X.509v3 Certificates for SSH Authentication

The X.509v3 Certificates for SSH Authentication feature uses public key algorithm (PKI) for server and user authentication, and allows the Secure Shell (SSH) protocol to verify the identity of the owner of a key pair via digital certificates, signed and issued by a Certificate Authority (CA).

This module describes how to configure server and user certificate profiles for a digital certificate.

- Finding Feature Information, on page 1
- Prerequisites for X.509v3 Certificates for SSH Authentication, on page 1
- Restrictions for X.509v3 Certificates for SSH Authentication, on page 2
- Information About X.509v3 Certificates for SSH Authentication, on page 2
- How to Configure X.509v3 Certificates for SSH Authentication, on page 3
- Verifying the Server and User Authentication Using Digital Certificates, on page 6
- Configuration Examples for X.509v3 Certificates for SSH Authentication, on page 11
- Additional References for X.509v3 Certificates for SSH Authentication, on page 11
- Feature Information for X.509v3 Certificates for SSH Authentication, on page 12

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for X.509v3 Certificates for SSH Authentication

The X.509v3 Certificates for SSH Authentication feature replaces the \texttt{ip ssh server authenticate user} command with the \texttt{ip ssh server algorithm authentication} command. Configure the \texttt{default ip ssh server authenticate user} command to remove the \texttt{ip ssh server authenticate user} command from the configuration. The IOS secure shell (SSH) server will start using the \texttt{ip ssh server algorithm authentication} command.

When you configure the \texttt{ip ssh server authenticate user} command, the following message is displayed:
Warning

SSH command accepted; but this CLI will be deprecated soon. Please move to new CLI `ip ssh server algorithm authentication`. Please configure the “default ip ssh server authenticate user” to make the CLI ineffective.

Restrictions for X.509v3 Certificates for SSH Authentication

- The X.509v3 Certificates for SSH Authentication feature implementation is applicable only on the Cisco IOS Secure Shell (SSH) server side.
- The Cisco IOS SSH server supports only the x509v3-ssh-rsa algorithm-based certificate for server and user authentication.

Information About X.509v3 Certificates for SSH Authentication

X.509v3 Certificates for SSH Authentication Overview

The Secure Shell (SSH) protocol provides a secure remote access connection to network devices. The communication between the client and server is encrypted.

There are two SSH protocols that use public key cryptography for authentication. The Transport Layer Protocol, uses a digital signature algorithm (called the public key algorithm) to authenticate the server to the client. And the User Authentication Protocol uses a digital signature to authenticate (public key authentication) the client to the server.

The validity of the authentication depends upon the strength of the linkage between the public signing key and the identity of the signer. Digital certificates, such as those in X.509 Version 3 (X.509v3), are used to provide identity management. X.509v3 uses a chain of signatures by a trusted root certification authority and intermediate certificate authorities to bind a public signing key to a specific digital identity. This implementation allows the use of a public key algorithm for server and user authentication, and allows SSH to verify the identity of the owner of a key pair via digital certificates, signed and issued by a Certificate Authority (CA).

Server and User Authentication Using X.509v3

For server authentication, the Secure shell (SSH) server sends its own certificate to the SSH client for verification. This server certificate is associated with the trustpoint configured in the server certificate profile (ssh-server-cert-profile-server configuration mode).

For user authentication, the SSH client sends the user's certificate to the IOS SSH server for verification. The SSH server validates the incoming user certificate using public key infrastructure (PKI) trustpoints configured in the server certificate profile (ssh-server-cert-profile-user configuration mode).

By default, certificate-based authentication is enabled for server and user at the IOS SSH server end.
OCSP Response Stapling

The Online Certificate Status Protocol (OCSP) enables applications to determine the (revocation) state of an identified certificate. This protocol specifies the data that needs to be exchanged between an application checking the status of a certificate and the server providing that status. An OCSP client issues a status request to an OCSP responder and suspends acceptance of the certificate until a response is received. An OCSP response at a minimum consists of a responseStatus field that indicates the processing status of the a request.

For the public key algorithms, the key format consists of a sequence of one or more X.509v3 certificates followed by a sequence of zero or more OCSP responses.

