SSH技术白皮书

关键词：SSH、SFTP、RSA、DSA、DES、AES、AAA

摘　要：SSH协议是一种在不安全的网络环境中，通过加密和认证机制，实现安全的远程访问以及文件传输等业务的网络安全协议。本文介绍了SSH的产生背景、安全机制、工作过程及组网应用，并重点描述了Comware实现的技术特色。

缩略语：

缩略语	英文全名	中文解释
AAA	Authentication, Authorization, Accounting	认证、授权、计费
AES	Advanced Encryption Standard	高级加密标准
DES	Data Encryption Standard	数据加密标准
DSA	Digital Signature Algorithm	数字签名算法，非对称密钥算法的一种
FTP	File Transfer Protocol	文件传输协议
MAC	Message Authentication Code	消息验证码
RSA	Rivest Shamir and Adleman	非对称密钥算法的一种
SFTP	Secure File Transfer Protocol	安全的文件传输协议
SSH	Secure Shell	安全外壳
Stelnet	Secure Telnet	安全的Telnet

目 录

1 概述

1.1 产生背景

SSH协议出现之前，在网络设备管理上广泛应用的一种方式是Telnet。

Telnet协议的优势在于通过它可以远程地登录到网络设备上，对网络设备进行配置，为网络管理员异地管理网络设备提供了极大的方便。

但是，Telnet协议存在三个致命的弱点：

- 数据传输采用明文方式，传输的数据没有任何机密性可言。
- 认证机制脆弱。用户的认证信息在网络上以明文方式传输，很容易被窃听；Telnet 只支持传统的密码认证方式，很容易被攻击。
- 客户端无法真正识别服务器的身份，攻击者很容易进行“伪服务器欺骗”。

SSH协议正是为了克服Telnet协议存在的问题而诞生的。

网络中另外一个广泛应用的协议——FTP，也面临着和Telnet相同的问题。为了解决FTP应用中的安全性问题，在SSH协议基础上扩展了对FTP安全性的支持，即SFTP。

1.2 技术优点

SSH协议是一种在不安全的网络环境中，通过加密和认证机制，实现安全的远程访问以及文件传输等业务的网络安全协议。

SSH协议具有以下一些优点：

- 数据传输采用密文的方式，保证信息交互的机密性；
- 用户的认证信息以密文的方式传输，可以有效地防止用户信息被窃听；
- 除了传统的密码认证，SSH 服务器还可以采用多种方式对用户进行认证（如安全性级别更高的公钥认证），提高了用户认证的强度；
- 客户端和服务器端之间通信使用的加解密密钥，都是通过密钥交互过程动态生成的，可以防止对加解密密钥的暴力猜测，安全性级别比手工配置密钥的方式高；
- 为客户端提供了认证服务器的功能，可以防止“伪服务器欺骗”。

