



# Understanding Secure Shell Host Keys



VANDYKE  
SOFTWARE

4848 tramway ridge dr. ne  
suite 101  
albuquerque, nm 87111

505 - 332 -5700

[www.vandyke.com](http://www.vandyke.com)

## Understanding Host Keys

Think about the last time you faxed personal or company information to someone for the first time. Did you wonder if the number you were sending this information to was the right one? Unlike a phone call, where no personal information is exchanged until you have identified who you are speaking to, when you send a fax you might wonder where your information is ending up. When using the public network (internet), verifying that the server being connected to is the “right number” is taken for granted far too often. In this white paper, we will talk about the importance of knowing that the server you (or one of your end users) is connecting to is the “right number” and how Secure Shell server host keys are used to verify a server’s identity.

This paper assumes the reader has a general familiarity with the Secure Shell protocol. For more information, refer to our Secure Shell Overview white paper which can be read online or downloaded from our web site:

[http://www.vandyke.com/solutions/ssh\\_overview/index.html](http://www.vandyke.com/solutions/ssh_overview/index.html).

Here is a brief excerpt from that white paper’s introduction:

*Secure Shell (SSH) provides an open protocol for securing network communications which is less complex and expensive than hardware-based VPN solutions. Secure Shell client/server solutions provide command shell, file transfer, and data tunneling services for TCP/IP applications. SSH connections provide highly secure authentication, encryption, and data integrity to combat password theft and other security threats.*

### Introduction

Users and administrators turn to Secure Shell for many reasons. Some need to replace Telnet or FTP. Others are looking to move away from simple passwords to public-key, Kerberos, or keyboard-interactive authentication. Still others are looking for a low cost alternative to VPNs. Whatever the reason for using Secure Shell, every user and administrator of Secure Shell needs to understand host keys. Understanding what host keys are, how they work, and the security they provide can reduce confusion and possible frustration about administering and using this authentication tool for both administrators and end users.

Failure to understand host keys can cause many problems. Some users may be frightened or confused by messages when a new or changed host key is encountered. This can result in additional and unnecessary support costs. Even worse, improper handling of host keys can lead to a compromised system.

### What vulnerabilities do host keys help address?

**Man-in-the middle attacks:** When you connect to a remote host, if you cannot reliably verify that the host key is from the host you intend to connect to, you risk the possibility that an adversary has placed a server pretending to be the Secure Shell server between you and the final destination. The server you’re actually connecting to is a “man-in-the-middle”. This man-in-the-middle is able to see both the username and password information you transmit as you attempt to authenticate. Once intercepted, a man-in-the-middle can use this information to establish a connection with the remote server and see all traffic between you and the remote server.

Depending on your network topology, the risk of man-in-the-middle attack can vary widely. If users and administrators take some very basic steps, the risk of a man-in-the-middle attack can be significantly reduced.

**How web servers prevent man-in-the-middle attacks:** This uncertainty about the authenticity of a remote host is not a unique problem. Most of us have connected to a secure web server at one time or another. We look down at the bottom corner of the browser and we see a little padlock. It gives us a warm fuzzy feeling that we know we have connected to a secure web server. So, how do secure web servers solve this problem? A secure web server uses a certificate issued by a trusted third-party Certificate Authority (CA) such as VeriSign® and the client is responsible for checking that certificate upon connecting. In Internet Explorer and other browsers, you can review a list of certificate authorities that are deemed to be trusted. When you initially connect to the web server, the browser checks the certificate it receives from the remote server to see that it has been signed by one of the known trusted authorities. It also checks to see if the certificate has been revoked or expired. If a certificate has been revoked or expired, a dialogue will pop up indicating there is a problem with the certificate. You can also get a pop-up dialog if the certificate doesn't match the host you are connecting to.

In general, Secure Shell servers don't have the same type of key infrastructure that web servers depend on. Most Secure Shell servers rely on host keys that are created automatically by the server after installation. And these host keys can't easily be verified the first time a client connects.

### **What is the purpose of the host key?**

A host key is the server's public key. The host key is used by the client to decrypt an authentication message sent from the server when connecting. The basic purpose of the host key is to ensure that when you connect to a remote host, it is actually the host that you intended to connect to. Unfortunately, this presents something of a catch-22, as described in the following illustrations. If you've never connected to the host before, how do you know the host you are connecting to is the right one?

