authentication modes: password authentication and RSA authentication. To access the server using a client, an authorized user needs to enter the correct user name, password, and private key to pass the authentication of the server. After that, the user can use SFTP that is similar to FTP to manage remote file transfer on the network. The system uses the negotiated session key to encrypt user's data.

- A device can function as the SFTP server. Alternatively, it can function as the SFTP client to access other SFTP servers.
- SFTP services can be enabled or disabled as required. By default, SFTP services are disabled. Enabling or disabling of SFTP services must be configured in global mode.
- Different users are allowed to use SFTP to access different file directories. Users can access only the set SFTP directories. Available files for different users are isolated from each other.

Figure 1-24 Networking diagram of SSH for SFTP



- SSH for SCP

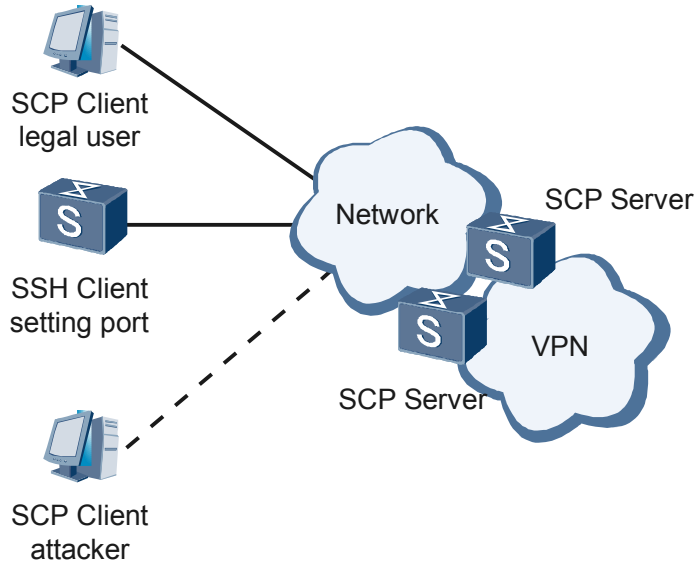
  Attackers cannot pass the authentication because they cannot provide the correct private key or password.. In addition, they cannot obtain the session key between another client and the server. Only the server and the related client can decrypt packets exchanged between them. Even if attackers intercept packets exchanged between the server and the client, they cannot decrypt the packets. In this manner, the secure data transmission on the network is guaranteed.

  SCP is based on SSH2.0, which supports two authentication modes: password authentication and RSA authentication. To access the server using a client, an authorized user needs to enter the correct user name, password, and private key to pass the authentication of the server. After that, the user can use SCP that is similar to FTP to manage

remote file transfer on the network. The system uses the negotiated session key to encrypt user's data

- A device can function as the SCP server. Alternatively, it can function as the SCP client to access other SCP servers.
- SCP services can be enabled or disabled as required. By default, SCP services are disabled. Enabling or disabling of SFTP services must be configured in global mode.

Figure 1-25 Networking diagram of SSH for SCP

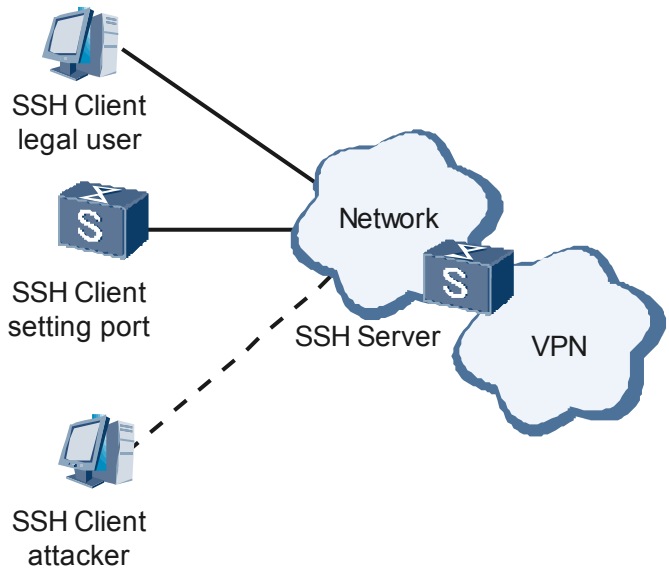


- SSH for the private network

  A device can function as the STelnet client, SFTP client, and SCP client. So that the client (device) on the public network can set up a Socket connection with the server in a VPN:

  - The STelnet client can access the SSH server on the private network.
  - The SFTP client can access the SSH server on the private network.
  - The SCP client can access the SSH server on the private network.