2 协议总体框架

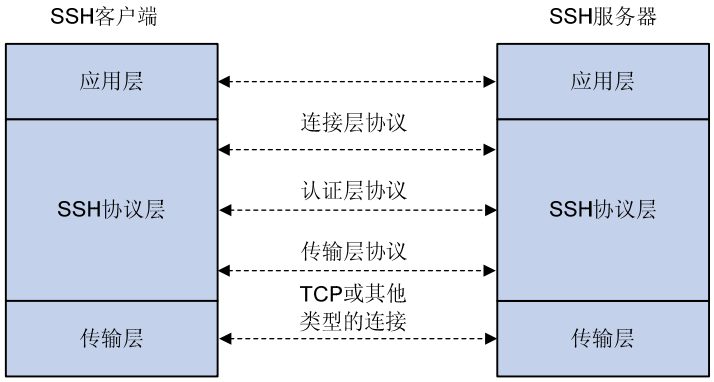

图1 SSH协议框架结构

如图1所示，SSH协议采用客户端/服务器架构，分为传输层、认证层和连接层。

2.1 传输层协议

传输层协议主要用来在客户端和服务器之间建立一条安全的加密通道，为用户认证、数据交互等对数据传输安全性要求较高的阶段提供足够的机密性保护。

传输层提供了如下功能：

- 数据真实性检查
- 数据完整性检查
- 为客户端提供了对服务器进行认证的功能

传输层协议通常运行在TCP/IP连接之上（服务器端使用的知名端口号为22），也可以运行在其他任何可以信赖的数据连接之上。

2.2 认证层协议

认证层协议运行在传输层协议之上，完成服务器对登录用户的认证。

2.3 连接层协议

连接层协议负责在加密通道上划分出若干逻辑通道，以运行不同的应用。它运行在认证层协议之上，提供交互会话、远程命令执行等服务。

3 协议安全性机制

3.1 保证数据传输的机密性

SSH协议需要解决Telnet协议明文传输的缺陷，它通过以下两方面保证数据传输的机密性：

- 在通信双方之间建立加密通道，保证传输的数据不被窃听；
- 使用密钥交换算法保证密钥本身的安全。

3.1.1 使用加密通道保证数据不被窃听

所谓加密通道，是指发送方在发送数据前，使用加密密钥对数据进行加密，然后将数据发送给对方；接收方接收到数据后，利用解密密钥从密文中获取明文。

加解密算法分为两类：

- 对称密钥算法：数据加密和解密时使用相同的密钥和相同的算法。
- 非对称密钥算法：数据加密和解密时使用不同的密钥，一个是公开的公钥，一个是由用户秘密保存的私钥。

由于非对称密钥算法比较耗时，一般多用于数字签名以及身份认证。SSH加密通道上的数据加解密使用对称密钥算法，目前主要支持的算法有DES、3DES、AES等，这些算法都可以有效地防止交互数据被窃听，而且由于采用了初始化向量保护，可以防止类似于密码流复用等密码分析工具的攻击。

3.1.2 使用密钥交换算法保证密钥本身的安全

对称密钥算法要求解密密钥和加密密钥完全一致，才能顺利从密文中解密得到明文。因此，要建立加密通道，必须先在通信双方部署一致的加解密密钥。部署加解密密钥的方式有多种：手工配置和第三方机构分配等。手工配置的方式扩展性差，只适合一些小型的本地网络；使用第三方机构分配密钥的方式，需要额外的第三方服务器，而且密钥在网络中传输容易被窃听。

SSH协议使用一种安全的方式在通信双方部署密钥：密钥交换算法。利用密钥交换算法可以在通信双方动态地产生密钥，这个密钥只能被通信的双方获得，任何第三者都无法窃听，这就在源头上保证了加解密使用密钥的安全性，很好地解决了密钥分发问题。

密钥交换算法的基本原理是：

(1) 客户端随机选择一个私钥 Xc，1<Xc<p-1，计算出 Yc=g^Xc mod p，将计算出的 Yc 发送给服务器端。其中，p 是一个很大的素数，g 是 p 的素根。p 和 g 是双方共有的一对参数，协议允许双方通过协商获得相同的 p 和 g 参数。

(2) 服务器也随机生成一个私钥 Xs，1<Xs<p-1，计算出 Ys＝g^Xs mod p，也将计算出的 Ys 发送给客户端。

(3) 服务器接收到客户端发送过来的 Yc，按照下面的公式计算出密钥：

K ＝ （Yc）^Xs mod p

(4) 客户端收到服务器端发送过来的 Ys，同样按照下面的公式计算出密钥：

K ＝ （Ys）^Xc mod p

通过上面的方法，客户端和服务器端就可以获得相同的密钥。

由上面的分析可以看出，密钥交换算法的安全性建立在计算离散对数的难度之上。算式Y=g^x mod p中，由X计算Y是很容易的，但是要由Y计算X是非常困难的。在密钥交换算法中对外公开的只有p、g、Yc和Ys，私钥Xc和Xs是保密的，其他用户即便获取了p、g、Yc和Ys也很难推断出私钥Xc和Xs，从而保证了密钥的安全性。

密钥交换算法具有如下优势：

- 扩展性更好，不需要网络管理员的多余配置；
- 交换得到的密钥是保存在内存中，不易被读取和篡改；
- 每个连接都会动态生成一次新的密钥，安全性更高。

📖 **说明：**

素数和素根是离散数学中的术语，详细定义可以参考相关书籍。

3.2 完善的用户认证机制

为了防止非法用户登录到设备，对设备进行破坏性配置，SSH协议需要支持多种用户认证方式，提高对用户认证的强度。常用的用户认证方式包括密码认证和公钥认证，Comware V5平台还定义了一种新的认证方式——password-publickey认证。

