



セキュア シェル コンフィギュレーション ガイド

First Published: 2025-08-25

Last Modified: 2025-08-25

Americas Headquarters

Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
<http://www.cisco.com>
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: <https://www.cisco.com/c/en/us/about/legal/trademarks.html>. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© Cisco Systems, Inc. All rights reserved.



CONTENTS

CHAPTER 1

セキュア シェル 1

セキュアシェルの機能履歴 1

セキュアシェルとは 2

SSH の機能 2

SSH サーバ 2

SSH 統合クライアント 2

RSA 認証のサポート 3

SSH 設定時の注意事項 3

Strict-KEX 4

セキュア シェルを設定するための前提条件 4

セキュア シェルの設定に関する制約事項 5

セキュア シェルの設定 6

SSH を実行するためのデバイスの設定 6

SSH サーバーの設定 7

SSH クライアントの呼び出し 8

セキュアシェルの設定例 9

例: SSH サーバーの設定 9

例: SSH クライアントの呼び出し 10

SSH の確認 10

CHAPTER 2

SSH バージョン 2 13

セキュアシェルバージョン 2 の機能履歴 13

セキュアシェルバージョン 2 とは 14

SSHv2 の拡張機能 14

RSA キー認証	15
SSH Ed25519 暗号化	15
セキュアシェルバージョン 2 の前提条件	16
セキュアシェルバージョン 2 の制約事項	17
設定時の注意事項	17
セキュアシェルバージョン 2 の設定方法	18
ホスト名およびドメイン名を使用した SSH バージョン 2 のデバイス設定	18
RSA キー ペアを使用した SSH バージョン 2 のデバイス設定	19
ユーザー認証を実行するための Cisco SSH サーバーの設定	20
サーバー認証を実行するための Cisco IOS SSH クライアントの設定	23
リモート デバイスとの暗号化セッションの開始	25
セキュア シェル接続のステータスの確認	26
例	26
セキュアシェルバージョン 2 のステータスの確認	27
例	27
セキュア シェル バージョン 2 のモニタリングと維持	28
例	28
セキュアシェルバージョン 2 の設定例	31
例: セキュア シェル バージョン 2 の設定	31
例: セキュア シェル バージョン 1 および 2 の設定	31
例: リモート デバイスでの暗号化セッションの開始	31
例: SNMP トラップの設定	31
例: SNMP のデバッグ	31
例: SSH のデバッグの強化	32

CHAPTER 3

SSH 認証の X.509v3 証明書	35
SSH 認証用の X.509v3 証明書の機能履歴	35
SSH 認証の X.509v3 証明書	36
SSH 認証用 X.509v3 証明書の機能	36
SSH 認証の X.509v3 証明書の前提条件	37
SSH 認証の X.509v3 証明書の制約事項	37

SSH 認証用の X.509v3 証明書の設定方法	37
ユーザー認証用のデジタル証明書を確認するための SSH サーバーの設定	39
トラストポイント認証の設定とデバイス証明書の作成	41
デジタル証明書を使用したサーバーユーザー認証の設定確認	43
SSH 認証用の X.509v3 証明書の設定例	44



CHAPTER 1

セキュア シェル

- [セキュアシェルの機能履歴, on page 1](#)
- [セキュアシェルとは, on page 2](#)
- [SSH の機能, on page 2](#)
- [SSH 設定時の注意事項, on page 3](#)
- [Strict-KEX, on page 4](#)
- [セキュア シェルを設定するための前提条件, on page 4](#)
- [セキュア シェルの設定に関する制約事項, on page 5](#)
- [セキュア シェルの設定, on page 6](#)
- [セキュアシェルの設定例, on page 9](#)

セキュアシェルの機能履歴

次の表に、このモジュールで説明する機能のリリース情報とプラットフォームサポート情報を示します。

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	機能名と説明	サポートされるプラットフォーム
Cisco IOS XE 26.2.1ea	SSH: SSH 機能のサポートが導入されました。	Cisco C9550 シリーズ スマートスイッチ
Cisco IOS XE 17.18.1	SSH: SSH は、Telnet や Berkeley r-tools などの安全性の低いリモート接続ツールの暗号化された安全な代替手段を提供するアプリケーションおよびプロトコルです。	Cisco C9350 シリーズ スマートスイッチ Cisco C9610 シリーズ スマートスイッチ