The X.509v3 Certificate for SSH Authentication feature uses OCSP Response Stapling. By using OCSP response stapling, a device obtains the revocation information of its own certificate by contacting the OCSP server and then stapling the resultalong with its certificates and sending the information to the peer rather than having the peer contact the OCSP responder.

How to Configure X.509v3 Certificates for SSH Authentication

Configuring Digital Certificates for Server Authentication

SUMMARY STEPS

1. enable
2. configure terminal
3. ip ssh server algorithm hostkey \{x509v3-ssh-rsa [ssh-rsa] | ssh-rsa [x509v3-ssh-rsa]\}
4. ip ssh server certificate profile
5. server
6. trustpoint sign PKI-trustpoint-name
7. ocs response include
8. end
9. line vty line_number [ending_line_number]
10. transport input ssh

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Switch&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Switch# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>3</td>
<td>ip ssh server algorithm hostkey {x509v3-ssh-rsa [ssh-rsa]</td>
</tr>
<tr>
<td></td>
<td>x509v3-ssh-rsa }</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ip ssh server certificate profile</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Switch(config)# ip ssh server certificate profile</td>
</tr>
<tr>
<td>5</td>
<td>server</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Switch(ssh-server-cert-profile)# server</td>
</tr>
<tr>
<td>6</td>
<td>trustpoint sign PKI-trustpoint-name</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Switch(ssh-server-cert-profile-server)# trustpoint sign trust1</td>
</tr>
<tr>
<td>7</td>
<td>ocsp-response include</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Switch(ssh-server-cert-profile-server)# ocsp-response include</td>
</tr>
<tr>
<td>8</td>
<td>end</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Switch(ssh-server-cert-profile-server)# end</td>
</tr>
<tr>
<td>9</td>
<td>line vty line_number [ending_line_number]</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Switch(config)# line vty line_number [ending_line_number]</td>
</tr>
<tr>
<td>10</td>
<td>transport input ssh</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>Switch(config-line)#transport input ssh</td>
</tr>
</tbody>
</table>
## Configuring Digital Certificates for User Authentication

### SUMMARY STEPS

1. enable
2. configure terminal
3. `ip ssh server algorithm authentication {publickey | keyboard | password}`
4. `ip ssh server algorithm publickey {x509v3-ssh-rsa [ssh-rsa] | ssh-rsa [x509v3-ssh-rsa]}`
5. `ip ssh server certificate profile`
6. `user`
7. `trustpoint verify PKI-trustpoint-name`
8. `ocsp-response required`
9. `end`
10. `line vty line_number [ending_line_number]`
11. `transport input ssh`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Switch&gt; enable</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Switch# configure terminal</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> `ip ssh server algorithm authentication {publickey</td>
<td>keyboard</td>
<td>password}`</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td>• To use the certificate method for user authentication, the <strong>publickey</strong> keyword must be configured.</td>
</tr>
<tr>
<td>Switch(config)# <code>ip ssh server algorithm authentication publickey</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> `ip ssh server algorithm publickey {x509v3-ssh-rsa [ssh-rsa]</td>
<td>ssh-rsa [x509v3-ssh-rsa]}`</td>
<td>Defines the order of public key algorithms. Only the configured algorithm is accepted by the SSH client for user authentication.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td>• <code>x509v3-ssh-rsa</code>—Certificate-based authentication</td>
</tr>
<tr>
<td>Switch(config)# <code>ip ssh server algorithm publickey x509v3-ssh-rsa</code></td>
<td></td>
<td>• <code>ssh-rsa</code>—Public-key-based authentication</td>
</tr>
</tbody>
</table>
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 5** ip ssh server certificate profile  
Example: Switch(config)# ip ssh server certificate profile | Configures server certificate profile and user certificate profile and enters SSH certificate profile configuration mode. |
| **Step 6** user  
Example: Switch(ssh-server-cert-profile)# user | Configures user certificate profile and enters SSH server certificate profile user configuration mode. |
| **Step 7** trustpoint verify PKI-trustpoint-name  
Example: Switch(ssh-server-cert-profile-user)# trustpoint verify trust2 | Configures the public key infrastructure (PKI) trustpoint that is used to verify the incoming user certificate.  
**Note** Configure multiple trustpoints by executing the same command multiple times. A maximum of 10 trustpoints can be configured. |
| **Step 8** ocsp-response required  
Example: Switch(ssh-server-cert-profile-user)# ocsp-response required | (Optional) Mandates the presence of the Online Certificate Status Protocol (OCSP) response with the incoming user certificate.  
**Note** By default, the user certificate is accepted without an OCSP response. |
| **Step 9** end  
Example: Switch(ssh-server-cert-profile-user)# end | Exits SSH server certificate profile user configuration mode and returns to privileged EXEC mode. |
| **Step 10** line vty line_number [ending_line_number]  
Example: Switch(config)# line vty line_number [ending_line_number] | Enters line configuration mode to configure the virtual terminal line settings. For line_number and ending_line_number, specify a pair of lines. The range is 0 to 15. |
| **Step 11** transport input ssh  
Example: Switch(config-line)#transport input ssh | Specifies that the Switch prevent non-SSH Telnet connections. This limits the router to only SSH connections. |