In Secure Shell, host keys can be used for host-based authentication, but this paper will not be addressing host-based authentication since it is not widely used.

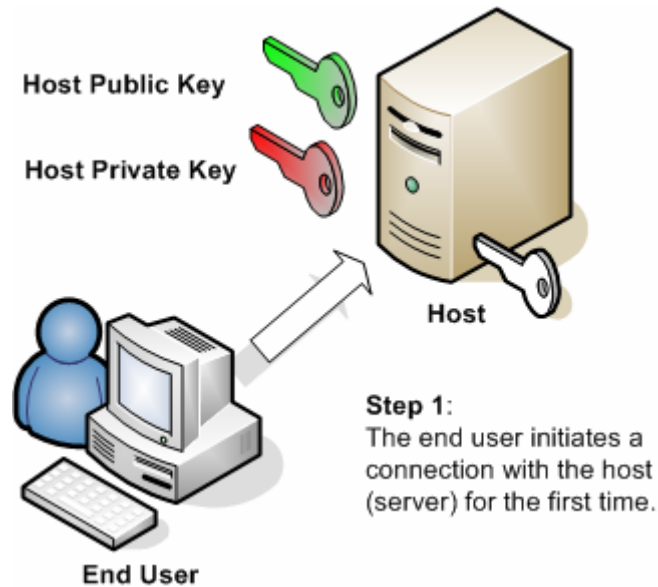
### **Creating host keys**

Creating a host key for a Secure Shell server is usually done only once. The server software creates the host key automatically during installation and configuration. Less commonly, an administrator can elect to manually generate a host key and select the encryption algorithm and key length. For those of you familiar with Secure Shell's public-key authentication, the choice of algorithms (DSA or RSA) and the key length (usually between 1024 and 2048 bits long) is identical to the options for those user authentication keys.

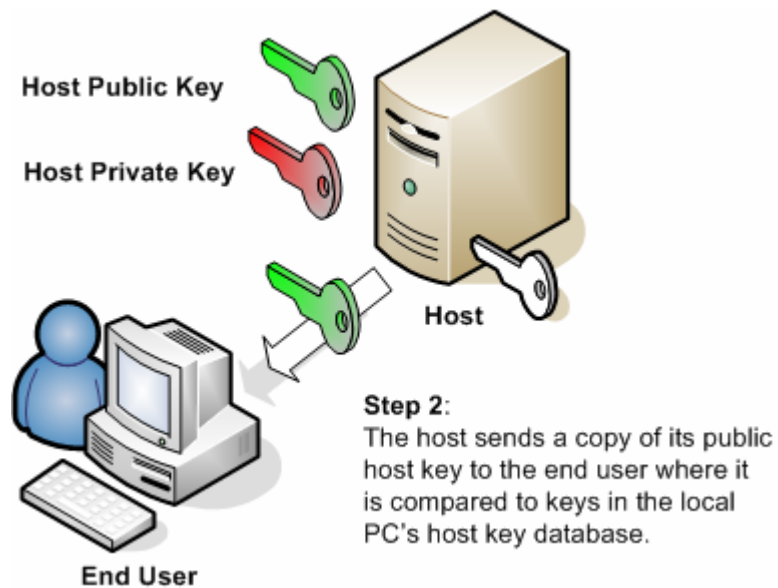
A host key consists of two components, a private and a public component. The public component is sent to the client when the client connects. **The private component should be protected so that only the administrator and the Secure Shell server have access to it.** This cannot be overemphasized. If an adversary acquires a copy of the private host key, it can be used to impersonate your server with complete impunity.

## Accepting a new host key

The illustrations below describe how the client application and the host perform a host key exchange when connecting for the first time.



Once the connection is established, the key exchange takes place before any personal data is sent by the client application.