セキュアシェルとは

セキュアシェル（SSH）は、TelnetやBerkeley r-toolsなどの安全性の低いリモート接続ツールの暗号化された安全な代替手段を提供するアプリケーションおよびプロトコルです。

セッションを

- 標準暗号化メカニズムを使用して保護し、
- デバイス認証および暗号化を提供して、
- 保護されていないネットワークを介したセキュアな通信を可能にします。

SSH でサポートされている認証方式は次のとおりです。

- TACACS+
- RADIUS
- ローカル認証および許可

セキュアで認証および暗号化されたリモートアクセスを提供するために、SSHの実装には、プライマリアプリケーション、メカニズム、暗号化アルゴリズム、プロトコル拡張などの機能が含まれています。以降では、これらの機能について説明します。

SSH の機能

SSH サーバ



Note 特に明記しない限り、「SSH」という用語は「SSH バージョン 1」のみを意味します。

SSH サーバーの機能によって、SSH クライアントはシスコデバイスに対してセキュアで暗号化された接続を実行できます。インバウンド Telnet 接続と同様の機能が提供されますが、シスコソフトウェア認証で使用する強力な暗号化が可能です。Cisco ソフトウェアの SSH サーバーは、市販の一般的な SSH クライアントと相互運用できます。

SSH 統合クライアント

SSH 統合クライアントは、SSH プロトコル上で動作し、デバイス認証および暗号化を提供するアプリケーションです。シスコデバイスから SSH サーバーを実行している別のデバイスに対してセキュアで暗号化された接続を確立でき、アウトバウンド Telnet 接続と同様の機能で暗号化されます。シスコソフトウェアの SSH クライアントは、市販の一般的な SSH サーバーと使用します。

SSH クライアントでサポートされている暗号は次のとおりです。

- データ暗号規格（DES）
- 3DES

SSH クライアントはパスワード認証をサポートしています。



Note SSH クライアント機能を使用できるのは、SSH サーバーが有効になっている場合だけです。

SSH サーバー機能を使用すると、SSH クライアントはセキュアに接続できます。SSH 統合クライアント機能を使用すると、デバイスは別の SSH サーバーへのセキュアな接続を開始できます。両方が連携して、安全でない可能性のあるネットワーク上でセキュアな通信を確保します。

RSA 認証のサポート

セキュアシェル（SSH）クライアントで使用できる Rivest、Shamir、Adleman（RSA）認証は、シスコソフトウェアの SSH サーバではデフォルトでサポートされていません。

セキュアシェル（SSH）統合クライアント機能は、SSH プロトコル上で動作し、デバイスの認証および暗号化を実現するアプリケーションです。SSH クライアントによって、シスコデバイスは別のシスコデバイスなど SSH サーバを実行するデバイスに対して、セキュアで暗号化された接続を実行できます。この接続は、接続が暗号化される点を除いて Telnet のアウトバウンド接続と同様の機能を提供します。SSH クライアントは、認証および暗号化により、保護されていないネットワーク上でもセキュアな通信ができます。

SSH サーバおよび SSH 統合クライアントは、スイッチ上で実行されるアプリケーションです。SSH サーバは、このリリースでサポートされている SSH クライアントおよび、他社製の SSH クライアントと使用します。SSH クライアントは、市販の一般的な SSH サーバと連動します。SSH クライアントは、Data Encryption Standard（DES）、3DES、およびパスワード認証の暗号をサポートします。

ユーザ認証は、デバイスに対する Telnet セッションの認証と同様に実行されます。SSH は、次のユーザ認証方式もサポートします。

- TACACS+
- RADIUS
- ローカル認証および許可

SSH 設定時の注意事項

- SSHv2 サーバは、SSHv1 サーバで生成される RSA キーのペアを使用できます（逆の場合も同様です）。

- SSH サーバーがアクティブスイッチ上で動作しており、アクティブスイッチに障害が発生した場合、新しいアクティブスイッチは、以前のアクティブスイッチによって生成された RSA キーペアを使用します。
- 暗号キー生成 **rsa** コマンドを入力した後に CLI エラーメッセージが表示される場合は、ホスト名とドメインを再設定してから、もう一度コマンドを入力してください。
- RSA キーペアを生成するときに「No host name specified」というメッセージが表示される場合は、グローバル コンフィギュレーション モードで **hostname** コマンドを使用してホスト名を設定します。
- RSA キーペアを生成するときに「No domain specified」というメッセージが表示される場合は、グローバル コンフィギュレーション モードで **ip domain name** コマンドを使用して IP ドメイン名を設定します。
- ローカル認証および許可の方法を設定する場合に、コンソール上で AAA が無効になっていることを確認します。