3.2.1 密码认证

SSH协议可以利用AAA提供的认证功能，完成对登录用户进行密码认证。根据AAA采取的认证策略的不同，密码认证分为本地认证和远程认证两种方式。

1. 本地认证

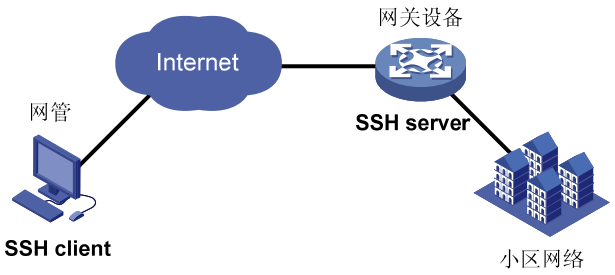

图2 本地认证组网示意图

本地认证是指在SSH服务器本地保存用户的信息，认证过程在本地完成。如图2所示，网管人员通过网络远程登录到小区的网关设备上，对设备进行相关配置。网关设备作为SSH服务器根据AAA本地用户数据库中的用户信息对登录用户进行身份认证。

2. 远程认证

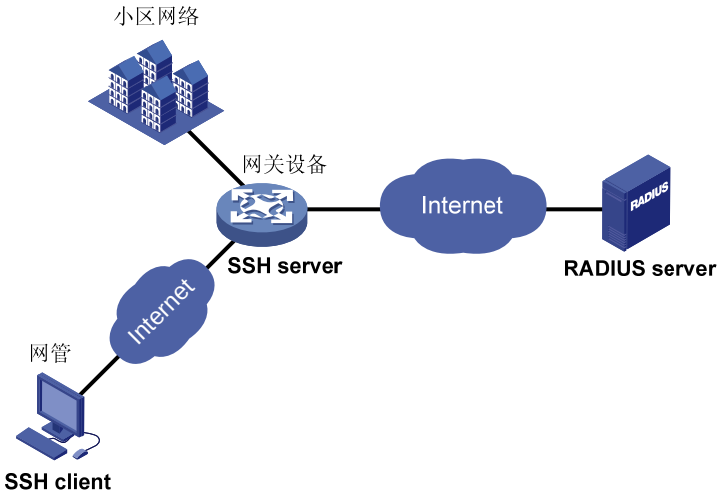

图3 远程认证组网示意图

远程认证是指将用户信息保存在远端的RADIUS等认证服务器上，认证过程在本地设备和远程认证服务器之间完成。如图3所示，网络管理员通过网络远程登录到小区的网关设备上，对网关设备进行配置。网关设备作为SSH服务器，将登录用户的认证信息，传递到远程认证服务器（如RADIUS服务器）上，根据认证服务器返回的对该用户的认证结果决定是否允许该用户登录。

采用远程认证方式可以将某个区域内所有用户的配置、管理都集中到远程认证服务

器上，便于对用户的集中管理。此外，用户的身份信息等关键数据都保存在认证服务器上，在很大程度上提高了用户信息的安全性。

当然，采用远程认证方式，需要保证网关设备与远程认证服务器之间的连接是绝对安全的，这样才能保证用户信息的安全。

3.2.2 公钥认证

由于密码认证方式的认证强度较弱，SSH协议引入了公钥认证方式。目前，设备上可以利用RSA和DSA两种非对称密钥算法实现公钥认证。

公钥认证的过程分为两个部分：：

(1) 公钥验证：客户端首先将自己本地密钥对的公钥部分，按照字符串格式发送到服务器。服务器使用本地保存的客户端公钥，与报文中携带的客户端公钥进行比较，验证客户端持有公钥的正确性。

(2) 数字签名验证：如果公钥验证成功，客户端继续使用自己本地密钥对的私钥部分，对特定报文进行摘要运算，将所得的结果（即数字签名）发送给服务器，向服务器证明自己的身份；服务器使用预先配置的该用户的公钥，对客户端发送过来的数字签名进行验证。