If the host key does not match an existing key in the client application's host key database, a challenge message is generated by the client application. An example text message and dialog box are shown below.

```
The host key database does not contain an entry for the hostname
myserver, which resolved to 192.168.0.29, port 22.
```

```
It is recommended you verify your host key before accepting.
```

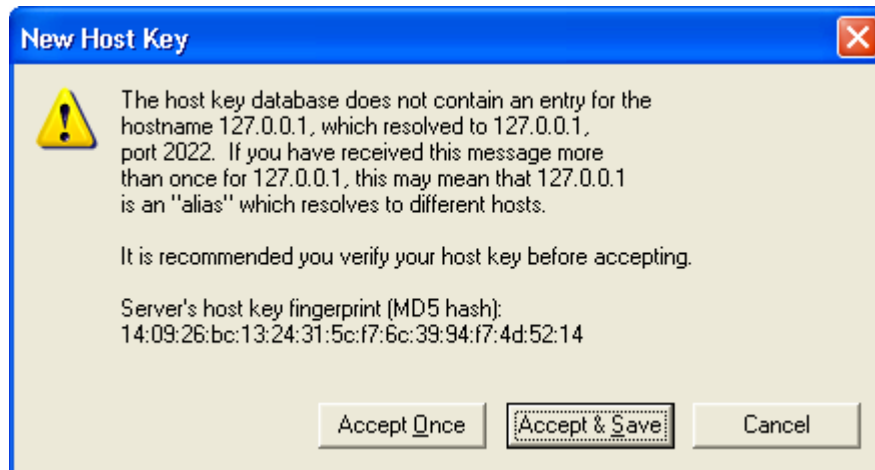
```
Server's host key fingerprint (MD5 hash):
```

```
14:09:26:bc:13:24:31:5c:f7:6c:39:94:f7:4d:52:14
```

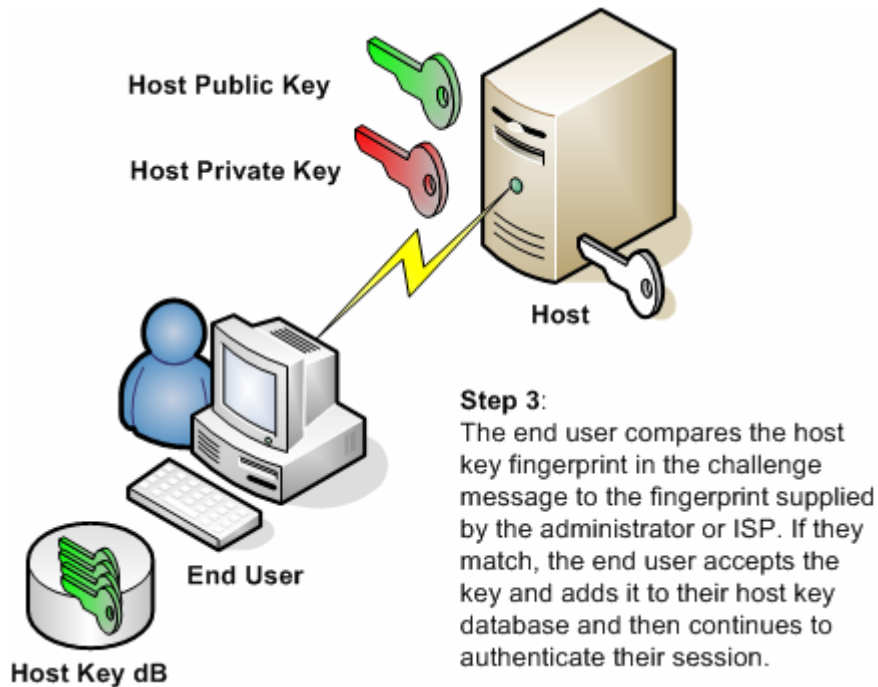
```
If you trust this host, enter "y" to add the key to the host key database
and connect. If you do not trust this host, enter "n" to abandon the
connection.
```

```
Accept and save? (y/n)
```

If you're using a Windows client, you may see a dialog such as:



Many users, not understanding exactly what the text or dialog means, simply accept the new host key. Accepting the key is problematic if the user does not know for certain that the host key it has just accepted is actually from the server that the client believes it is connecting to.



Before accepting the new host keys, the user should use a secure method to verify that the host key corresponds to the actual server. When a Secure Shell server host key is created, a unique fingerprint is also generated. This fingerprint is a human-readable cryptographic hash that can be used to verify the authenticity of the key being presented by the server to the client. Methods for verifying host keys, including using the host key's fingerprint, are discussed below.

