SSH 認証の X.509v3 証明書

SSH 認証用の X.509v3 証明書機能は、公開キー アルゴリズム（PKI）を使用してサーバおよびユーザの認証を行い、認証局（CA）が署名し発行したデジタル証明書を介してキー ペアの所有者のアイデンティティをセキュア シェル（SSH）プロトコルによって検証することを可能します。

このモジュールでは、デジタル証明書用のサーバおよびユーザ証明書プロファイルを設定する方法について説明します。

SSH 認証の X.509v3 証明書の前提条件

SSH 認証用の X.509v3 証明書機能では、 **ip ssh server algorithm authentication** コマンドの代わりに **ip ssh server authenticate user** コマンドが置き換えられます。**default ip ssh server authenticate user** コマンドを設定し、コンフィギュレーションから **ip ssh server authenticate user** コマンドを削除します。IOS セキュアシェル（SSH）サーバは **ip ssh server algorithm authentication** コマンドを使用して起動します。

ip ssh server authenticate user コマンドを実行すると、次のメッセージが表示されます。

警告

SSH コマンドを受け入れました。ただし、この CLI はまもなく廃止されます。新しい CLI **ip ssh server algorithm authentication**に移動してください。**default ip ssh server authenticate user** を設定し、CLI を無効にします。

SSH 認証の X.509v3 証明書の制約事項

- SSH 認証用の X.509v3 証明書機能の実装は、Cisco IOS セキュア シェル（SSH）サーバ側にのみ適用できます。
- Cisco IOS SSH サーバは、サーバおよびユーザ認証について、x509v3-ssh-rsa アルゴリズム ベースの証明書のみをサポートします。

SSH 認証用の X.509v3 証明書に関する情報

SSH 認証用の X.509v3 証明書の概要

セキュア シェル（SSH）プロトコルは、ネットワーク デバイスへの安全なリモート アクセス接続を提供します。クライアントとサーバの間の通信は暗号化されます。

公開キー暗号化を使用して認証を行う SSH プロトコルが 2 つあります。トランスポート層プロトコルは、デジタル署名アルゴリズム（公開キーアルゴリズムと呼ばれます）を使用して、サーバをクライアントに対して認証します。一方、ユーザ認証プロトコルは、デジタル署名を使用して、クライアントをサーバに対して認証します（公開キー認証）。

認証の有効性は、公開署名キーとその署名者のアイデンティティとの関連の強さに依存します。X.509 バージョン 3（X.509v3）などのデジタル証明書は、アイデンティティ管理のために使用されます。X.509v3 は、信頼できるルート認証局とその中間認証局による署名の連鎖を使用して、公開署名キーを特定のデジタル アイデンティティにバインドします。この実装により、公開キー アルゴリズムを使用したサーバとユーザの認証が可能になるとともに、認証局（CA）が署名し発行したデジタル証明書を介してキー ペアの所有者のアイデンティティを SSH で検証することが可能になります。

X.509v3 を使用したサーバおよびユーザ認証

サーバ認証の場合、セキュア シェル（SSH）サーバが確認のためにそれ自体の証明書を SSH クライアントに送信します。このサーバ証明書は、サーバ証明書プロファイル（ssh-server-cert-profile-server コンフィギュレーション モード）で設定されたトラストポイントに関連付けられます。

ユーザ認証の場合、SSH クライアントが確認のためにユーザの証明書を IOS SSH サーバに送信します。SSH サーバは、サーバ証明書プロファイル（ssh-server-cert-profile-user コンフィギュレーション モード）で設定された公開キー インフラストラクチャ（PKI）トラストポイントを使用して、受信したユーザ証明書を確認します。

デフォルトでは、証明書ベースの認証が、IOS SSH サーバ端末でサーバおよびユーザに対して有効になっています。

OCSP 応答ステープリング

オンライン証明書ステータス プロトコル（OCSP）では、識別された証明書の（失効）状態をアプリケーションが判断することが可能です。このプロトコルは、証明書のステータスをチェックするアプリケーションとそのステータスを提供するサーバとの間でやり取りする必要があるデータを指定します。OCSP クライアントは OCSP レスポンダにステータス要求を発行し、応答を受信するまで証明書の受け入れを保留します。OCSP 応答には、少なくとも、要求の処理ステータスを示す responseStatus フィールドが含まれます。