### Verifying the Server and User Authentication Using Digital Certificates

**SUMMARY STEPS**

1. enable
2. show ip ssh
3. debug ip ssh detail
4. show log
5. debug ip packet
6. show log

DETAILED STEPS

Step 1  enable
Enables privileged EXEC mode.
  • Enter your password if prompted.

Example:
Device> enable

Step 2  show ip ssh
Displays the currently configured authentication methods. To confirm the use of certificate-based authentication, ensure that the x509v3-ssh-rsa algorithm is the configured host key algorithm.

Example:
Device# show ip ssh
SSH Enabled - version 1.99
Authentication methods:publickey,keyboard-interactive,password
Authentication Publickey Algorithms:x509v3-ssh-rsa,ssh-rsa
Hostkey Algorithms:x509v3-ssh-rsa,ssh-rsa
Authentication timeout: 120 secs; Authentication retries: 3
Minimum expected Diffie Hellman key size : 1024 bits

Step 3  debug ip ssh detail
Turns on debugging messages for SSH details.

Example:
Device# debug ip ssh detail
ssh detail messages debugging is on

Step 4  show log
Shows the debug message log.

Example:
Device# show log
Syslog logging: enabled (0 messages dropped, 9 messages rate-limited, 0 flushes, 0 overruns, xml disabled, filtering disabled)
No Active Message Discriminator.
No Inactive Message Discriminator.

Console logging: disabled
Monitor logging: level debugging, 0 messages logged, xml disabled, filtering disabled
Buffer logging: level debugging, 233 messages logged, xml disabled, filtering disabled
Exception Logging: size (4096 bytes)
Count and timestamp logging messages: disabled
File logging: disabled
Persistent logging: disabled

No active filter modules.