公钥验证和数字签名验证中任何一个验证失败，都会导致本次公钥认证失败。

公钥认证具有以下优势：

- 认证强度高，不易受到“暴力猜测”等攻击方式的影响。
- 具有较高的易用性。一次配置成功后，后续认证过程自动完成，不需要用户记忆和输入密码。

但是，公钥认证还存在以下缺点：

- 公钥认证配置比密码认证复杂。
- 公钥认证只能区分私钥，如果要实现充分的“粒度”，则必须为每一个用户创建一个私钥，相应地需要在服务器上为每一个用户配置对应的公钥，对于有多个用户使用同一个终端登录的情况，这种方式显然是不适合的，也是不必要的。

3.2.3 password-publickey认证

Comware V5平台定义了一种新的认证方式：password-publickey认证，这种认证方式要求用户同时完成公钥认证和密码认证，只有两种认证都成功后，才能通过服务器端的认证。

password-publickey认证方式充分利用了密码认证和公钥认证的优势，具有如下优点：

- 同时要求用户进行两种认证，安全性更高。在公钥上绑定密码，可以防止由于 SSH 客户端上的安全性隐患，影响到 SSH 服务器的安全性。
- 可以在一对公私密钥对的基础上，通过设置不同的密码，配置不同的用户，为这些用户设置不同的权限，方便了管理员的配置。
- 既利用了公钥认证的安全性，又节约了存储成本和配置成本。公钥认证实现方案中，要实现对用户认证的充分“粒度”，就必须为每个用户都配置一对密钥对，在 password-publickey 认证方式，多个用户可以共用一对密钥对。
- 在公钥认证技术上应用密码认证，结合 Comware V5 原有实现（结合 AAA 实现对用户的认证）可以很方便地同远程认证服务器配合，从而利用远程服务器上的诸多功能。

3.3 对“伪服务器欺骗”的防御

用户认证机制只实现了服务器端对客户端的认证，而客户端无法认证服务器端，因此存在着“伪服务器欺骗”的安全隐患。

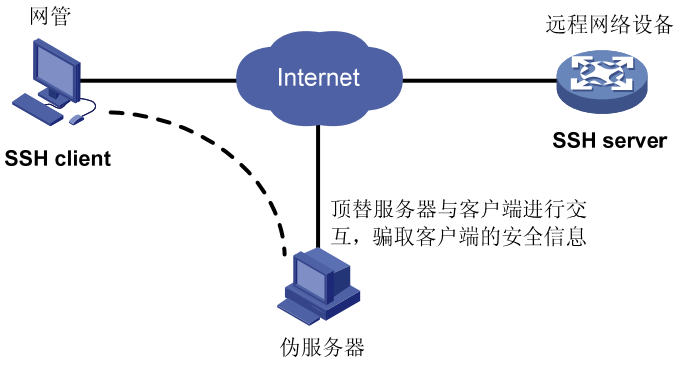

图4 伪服务器欺骗

如图4所示，当客户端主机需要与服务器建立连接时，第三方攻击者冒充真正的服务器，与客户端进行数据交互，窃取客户端主机的安全信息，并利用这些信息去登录真正的服务器，获取服务器资源，或对服务器进行攻击。

为了防止类似这样的伪服务器欺骗，SSH协议支持客户端对服务器端进行认证。服务器端在密钥交换阶段，使用自己的私钥对一段固定格式的数据进行数字签名，并将该签名发往客户端，客户端使用本地保存的服务器公钥，对签名进行鉴别，从而达到认证服务器的目的。

客户端对服务器进行认证的基础是本端存储的服务器公钥是真实服务器的公钥。因此，需要保证客户端获取的服务器公钥是正确的。目前，Comware V5支持两种获取服务器公钥的方式：

- 手工配置：既通过手工命令方式，将服务器公钥配置在本地，并在本地建立服务器名称和公钥之间的关联；
- 首次认证：SSH协议交互过程中，服务器会将自己的公钥通过协议报文发送到客户端，Comware V5 借用这个特点，允许SSH客户端配置首次认证功能，即SSH客户端第一次登录SSH服务器时，可以从协议报文中获该服务器端公钥并保存到本地，作为后续认证的依据。详细介绍请参见“5.2 首次认证”。

4 协议工作过程

SSH的报文交互主要有以下几个阶段：

- 连接建立
- 版本协商
- 算法协商
- 密钥交换
- 用户认证
- 服务请求
- 数据传输和连接关闭

4.1 连接建立

SSH服务器端在22端口侦听客户端的连接请求，接收到客户端的连接建立请求后，与客户端进行三次握手，建立起一条TCP连接，后续的所有报文交互都在这个TCP连接之上进行。