Strict-KEX

Strict-KEX は、予期しないパケットやシーケンス外のパケットによって接続がすぐに切断されるようにし、キー交換後にシーケンス番号をリセットすることで、セキュリティを強化するプロトコル拡張です。継続的な保護を提供するために、strict-KEX はデフォルトで有効になっており、無効にすることはできません。

セキュア シェルを設定するための前提条件

セキュアで信頼性の高い操作を保証するために、セキュアシェル（SSH）を設定する前に、すべてのデバイスおよびソフトウェアの前提条件を満たしていることを確認します。

一般的な SSH 前提条件

デバイスで SSH またはセキュアコピープロトコル（SCP）を有効にする前に、次の要件を確認します。

- SSH を動作させるには、スイッチに Rivest、Shamir、および Adleman（RSA）の公開キーと秘密キーのペアが必要です。これは SSH が必要な Secure Copy Protocol（SCP）も同様に、セキュアな転送を実現させるには、これらのキーのペアが必要です。
- デバイスに必要なイメージをダウンロードします。セキュアシェル（SSH）サーバーには、IPsec（データ暗号規格（DES）または 3DES）の暗号化ソフトウェアイメージが必要です。SSH クライアントには、IPsec（DES または 3DES）の暗号化ソフトウェアイメージが必要です。
- グローバル コンフィギュレーション モードで **hostname** および **ip domain name** コマンドを使用して、デバイスのホスト名とホストドメインを設定します。

- デバイスの Rivest, Shamir、および Adleman (RSA) キーペアを生成します。このキーペアは、**crypto key generate rsa** コマンドを特権 EXEC モードで入力すると、自動的に SSH とリモート認証を有効にします。

認証および互換性の要件

適切な SSH 操作を確保するために、次の認証および互換性の要件を満たします。

- ローカルアクセスまたはリモートアクセス用にユーザー認証を設定します。認証、許可、アカウントिंग (AAA) の有無に関係なく、認証を設定できます。
- セキュアシェル (SSH) サーバーには、IPsec (データ暗号規格 (DES) または 3DES) の暗号化ソフトウェアイメージが必要です。SSH クライアントには、IPsec (DES または 3DES) の暗号化ソフトウェアイメージが必要です。
- Strict-KEX が動作するには、デバイスで SSH サポートが有効な Cisco IOS-XE を実行している必要があります。
- Strict-KEX が効果的に機能するには、クライアントデバイスとサーバーデバイスの両方が strict-kex アルゴリズムと互換性がある必要があります。

特記事項



Note 特に明記しない限り、「SSH」という用語は「SSH バージョン 1」のみを意味します。



Note RSA キーのペアを削除するには、**crypto key zeroize rsa** グローバル コンフィギュレーション コマンドを使用します。RSA キーペアを削除すると、SSH サーバーは自動的にディセーブルになります。

セキュア シェルの設定に関する制約事項

セキュアシェル (SSH) サーバーと SSH クライアントは、Data Encryption Standard (DES) (56 ビット) および 3DES (168 ビット) データ暗号化ソフトウェアイメージのみでサポートされます。DES ソフトウェア イメージの場合、使用できる暗号化アルゴリズムは DES だけです。3DES ソフトウェア イメージの場合、DES と 3DES の両方の暗号化アルゴリズムを使用できます。

- 実行シェルは、唯一サポートされるアプリケーションです。
- ログイン バナーはセキュア シェル バージョン 1 ではサポートされません。セキュア シェル バージョン 2 ではサポートされています。
- SFTP サーバーはサポートされていません。