Trap logging: level informational, 174 message lines logged
Logging Source-Interface: VRF Name:
Log Buffer (4096 bytes):
5 IST: SSH2 CLIENT 0: SSH2_MSG_KEXINIT sent
*Sep 6 14:44:08.496 IST: SSH0: protocol version id is = SSH-1.99-Cisco-1.25
*Sep 6 14:44:08.496 IST: SSH2 0: kexinit sent: kex algo = diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1
*Sep 6 14:44:08.496 IST: SSH2 0: Server certificate trustpoint not found. Skipping hostkey algo = x509v3=ssh-rsa
*Sep 6 14:44:08.496 IST: SSH2 0: kexinit sent: hostkey algo = ssh-rsa
*Sep 6 14:44:08.496 IST: SSH2 0: kexinit sent: encryption algo = aes128-ctr,aes192-ctr,aes256-ctr
*Sep 6 14:44:08.496 IST: SSH2 0: kexinit sent: mac algo = hmac-sha2-256,hmac-sha2-512,hmac-sha1,hmac-sha1-96
*Sep 6 14:44:08.496 IST: SSH2 0: kexinit sent: hostkey algo = ssh-rsa
*Sep 6 14:44:08.496 IST: SSH2 0: kexinit sent: mac algo = hmac-sha2-256,hmac-sha2-512,hmac-sha1,hmac-sha1-96
*Sep 6 14:44:08.496 IST: SSH2 0: kexinit sent: hostkey algo = ssh-rsa
*Sep 6 14:44:08.496 IST: SSH2 0: kexinit sent: mac algo = hmac-sha2-256,hmac-sha2-512,hmac-sha1,hmac-sha1-96
*Sep 6 14:44:08.497 IST: SSH2 CLIENT 0: SSH2_MSG_KEXINIT sent
*Sep 6 14:44:08.497 IST: SSH2 0: SSH2_MSG_KEXINIT received
*Sep 6 14:44:08.497 IST: SSH2 0: kex: client->server enc:aes128-ctr mac:hmac-sha2-256
*Sep 6 14:44:08.497 IST: SSH2 0: kex: server->client enc:aes128-ctr mac:hmac-sha2-256
*Sep 6 14:44:08.497 IST: SSH2 0: Using hostkey algo = ssh-rsa
*Sep 6 14:44:08.497 IST: SSH2 0: Using kex algo = diffie-hellman-group-exchange-sha1
*Sep 6 14:44:08.497 IST: SSH2 CLIENT 0: SSH2_MSG_KEXINIT received
*Sep 6 14:44:08.497 IST: SSH2 CLIENT 0: kex: server->client enc:aes128-ctr mac:hmac-sha2-256
*Sep 6 14:44:08.497 IST: SSH2 CLIENT 0: kex: client->server enc:aes128-ctr mac:hmac-sha2-256
*Sep 6 14:44:08.497 IST: SSH2 CLIENT 0: Using hostkey algo = ssh-rsa
*Sep 6 14:44:08.497 IST: SSH2 CLIENT 0: Using kex algo = diffie-hellman-group-exchange-sha1
*Sep 6 14:44:08.497 IST: SSH2 CLIENT 0: SSH2_MSG_KEX_DH_GEX_REQUEST sent
*Sep 6 14:44:08.497 IST: SSH2 CLIENT 0: Range sent- 2048 < 2048 < 4096
*Sep 6 14:44:08.497 IST: SSH2 CLIENT 0: SSH2_MSG_KEX_DH_GEX_REPLY received
*Sep 6 14:44:08.497 IST: SSH2 CLIENT 0: Range sent by client is - 2048 < 2048 < 4096
*Sep 6 14:44:08.510 IST: SSH2 0: Modulus size established : 2048 bits
*Sep 6 14:44:08.510 IST: SSH2 0: expecting SSH2_MSG_KEX_DH_GEX_INIT
*Sep 6 14:44:08.510 IST: SSH2 0: expecting SSH2_MSG_KEX_DH_GEX_GROUP received
*Sep 6 14:44:08.510 IST: SSH2 CLIENT 0: Server has chosen 2048 -bit dh keys
*Sep 6 14:44:08.523 IST: SSH2 CLIENT 0: expecting SSH2_MSG_KEX_DH_GEX_REPLY
*Sep 6 14:44:08.524 IST: SSH2 0: SSH2_MSG_KEXDH_INIT received
*Sep 6 14:44:08.555 IST: SSH2: kex_derive_keys complete
*Sep 6 14:44:08.555 IST: SSH2 0: SSH2_MSG_NEWKEYS sent
*Sep 6 14:44:08.555 IST: SSH2 0: waiting for SSH2_MSG_NEWKEYS
*Sep 6 14:44:08.555 IST: SSH2 CLIENT 0: SSH2_MSG_KEX_DH_GEX_REPLY received
*Sep 6 14:44:08.555 IST: SSH2 CLIENT 0: Skipping ServerHostKey Validation
*Sep 6 14:44:08.571 IST: SSH2 CLIENT 0: signature length 271
*Sep 6 14:44:08.571 IST: SSH2: kex_derive_keys complete
*Sep 6 14:44:08.571 IST: SSH2 CLIENT 0: SSH2_MSG_NEWKEYS sent
*Sep 6 14:44:08.571 IST: SSH2 0: waiting for SSH2_MSG_NEWKEYS
*Sep 6 14:44:08.571 IST: SSH2 0: SSH2_MSG_NEWKEYS received
*Sep 6 14:44:08.571 IST: SSH2 0: SSH2_MSG_NEWKEYS received
*Sep 6 14:44:08.571 IST: SSH2 0: Authentication that can continue = publickey,keyboard-interactive,password
*Sep 6 14:44:08.572 IST: SSH2 0: Using method = none