4.2 版本协商

TCP连接建立之后，服务器和客户端都会向对端发送自己支持的版本号。服务器端和客户端收到对端发送过来的版本后，与本端的版本号进行比较，双方都支持的最高版本号即为协商出的版本号。

说明：

SSH1.99 为特殊的版本号，这个版本既可以与 SSH2.0 版本互通，又可以与 SSH1.5 版本互通。

版本协商成功后，进入下一个阶段，即算法协商阶段。否则，中断连接。

4.3 算法协商

SSH协议报文交互需要使用多种算法：

- 用于产生会话密钥的密钥交换算法，包括 diffie-hellman-group-exchange-sha1、diffie-hellman-group1-sha1 和 diffie-hellman-group14-sha1 算法等。
- 用于数据信息加密的加密算法，包括 3des-cbc、aes128-cbc 和 des-cbc 加密算法等。
- 用于进行数字签名和认证的主机公钥算法，包括 RSA 和 DSA 公钥算法等。
- 用于数据完整性保护的 MAC 算法，包括 hmac-md5、hmac-md5-96、hmac-sha1 和 hmac-sha1-96 算法等。

由于各种客户端和服务器对这些算法的支持情况不一样，因此需要通过算法协商阶段，使客户端和服务器协商出双方都支持的算法。

SSH协议的算法协商过程为：

(1) 客户端和服务器端都将自己支持的算法列表发送给对方；

(2) 双方依次协商每一种算法（密钥交换算法、加密算法等）。每种算法的协商过程均为：从客户端的算法列表中取出第一个算法，在服务器端的列表中查找相应的算法，如果匹配上相同的算法，则该算法协商成功；否则继续从客户端算法列表中取出下一个算法，在服务器端的算法列表中匹配，直到匹配成功。如果客户端支持的算法全部匹配失败，则该算法协商失败。

(3) 某一种算法协商成功后，继续按照上述方法协商其他的算法，直到所有算法都协商成功；如果某一种算法协商失败，则客户端和服务器之间的算法协商失败，服务器断开与客户端的连接。

以加密算法为例，算法的协商方式为：

表1 加密算法协商举例

客户端的加密算法列表	服务端的加密算法列表	最后协商出的加密算法
3des、3des-cbc、aes128-cbc	aes128-cbc、 3des-cbc、des-cbc	3des-cbc

4.4 密钥交换

加密算法协商成功后，为了保证加解密密钥的安全性，SSH利用密钥交换算法在通信双方安全动态地生成和交互数据的加解密密钥，并能够有效防止第三方窃听加解密密钥。密钥交换算法的详细介绍请参见“3.1.2 使用密钥交换算法保证密钥本身的安全”。