公開キー アルゴリズムの場合、キーの形式は、1 つ以上の X.509v3 証明書のシーケンスと、その後に続く 0 個以上の OCSP 応答のシーケンスから成ります。

SSH 認証機能向けの X.509v3 証明書は、OCSP 応答ステープリングを使用します。OCSP 応答ステープリングを使用することにより、デバイスは、OCSP サーバにアクセスしてから結果を証明書とともにステープリングして、ピアから OCSP レスポンダにアクセスさせるのではなくピアに情報を送ることで、自身の証明書の失効情報を取得します。

SSH 認証用の X.509v3 証明書の設定方法

サーバ認証用のデジタル証明書の設定

手順

	コマンドまたはアクション	目的
ステップ 1	**enable** 例： `Device> enable`	特権 EXEC モードを有効にします。 • パスワードを入力します（要求された場合）。
ステップ 2	**configure terminal** 例： `Device# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 3	**ip ssh server algorithm hostkey** {**x509v3-ssh-rsa** [**ssh-rsa**] | **ssh-rsa** [**x509v3-ssh-rsa**]} 例： `Device(config)# ip ssh server algorithm hostkey x509v3-ssh-rsa`	ホスト キー アルゴリズムの順序を定義します。セキュア シェル（SSH）クライアントとネゴシエートされるのは、設定済みアルゴリズムのみです。

	コマンドまたはアクション	目的
		（注） IOS SSH サーバには、1 つ以上の設定済みホスト キー アルゴリズムが必要です。 • **x509v3-ssh-rsa**：証明書ベースの認証 • **ssh-rsa**：公開キーベースの認証
ステップ 4	**ip ssh server certificate profile** 例： `Device(config)# ip ssh server certificate profile`	サーバ証明書プロファイルおよびユーザ証明書プロファイルを設定し、SSH 証明書プロファイル コンフィギュレーション モードを開始します。
ステップ 5	**server** 例： `Device(ssh-server-cert-profile)# server`	サーバ証明書プロファイルを設定し、SSH サーバ証明書プロファイルのユーザ コンフィギュレーション モードを開始します。 • サーバ プロファイルは、サーバ認証時にサーバ証明書を SSH クライアントに送信するために使用されます。
ステップ 6	**trustpoint sign** *PKI-trustpoint-name* 例： `Device(ssh-server-cert-profile-server)# trustpoint sign trust1`	公開キー インフラストラクチャ（PKI）トラストポイントをサーバ証明書プロファイルにアタッチします。 • SSH サーバは、この PKI トラストポイントに関連付けられた証明書をサーバ認証に使用します。
ステップ 7	**ocsp-response include** 例： `Device(ssh-server-cert-profile-server)# ocsp-response include`	（任意）Online Certificate Status Protocol（OCSP）の応答または OCSP ステープリングをサーバ証明書と一緒に送信します。 （注） デフォルトでは、OCSP 応答はサーバ証明書と一緒には送信されません。

	コマンドまたはアクション	目的
ステップ 8	**end** 例： `Device(ssh-server-cert-profile-server)# end`	SSH サーバ証明書プロファイルのサーバ コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。
ステップ 9	**line vty line_number** [*ending_line_number*] 例： `Device(config)# line vty line_number [ending_line_number]`	ライン コンフィギュレーション モードを開始して、仮想端末回線設定を設定します。line_number および ending_line_number には、回線のペアを指定します。指定できる範囲は 0 ～ 15 です。
ステップ 10	**transport input ssh** 例： `Device(config-line)#transport input ssh`	非 SSH Telnet によるデバイスへの接続を許可しない設定です。これにより、ルータは SSH 接続に限定されます。