Verifying the Server and User Authentication Using Digital Certificates

X.509v3 Certificates for SSH Authentication
*Sep 6 14:44:08.572 IST: SSH2 0: Authentications that can continue =
  publickey,keyboard-interactive,password
*Sep 6 14:44:08.572 IST: SSH2 0: Using method = keyboard-interactive
*Sep 6 14:44:11.983 IST: SSH2 0: authentication successful for cisco
*Sep 6 14:44:11.983 IST: %SEC_LOGIN-5-LOGIN_SUCCESS: Login Success [user: cisco] [Source:
  192.168.121.40] [localport: 22] at 14:44:11 IST Thu Sep 6 2018
*Sep 6 14:44:11.984 IST: SSH2 0: channel open request
*Sep 6 14:44:11.985 IST: SSH2 0: pty-req request
*Sep 6 14:44:11.985 IST: SSH2 0: setting TTY - requested: height 24, width 80; set: height 24, width
  80
*Sep 6 14:44:11.985 IST: SSH2 0: shell request
*Sep 6 14:44:11.985 IST: SSH2 0: shell message received
*Sep 6 14:44:11.985 IST: SSH2 0: starting shell for vty
*Sep 6 14:44:22.066 IST: %SYS-6-LOGOUT: User cisco has exited tty session 1(192.168.121.40)
*Sep 6 14:44:22.166 IST: SSH0: Session terminated normally
*Sep 6 14:44:22.167 IST: SSH CLIENT0: Session terminated normally

Step 5  debug ip packet

Turns on debugging for IP packet details.

Example:
Device# debug ip packet

Step 6  show log

Shows the debug message log.

Example:
Device# show log

yslog logging: enabled (0 messages dropped, 9 messages rate-limited, 0 flushes, 0 overruns, xml
disabled, filtering disabled)

No Active Message Discriminator.

No Inactive Message Discriminator.

  Console logging: disabled
  Monitor logging: level debugging, 0 messages logged, xml disabled,
  filtering disabled
  Buffer logging: level debugging, 1363 messages logged, xml disabled,
  filtering disabled
  Exception Logging: size (4096 bytes)
  Count and timestamp logging messages: disabled
  File logging: disabled
  Persistent logging: disabled

No active filter modules.

  Trap logging: level informational, 176 message lines logged
  Logging Source-Interface:  VRF Name:

  Log Buffer (4096 bytes):
  bleid=0, s=192.168.121.40 (local), d=192.168.121.40 (FortyGigabitEthernet1/0/1), routed via RIB
  *Sep 6 14:45:45.177 IST: IP: s=192.168.121.40 (local), d=192.168.121.40 (FortyGigabitEthernet1/0/1),
  len 40, sending
  *Sep 6 14:45:45.177 IST: IP: s=192.168.121.40 (local), d=192.168.121.40 (FortyGigabitEthernet1/0/1),
  len 40, output feature, NAT Inside(8), rtype 1, forus FALSE, sendself FALSE, sendself FALSE, mtu 0, fwdchk FALSE

X.509v3 Certificates for SSH Authentication
Verifying the Server and User Authentication Using Digital Certificates