4.5 用户认证

密钥交换完成之后，进入用户认证阶段。

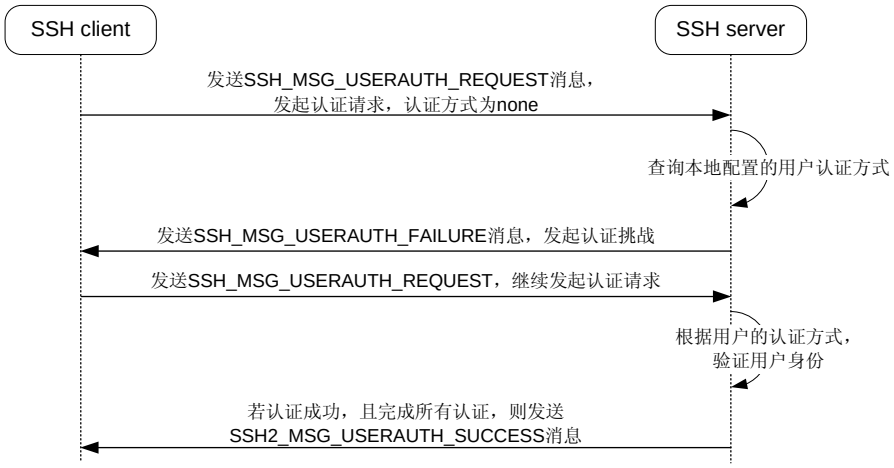

图5 用户认证过程

如图5所示，用户认证过程为：

(1) 客户端向服务器端发送认证请求报文，其中携带的认证方式为“none”。

(2) 服务器收到 none 方式认证请求，回复认证挑战报文，其中携带服务器支持、且需要该用户完成的认证方式列表。

(3) 客户端从服务器发送给自己的认证方式列表中选择某种认证方式，向服务器发起认证请求，认证请求中包含用户名、认证方法、与该认证方法相关的内容：

- 密码认证方式中，内容为用户的密码；
- 公钥认证方式中，内容为用户本地密钥对的公钥部分（公钥验证阶段）或者数字签名（数字签名验证阶段）。

(4) 服务器接收到客户端的认证请求，验证客户端的认证信息：

- 密码认证方式：服务器将客户端发送的用户名和密码信息，与设备上或者远程认证服务器上保存的用户名和密码进行比较，从而判断认证成功或失败。
- 公钥认证方式：服务器对公钥进行合法性检查，如果不合法，则认证失败；否则，服务器利用数字签名对客户端进行认证，从而判断认证成功或失败。

(5) 服务器根据本端上该用户的配置，以及用户认证的完成情况，决定是否需要客户端继续认证，分为以下几种情况：

- 如果该种认证方式认证成功，且该用户不需要继续完成其他认证，则服务器回复认证成功消息，认证过程顺利完成。
- 如果该种认证方式认证成功，但该用户还需要继续完成其他认证，则回复认证失败消息，继续向客户端发出认证挑战，在报文中携带服务器需要客户端继续完成的认证方式列表；
- 如果该种认证方式认证失败，用户的认证次数尚未到达认证次数的最大值，服务器继续向客户端发送认证挑战；
- 如果该种认证方式认证失败，且用户的认证次数达到认证次数的最大值，用户认证失败，服务器断开和客户端之间的连接。

说明：

认证挑战是指 SSH 服务器将用户需要完成的认证方式列表发送给用户，要求用户从中选择一种，继续发起认证请求。用户未完成认证时，服务器都会通过这种发送认证列表的方式，要求用户进行认证。

4.6 服务请求

SSH协议支持多种应用服务。用户成功完成认证后，SSH客户端向服务器端发起服务请求，请求服务器提供某种应用。

服务请求的过程为：

(1) 客户端发送 SSH_MSG_CHANNEL_OPEN 消息，请求与服务器建立会话通道，即 session；

(2) 服务器端收到 SSH_MSG_CHANNEL_OPEN 消息后，如果支持该通道类型，则回复 SSH_MSG_CHANNEL_OPEN_CONFIRMATION 消息，从而建立会话通道；

(3) 会话通道建立之后，客户端可以申请在通道上进行 shell 或 subsystem 类型的会话，分别对应 SSH 和 SFTP 两种类型的服务。

4.7 数据传输和连接关闭

服务请求成功，建立会话后，服务器和客户端可以在该会话上进行数据的传输。客户端将要执行的命令加密后传给服务器，服务器接收到报文，解密后执行该命令，将执行的结果加密发送给客户端，客户端将接收到的结果解密后显示到终端上。

通信结束或用户空闲时间超时后，关闭会话，断开连接。

5 Comware V5平台实现的技术特色

5.1 支持两种应用

Comware V5目前实现了SSH协议的两种应用：SSH和SFTP。

5.1.1 SSH

SSH在Comware V5上也称为Stelnet，它使用SSH协议提供的安全通道，实现对网络设备的远程访问，是SSH协议最基础的一种应用。

目前，Comware V5同时支持SSH Client和SSH Server。其中SSH Client基于SSH2.0版本的实现，而SSH Server实现了两个版本：SSH1.5和SSH2.0版本。由于SSH1.5和SSH2.0版本是互不兼容的，所以所有基于Comware的SSH Client只支持登录2.0版本的服务器，而SSH Server同时兼容1.5和2.0的客户端登录。

5.1.2 SFTP

SFTP利用SSH协议提供的安全通道，实现对网络设备的远程文件操作，是SSH协议中规定的一项扩展应用。

目前，Comware V5平台同时支持SFTP Client和SFTP Server。

5.2 首次认证

为了实现客户端第一次登录服务器时，对服务器进行身份认证，Comware作为客户端不仅支持以手工配置方式获取服务器的公钥，还支持从协议报文中获取公钥并保存到本地，以作为后续认证的依据，这个功能称为首次认证。