## Handling a changed host key

After a host key has been accepted and saved, the user should not see this message again. However, if the host key presented by the server on a subsequent connection is different from the one saved on the user's local system, a second message will be displayed. Here is an example:

```
The host key sent by the server is different from the host key stored in
the host key database for myserver (192.168.0.1), port 22. This may mean
that a hostile party has "hijacked" your connection and you are not
connected to the server you specified.
```

```
It is recommended you verify your host key before accepting.
```

```
Server's host key fingerprint (MD5 hash):
```

```
14:09:26:bc:13:24:31:5c:f7:6c:39:94:f7:4d:52:14
```

```
If you trust this host, enter "y" to add the key to the host key database
and connect. If you do not trust this host, enter "n" to abandon the
connection.
```

```
Accept and save? (y/n)
```

As you can see from the text of the message, a user seeing words such as "hijacked" might get very nervous. There are several scenarios that could cause this situation. The first is that the

server has been compromised or you are experiencing a man-in-the-middle attack. However, this is not usually the case. Here are two more likely scenarios.

1. It's possible that the administrator changed the host key.
2. The machine the user is connecting to actually has more than one Secure Shell server running and the client is not keeping track of the different host keys for the different servers running on the same machine.

As with a new host key, before accepting the changed host key, the user should use a secure method to verify the host key being presented corresponds to the actual server. Here are a few methods to address this question of host authenticity.

## Known hosts

After the client connects to the server for the first time and accepts and saves the host key, it is stored in a local database. One solution for eliminating the need to manually verify the host key is for the administrator to pre-populate the database of known hosts on each of the client machines.

Where host keys are stored and the exact format they are stored in is usually client specific. For example, in SecureCRT or SecureFX, host keys are stored in the user's application data area and can be viewed, imported, or deleted using the Global Options dialog. For VanDyke's Linux and UNIX servers, they are stored in the user's home directory under `~/.vshell/known_hosts`.

In addition, most clients also look in a common location. For example, under Linux or UNIX, a common set of known host keys can be found in `/usr/local/etc/known_hosts`.

## Verifying host keys

Calling the system administrator and verifying the host key over the phone is a simple solution to making sure the host key is correct and that the client is not vulnerable to a man-in-the-middle attack. However, in many situations this is not a practical solution. There may be too many servers. There may be too many clients. Or, the administrator may not be available when the user first connects.

There are a number of other methods that can be used to distribute host keys or fingerprints:

- An ISP or network administrator might distribute host key fingerprints on a secure web page that all customers or users have access to.
- The host key fingerprint can be sent by e-mail to end users so they have it readily available to compare to the fingerprint displayed in the challenge message.
- For enterprises that already use a system such as SMS to push files out to client systems, host keys could also be distributed through this system.
- Organizations using Kerberos could take advantage of Secure Shell's GSSAPI key exchange which doesn't require hosts keys and instead leverages Kerberos host verification.

Recently, an IETF draft has been released that specifies a method of checking host key fingerprints using secure DNS (DNSSEC). Secure Shell solutions implementing this new mechanism are not yet widely available.

## **Backup your host keys**

As with any critical data, backing up your host keys is a good idea. If you should ever need to rebuild your Secure Shell server or migrate it to a new machine, these backups will be invaluable in effecting a smooth and transparent transition for your users.

## **The need for policy**

With any security solution, there is a need for policy. As part of a company's security awareness training, users should be educated about the value of host keys and made aware of a company's procedure for checking host keys. Clearly communicating this policy can alleviate a lot of the fear, uncertainty, and doubt that accompany users' decision-making when a new host key is presented.

Administrators should understand that changing host keys could have a ripple effect. In the case where a host key must be changed, the change should, if at all possible, be communicated to users of the service in advance of making it. By doing so, much of the help desk grief can be eliminated.

## **Summary**

Turning off Telnet and FTP and moving to Secure Shell makes sense and protects both passwords and data from being sent in the clear. In addition to encryption, Secure Shell provides a way to authenticate both the end user and host. This document has illustrated how host keys play an integral role in establishing this trusted connection. In order to maintain the security and integrity of the system, users and administrators need to understand host keys and the correct method for verifying them.