ユーザ認証用のデジタル証明書の設定

手順

	コマンドまたはアクション	目的
ステップ 1	**enable** 例： `Device> enable`	特権 EXEC モードを有効にします。 • パスワードを入力します（要求された場合）。
ステップ 2	**configure terminal** 例： `Device# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 3	**ip ssh server algorithm authentication** {**publickey** | **keyboard** | **password**} 例： `Device(config)# ip ssh server algorithm authentication publickey`	ユーザ認証アルゴリズムの順序を定義します。セキュア シェル（SSH）クライアントとネゴシエートされるのは、設定済みアルゴリズムのみです。

	コマンドまたはアクション	目的
		（注） • IOS SSH サーバには、1 つ以上の設定済みユーザ認証アルゴリズムが必要です。 • ユーザ認証に証明書方式を使用するには、**publickey** キーワードを設定する必要があります。
ステップ 4	**ip ssh server algorithm publickey** {**x509v3-ssh-rsa** [**ssh-rsa**] | **ssh-rsa** [**x509v3-ssh-rsa**]} 例： `Device(config)# ip ssh server algorithm publickey x509v3-ssh-rsa`	公開キーアルゴリズムの順序を定義します。SSH クライアントによってユーザ認証に許可されるのは、設定済みのアルゴリズムのみです。 （注） IOS SSH クライアントには、1 つ以上の設定済み公開キーアルゴリズムが必要です。 • **x509v3-ssh-rsa**：証明書ベースの認証 • **ssh-rsa**：公開キーベースの認証
ステップ 5	**ip ssh server certificate profile** 例： `Device(config)# ip ssh server certificate profile`	サーバ証明書プロファイルおよびユーザ証明書プロファイルを設定し、SSH 証明書プロファイルコンフィギュレーション モードを開始します。
ステップ 6	**user** 例： `Device(ssh-server-cert-profile)# user`	ユーザ証明書プロファイルを設定し、SSH サーバ証明書プロファイルのユーザ コンフィギュレーション モードを開始します。
ステップ 7	**trustpoint verify** *PKI-trustpoint-name* 例： `Device(ssh-server-cert-profile-user)# trustpoint verify trust2`	受信したユーザ証明書の確認に使用される公開キー インフラストラクチャ（PKI）トラストポイントを設定します。

	コマンドまたはアクション	目的
		（注） 同じコマンドを複数回実行することで、複数のトラストポイントを設定します。最大10のトラストポイントを設定できます。
ステップ 8	**ocsp-response required** 例： `Device(ssh-server-cert-profile-user)# ocsp-response required`	（任意）受信したユーザ証明書による Online Certificate Status Protocol（OCSP）の応答の有無を要求します。 （注） デフォルトでは、ユーザ証明書はOCSP応答なしで受け入れられます。
ステップ 9	**end** 例： `Device(ssh-server-cert-profile-user)# end`	SSH サーバ証明書プロファイルのユーザ コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。
ステップ 10	**line vty line_number** [*ending_line_number*] 例： `Device(config)# line vty line_number [ending_line_number]`	ライン コンフィギュレーション モードを開始して、仮想端末回線設定を設定します。line_number および ending_line_number には、回線のペアを指定します。指定できる範囲は 0 ～ 15 です。
ステップ 11	**transport input ssh** 例： `Device(config-line)#transport input ssh`	非 SSH Telnet によるデバイスへの接続を許可しない設定です。これにより、ルータは SSH 接続に限定されます。

デジタル証明書を使用したサーバおよびユーザ認証の確認

手順

ステップ 1 **enable**

特権 EXEC モードを有効にします。

- パスワードを入力します（要求された場合）。

例：

```
Device> enable
```

ステップ 2 show ip ssh

現在設定されている認証方式を表示します。証明書ベース認証の使用を確認するには、x509v3-ssh-rsa アルゴリズムが設定済みのホスト キー アルゴリズムであることを確認します。