X.509v3 Certificates for SSH Authentication

*Sep 6 14:45:45.177 IST: IP: tableid=0, s=192.168.121.40 (FortyGigabitEthernet1/0/1), d=192.168.121.40 (FortyGigabitEthernet1/0/1), routed via RIB
*Sep 6 14:45:45.177 IST: IP: tableid=0, s=192.168.121.40 (FortyGigabitEthernet1/0/1), d=192.168.121.40 (FortyGigabitEthernet1/0/1), routed via RIB
*Sep 6 14:45:45.177 IST: IP: s=192.168.121.40 (FortyGigabitEthernet1/0/1), d=192.168.121.40 (local), d=192.168.121.40, len 40, local feature, feature skipped, NAT(2), rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Sep 6 14:45:45.178 IST: IP: tableid=0, s=192.168.121.40 (FortyGigabitEthernet1/0/1), d=192.168.121.40 (local), d=192.168.121.40
(FortyGigabitEthernet1/0/1), routed via RIB
*Sep 6 14:45:45.178 IST: IP: s=192.168.121.40 (local), d=192.168.121.40 (FortyGigabitEthernet1/0/1), len 40, sending
*Sep 6 14:45:45.178 IST: IP: s=192.168.121.40 (local), d=192.168.121.40 (FortyGigabitEthernet1/0/1), len 40, output feature, NAT Inside(8), rtype 1, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Sep 6 14:45:45.178 IST: IP: tableid=0, s=192.168.121.40 (FortyGigabitEthernet1/0/1), d=192.168.121.40 (FortyGigabitEthernet1/0/1), routed via RIB
*Sep 6 14:45:45.178 IST: IP: s=192.168.121.40 (local), d=192.168.121.40, len 40, local feature, feature skipped, NAT(2), rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Sep 6 14:45:45.178 IST: IP: tableid=0, s=192.168.121.40 (local), d=192.168.121.40 (local), d=192.168.121.40
(FortyGigabitEthernet1/0/1), routed via RIB
*Sep 6 14:45:45.178 IST: IP: s=192.168.121.40 (local), d=192.168.121.40 (FortyGigabitEthernet1/0/1), len 40, sending
*Sep 6 14:45:45.178 IST: IP: s=192.168.121.40 (local), d=192.168.121.40 (FortyGigabitEthernet1/0/1), len 40, output feature, NAT Inside(8), rtype 1, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Sep 6 14:45:45.178 IST: IP: tableid=0, s=192.168.121.40 (FortyGigabitEthernet1/0/1), d=192.168.121.40 (FortyGigabitEthernet1/0/1), routed via RIB
*Sep 6 14:45:45.178 IST: IP: s=192.168.121.40 (local), d=192.168.121.40, len 40, local feature, feature skipped, NAT(2), rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Sep 6 14:45:45.178 IST: IP: tableid=0, s=192.168.121.40 (FortyGigabitEthernet1/0/1), d=192.168.121.40 (FortyGigabitEthernet1/0/1), routed via RIB
*Sep 6 14:45:45.178 IST: IP: s=192.168.121.40 (local), d=192.168.121.40 (FortyGigabitEthernet1/0/1), len 40, sending
*Sep 6 14:45:45.178 IST: IP: s=192.168.121.40 (local), d=192.168.121.40 (FortyGigabitEthernet1/0/1), len 40, output feature, NAT Inside(8), rtype 1, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Sep 6 14:45:45.178 IST: IP: tableid=0, s=192.168.121.40 (FortyGigabitEthernet1/0/1), d=192.168.121.40 (FortyGigabitEthernet1/0/1), routed via RIB
*Sep 6 14:45:45.178 IST: IP: s=192.168.121.40 (local), d=192.168.121.40, len 40, local feature, feature skipped, NAT(2), rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Sep 6 14:45:45.178 IST: IP: tableid=0, s=192.168.121.40 (local), d=192.168.121.40 (local), d=192.168.121.40
(FortyGigabitEthernet1/0/1), routed via RIB
*Sep 6 14:45:45.179 IST: IP: tableid=0, s=192.168.121.40 (FortyGigabitEthernet1/0/1), d=192.168.121.40 (FortyGigabitEthernet1/0/1), routed via RIB
*Sep 6 14:45:45.179 IST: IP: s=192.168.121.40 (local), d=192.168.121.40 (FortyGigabitEthernet1/0/1), len 40, sending
*Sep 6 14:45:45.179 IST: IP: s=192.168.121.40 (local), d=192.168.121.40 (FortyGigabitEthernet1/0/1), len 40, output feature, NAT Inside(8), rtype 1, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Sep 6 14:45:45.179 IST: IP: tableid=0, s=192.168.121.40 (FortyGigabitEthernet1/0/1), d=192.168.121.40 (FortyGigabitEthernet1/0/1), routed via RIB
*Sep 6 14:45:45.179 IST: IP: s=192.168.121.40 (local), d=192.168.121.40, len 40, local feature, feature skipped, NAT(2), rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Sep 6 14:45:45.179 IST: IP: tableid=0, s=192.168.121.40 (local), d=192.168.121.40 (local), d=192.168.121.40
(FortyGigabitEthernet1/0/1), routed via RIB
*Sep 6 14:45:45.179 IST: IP: tableid=0, s=192.168.121.40 (FortyGigabitEthernet1/0/1), d=192.168.121.40 (FortyGigabitEthernet1/0/1), routed via RIB
*Sep 6 14:45:45.179 IST: IP: s=192.168.121.40 (local), d=192.168.121.40 (FortyGigabitEthernet1/0/1), len 40, sending
*Sep 6 14:45:45.179 IST: IP: s=192.168.121.40 (local), d=192.168.121.40 (FortyGigabitEthernet1/0/1), len 40, output feature, NAT Inside(8), rtype 1, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Sep 6 14:45:45.179 IST: IP: tableid=0, s=192.168.121.40 (FortyGigabitEthernet1/0/1), d=192.168.121.40 (FortyGigabitEthernet1/0/1), routed via RIB
*Sep 6 14:45:45.179 IST: IP: s=192.168.121.40 (local), d=192.168.121.40, len 40, local feature, feature skipped, NAT(2), rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Sep 6 14:45:45.179 IST: IP: tableid=0, s=192.168.121.40 (local), d=192.168.121.40 (local), d=192.168.121.40
(FortyGigabitEthernet1/0/1), routed via RIB