Figure 1-26 Networking diagram of SSH for the private network



- SSH for non-standard ports

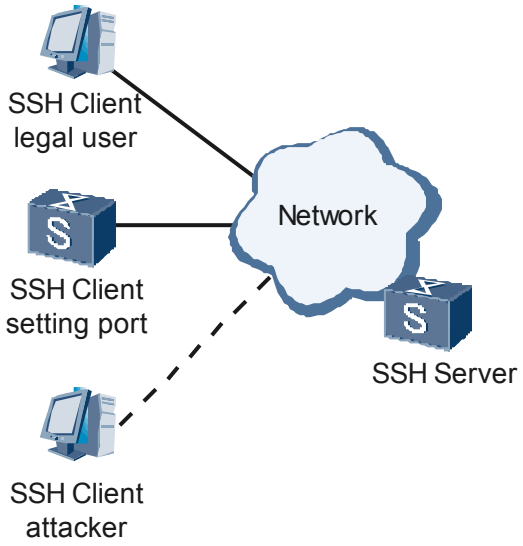
  The standard SSH listening port number is 22. If attackers continuously access this port, the available bandwidth and the performance of the server are reduced and authorized users cannot access this port.

  To address this problem, you can change the listening port on the SSH server to a non-standard port. The port change is invisible to attackers, so they continue to send socket connection requests to the standard listening port 22. If the SSH server detects that the connection requests are not forwarded to the actual listening port, it denies the requests.

  Only authorized clients can set up socket connections with the SSH server using non-standard ports on the server. The client and the server then negotiate the SSH version, algorithms and session keys. User authentication, session request, and interactive session are performed subsequently.

  SSH can be used on intermediate switching devices or edge devices on a network to secure the user access and device management.

Figure 1-27 Networking diagram of SSH for non-standard ports



- SSH for RADIUS

  If password authentication is required, SSH calls the interface provided by AAA in the same manner as FTP and Telnet. After user authentication is configured as RADIUS in AAA, when SSH authentication is enabled, the SSH server sends the authentication information (user name and password) to the RADIUS server (compatible with the HWTACACS server). The RADIUS server then sends the authentication result (pass or fail) to the SSH server for the SSH server to determine whether to establish a connection with the SSH client.

Figure 1-28 SSH for RADIUS

- SSH for ACLs

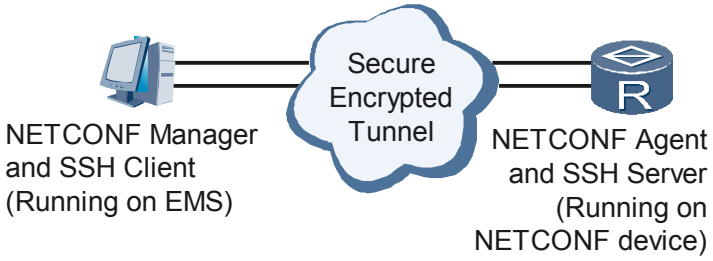
  The SSH server uses ACLs to limit the call-in and call-out rights of SSH users. This prevents unauthorized users from establishing TCP connections or entering the SSH negotiation phase, thus improving the security of the SSH server.

Figure 1-29 Networking diagram of SSH for ACLs

- SSH for SNETCONF

  NETCONF agent is an application running on top of SSH server. It uses the secure transport channel established by SSH. NETCONF is used to access configuration and state information and to modify configuration information, so the ability to access this protocol should be limited to users and system. To run NETCONF over SSH, the client first establishes an SSH transport connection using the SSH transport protocol. Client and server exchange keys for message integrity and encryption. Once the user is successfully authenticated, the client invokes the "ssh-connection" service which is also known as the SSH connection protocol. After the SSH connection service is established, the client opens a channel of type "session", which results in an SSH session. Once the SSH session is established, the user (or application) invokes SNETCONF as an SSH subsystem which is a feature of SSH version 2 (SSHv2). SSH Server makes sure of the reliability and packet sequencing for the data packets delivered for SNETCONF sub-system.