例：

```
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
```

ステップ 3 debug ip ssh detail

SSH 詳細のデバッグメッセージをオンにします。

例：

```
Device# debug ip ssh detail

ssh detail messages debugging is on
```

ステップ 4 show log

デバッグメッセージログを表示します。

例：

```
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: SSH2_MSG_KEXINIT sent
*Sep  6 14:44:08.496 IST: SSH2 0: SSH2_MSG_KEXINIT received
*Sep  6 14:44:08.496 IST: SSH2 0: kex: client->server enc:aes128-ctr mac:hmac-sha2-256
*Sep  6 14:44:08.496 IST: SSH2 0: kex: server->client enc:aes128-ctr mac:hmac-sha2-256
*Sep  6 14:44:08.496 IST: SSH2 0: Using hostkey algo = ssh-rsa
*Sep  6 14:44:08.496 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 0: SSH2_MSG_KEX_DH_GEX_REQUEST received
*Sep  6 14:44:08.497 IST: SSH2 0: Range sent by client is - 2048 < 2048 < 4096
*Sep  6 14:44:08.497 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 CLIENT 0: 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 CLIENT 0: waiting for SSH2_MSG_NEWKEYS
*Sep  6 14:44:08.571 IST: SSH2 CLIENT 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: Authentications that can continue =
publickey,keyboard-interactive,password
*Sep  6 14:44:08.572 IST: SSH2 0: Using method = none
*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 14:44:11.984 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
```

ステップ 5 debug ip packet

IP パケット詳細のデバッグをオンにします。

例：

```
Device# debug ip packet
```

ステップ 6 show log

デバッグメッセージログを表示します。

例：

```
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, mtu 0, fwdchk FALSE
*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 (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
(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
(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: 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: 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
(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 14:45:45.179 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
(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
```

SSH 認証用の X.509v3 証明書の設定例

例：サーバ認証用のデジタル証明書の設定

```
Device> enable
Device# configure terminal
Device(config)# ip ssh server algorithm hostkey x509v3-ssh-rsa
```

```
Device(config)# ip ssh server certificate profile
Device(ssh-server-cert-profile)# server
Device(ssh-server-cert-profile-server)# trustpoint sign trust1
Device(ssh-server-cert-profile-server)# exit
```

例：ユーザ認証用のデジタル証明書の設定

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

SSH 認証用の X.509v3 証明書に関するその他の参考資料

関連資料

関連項目	マニュアル タイトル
この章で使用するコマンドの完全な構文および使用方法の詳細。	*Cisco IOS* リリース *15.2(7)E*（*Catalyst* マイクロスイッチ）統合プラットフォーム コマンド リファレンス
PKI 設定	PKI 展開での Cisco IOS 証明書サーバの設定および管理

シスコのテクニカル サポート

説明	リンク
シスコのサポート Web サイトでは、シスコの製品やテクノロジーに関するトラブルシューティングにお役立ていただけるように、マニュアルやツールをはじめとする豊富なオンライン リソースを提供しています。 お使いの製品のセキュリティ情報や技術情報を入手するために、Cisco Notification Service（Field Notice からアクセス）、Cisco Technical Services Newsletter、Really Simple Syndication（RSS）フィードなどの各種サービスに加入できます。 シスコのサポート Web サイトのツールにアクセスする際は、Cisco.com のユーザ ID およびパスワードが必要です。	http://www.cisco.com/support

SSH 認証用の X.509v3 証明書の機能履歴

次の表に、このモジュールで説明する機能のリリースおよび関連情報を示します。

これらの機能は、特に明記されていない限り、導入されたリリース以降のすべてのリリースで使用できます。

リリース	機能	機能情報
Cisco IOS Release 15.2(7)E3k	SSH 認証の X.509v3 証明書	SSH認証のX.509v3証明書機能は、サーバ内でX.509v3デジタル証明書を使用し、SSHサーバ側でユーザ認証を使用します。

Cisco Feature Navigator を使用すると、プラットフォームおよびソフトウェアイメージのサポート情報を検索できます。Cisco Feature Navigator には、http://www.cisco.com/go/cfn [英語] からアクセスします。

SSH 認証用の X.509v3 証明書の機能履歴