X.509v3 Certificates for SSH Authentication
Configuration Examples for X.509v3 Certificates for SSH Authentication

Example: Configuring Digital Certificates for Server Authentication

Switch> enable
Switch# configure terminal
Switch(config)# ip ssh server algorithm hostkey x509v3-ssh-rsa
Switch(config)# ip ssh server certificate profile
Switch(ssh-server-cert-profile)# server
Switch(ssh-server-cert-profile-server)# trustpoint sign trust1
Switch(ssh-server-cert-profile-server)# exit

Example: Configuring Digital Certificate for User Authentication

Switch> enable
Switch# configure terminal
Switch(config)# ip ssh server algorithm authentication publickey
Switch(config)# ip ssh server algorithm publickey x509v3-ssh-rsa
Switch(config)# ip ssh server certificate profile
Switch(config)# trustpoint verify trust2
Switch(config)# end

Additional References for X.509v3 Certificates for SSH Authentication

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKI configuration</td>
<td>Configuring and Managing a Cisco IOS Certificate Server for PKI Deployment</td>
</tr>
</tbody>
</table>
Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Cisco Support website provides extensive online resources, including</td>
<td><a href="http://www.cisco.com/support">http://www.cisco.com/support</a></td>
</tr>
<tr>
<td>documentation and tools for troubleshooting and resolving technical issues</td>
<td></td>
</tr>
<tr>
<td>with Cisco products and technologies.</td>
<td></td>
</tr>
<tr>
<td>To receive security and technical information about your products, you can</td>
<td></td>
</tr>
<tr>
<td>subscribe to various services, such as the Product Alert Tool (accessed from</td>
<td></td>
</tr>
<tr>
<td>Field Notices), the Cisco Technical Services Newsletter, and Really Simple</td>
<td></td>
</tr>
<tr>
<td>Syndication (RSS) Feeds.</td>
<td></td>
</tr>
<tr>
<td>Access to most tools on the Cisco Support website requires a Cisco.com user</td>
<td></td>
</tr>
<tr>
<td>ID and password.</td>
<td></td>
</tr>
</tbody>
</table>

Feature Information for X.509v3 Certificates for SSH Authentication

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
Table 1: Feature Information for X.509v3 Certificates for SSH Authentication

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>X.509v3 Certificates for SSH Authentication</td>
<td>Cisco IOS 15.2(4)E1</td>
<td>The X.509v3 Certificates for SSH Authentication feature uses the X5.09v3 digital certificates in server and user authentication at the SSH server side.\n\nThe following commands were introduced or modified: <code>ip ssh server algorithm hostkey</code>, <code>ip ssh server algorithm authentication</code>, and <code>ip ssh server certificate profile</code>.\n\nThis feature was implemented on the following platforms:\n\n- Catalyst 2960C, 2960CX, 2960P, 2960X, and 2960XR Series Switches\n- Catalyst 3560CX and 3560X Series Switches\n- Catalyst 3750X Series Switches\n- Catalyst 4500E Sup7-E, Sup7L-E, Sup8-E, and 4500X Series Switches\n- Catalyst 4900M, 4900F-E Series Switches</td>
</tr>
</tbody>
</table>