  Figure 1-30 Networking diagram of applying NETCONF on the SSH server



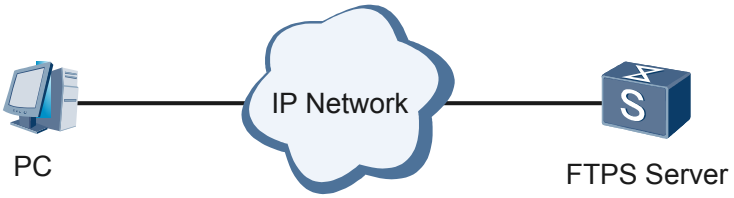
1.3.5 Applications of FTPS

Security Socket Layer (SSL) can be combined with FTP and HTTP to implement FTPS and HTTPS. In FTPS and HTTPS applications, a client and a server use SSL to authenticate each other and encrypt data to be transmitted. SSL implements secure device management.

- Logging in to the device that functions as an FTPS server from an operation terminal

  As shown in **Figure 1-31**, an SSL policy is configured on a device. After the FTPS server function is enabled on the device, the device functions as an FTPS server. You can log in to the server from a terminal on which the SSL-capable FTP client software is installed to securely operate files transmitted between the terminal and the server.

  Figure 1-31 Login to an FTPS server from a user terminal



- Logging in to an FTPS server from the device that functions as an FTPS client

  As shown in **Figure 1-32**:

  - The FTP client is configured with an SSL policy and loaded with a trusted-CA file to verify the validity of the server's digital certificate, sign the certificate to prevent eavesdropping and tampering, and manage the certificate and key.

- The device is configured with an SSL policy and enabled with the FTPS server function.

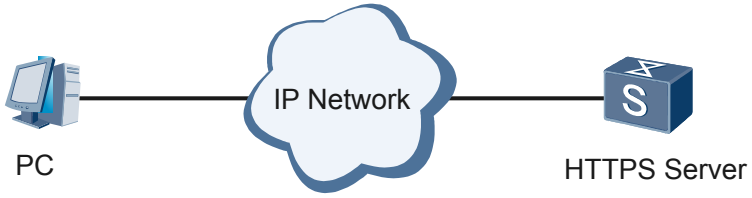
Figure 1-32 Login to an FTPS server from an FTPS client

If the FTPS client and server are routable, you can log in to the FTPS server from the FTPS client to remotely manage files.

- Logging in to the device from an operation terminal through HTTPS

  As shown in **Figure 1-33**, after a device is configured with an SSL policy and enabled with the HTTPS server function, the device functions as an HTTPS server. You can use a Web browser installed on the PC to log in to the HTTPS server to remotely manage the server using Web pages.

Figure 1-33 Login to an HTTPS server using a Web browser



1.4 References

The references of this feature are as follows:

Document	Description	Remarks
RFC 775	Directory oriented FTP commands	-
RFC 959	File Transfer Protocol	-
RFC 1635	How to Use Anonymous FTP	-
RFC 1350	The TFTP Protocol (Revision 2)	-
RFC 698	Telnet Extended ASCII Option	-
RFC 775	Directory oriented FTP commands	-
RFC 854	Telnet Protocol Specification	-
RFC 855	Telnet Option Specification	-
RFC 930	Telnet Terminal Type Option	-
RFC 1091	Telnet Terminal-Type Option	-

Document	Description	Remarks
RFC 2119	Key words for use in RFCs to Indicate Requirement Levels	-
RFC 4250	The Secure Shell (SSH) Protocol Assigned Numbers	-
RFC 4251	The Secure Shell (SSH) Protocol Architecture	-
RFC 4252	The Secure Shell (SSH) Authentication Protocol	-
RFC 4253	The Secure Shell (SSH) Transport Layer Protocol	This protocol supports neither compression nor the ssh-dss public key format.
RFC 4254	The Secure Shell (SSH) Connection Protocol	This protocol does not support some packets and functions, such as X11 forwarding, Env channel request packets, xon-xoff channel request packets, signal channel request packets, exit-status channel request packets, exit-signal channel request packets, and port forwarding.
RFC 4344	The Secure Shell (SSH) Transport Layer Encryption Modes	-
RFC 4345	Improved Arcfour Modes for the Secure Shell (SSH) Transport Layer	-
draft-ietf-secsh-publickey-subsystem-01	Authentication Mechanism that Is Based on Public Keys	-