- 如果不支持首次认证，则首次登录服务器时，客户端需要认证服务器的身份，这样就要求提前在客户端上配置登录服务器的公钥，否则客户端将拒绝访问该服务器。
- 如果支持首次认证，则当 SSH 客户端首次访问服务器，而客户端没有配置服务器端的主机公钥时，用户可以选择继续访问该服务器，并选择在客户端保存该主机公钥；当用户下次访问该服务器时，就以保存的主机公钥来认证该服务器。

5.3 支持password-publickey认证方式

Comware上定义了新的认证方式：password-publickey认证方式。详细介绍请参见“3.2.3 password-publickey认证”。

6 典型组网应用

SSH在不安全的网络环境中，通过协议的加密和认证机制，实现了安全的远程访问管理、文件操作等应用。如图6、图7所示，SSH客户端既可以通过本地连接，也可以通过广域网连接与SSH服务器建立SSH通道，实现对SSH服务器的访问和控制。

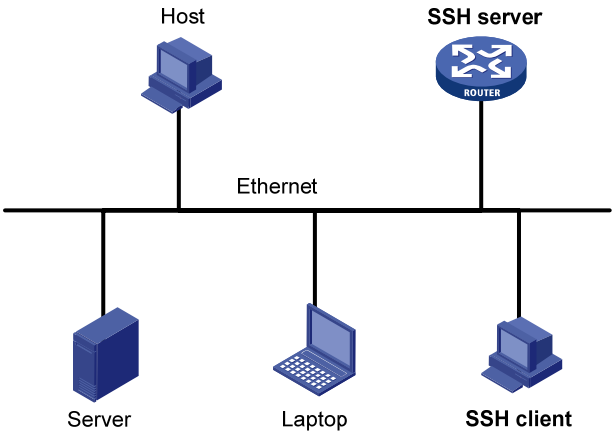

图6 通过本地连接建立SSH通道

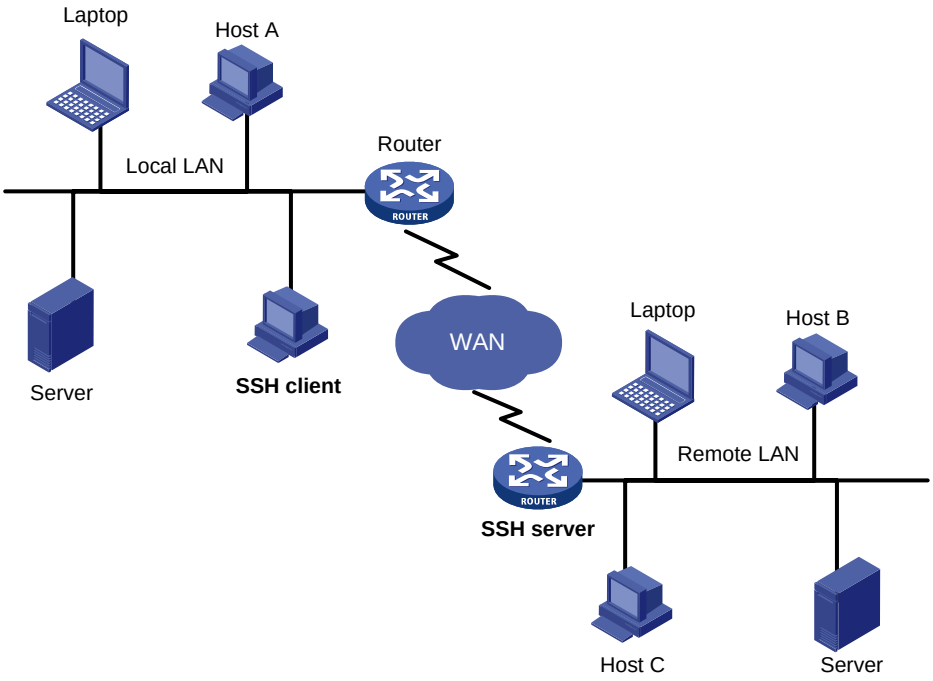

图7 通过广域网连接建立SSH通道

7 参考文献

- 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
- RFC 4254：The Secure Shell (SSH) Connection Protocol