Note 特に明記しない限り、「SSH」という用語は「SSH バージョン 1」のみを意味します。

セキュア シェルの設定

セキュアシェル（SSH）を設定して、デバイスへのセキュアなリモートアクセスを有効にし、管理セッションを不正アクセスから保護する方法を学習します。

SSH を実行するためのデバイスの設定

認証、ホスト名、ドメイン名を設定し、RSA キーを生成することにより、SSH を実行するようにデバイスを設定します。

この手順では、必要なすべての設定が行われていることを確認することにより、SSH を使用したデバイスへのセキュアなリモートアクセスが可能になります。

Before you begin

ローカルアクセスまたはリモートアクセス用にユーザ認証を設定します。この手順は必須です。

Procedure

Step 1 enable

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2 configure terminal

Example:

```
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。

Step 3 hostname *hostname*

Example:

```
Device(config)# hostname your_hostname
```

デバイスのホスト名および IP ドメイン名を設定します。

Note

この手順を実行するのは、デバイスを SSH サーバとして設定する場合だけです。

Step 4 ip domain name *domain_name*

Example:

```
Device(config)# ip domain name your_domain
```

デバイスのホストドメインを設定します。

Step 5 **exit**

Example:

```
Device(config)# exit
```

グローバル 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。

Step 6 **crypto key generate rsa**

Example:

```
Device# crypto key generate rsa
```

デバイス上でローカルおよびリモート認証用に SSH サーバをイネーブルにし、RSA キー ペアを生成します。デバイスの RSA キー ペアを生成すると、SSH が自動的にイネーブルになります。

最小モジュラス サイズは、1024 ビットにすることを推奨します。

RSA キーのペアを生成する場合に、モジュラスの長さの入力を求められます。モジュラスが長くなるほど安全ですが、生成と使用に時間がかかります。

Note

この手順を実行するのは、デバイスを SSH サーバとして設定する場合だけです。

Step 7 **show ip ssh**

Example:

```
Device# show ip ssh
```

(任意) SSH サーバーが有効であることを確認し、SSH 接続のバージョンおよび設定データを表示します。

SSH サーバーの設定



Note

特に明記しない限り、「SSH」という用語は「SSH バージョン 1」のみを意味します。

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2 **configure terminal**

Example:

```
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。

Step 3 **ip ssh { time-out seconds | authentication-retries integer }**

Example:

```
Device(config)# ip ssh time-out 30
```

セキュアシェル（SSH）制御パラメータを設定します。

Note

Note このコマンドは、ユーザーに表示するパスワードプロンプトの回数を設定するためにも使用できます。この数値は、次の 2 つの値の低い方です。

- **ssh -o numberofpasswordprompt** コマンドを使用してクライアントから提案された値。
- **ip ssh authentication-retries integer** コマンドを使用してデバイスに設定された値に 1 を加えた値。

Step 4 **ip ssh rekey { time time | volume volume }**

Example:

```
Device(config)# ip ssh rekey time 108
```

（任意）SSH の時間ベースのキー再生成またはボリュームベースのキー再生成を設定します。

Step 5 **exit**

Example:

```
Device(config)# exit
```

グローバル 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。

Step 6 **show ip ssh**

Example:

```
Device# show ip ssh
```

（任意）SSH サーバーが有効であることを確認し、SSH 接続のバージョンおよび設定データを表示します。

SSH クライアントの呼び出し



Note 特に明記しない限り、「SSH」という用語は「SSH バージョン 1」のみを意味します。

セキュアシェル（SSH）クライアントを呼び出すには、次の手順を実行します。SSH クライアントはユーザー EXEC モードで実行されます。設定作業は特にありません。

Procedure

Step 1 enable

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2 `ssh -l username -vrf vrf-name ip-address`

Example:

```
Device# ssh -l user1 -vrf vrf1 192.0.2.1
```

SSH クライアントを呼び出し、指定した仮想ルーティングおよび転送（VRF）インスタンスの IP ホストまたはアドレスに接続します。

セキュアシェルの設定例

次の手順と設定例を使用して、デバイスでセキュアシェル（SSH）を設定する方法を学習します。

例：SSH サーバーの設定

次に、サーバーに設定されたセキュアシェル（SSH）制御パラメータの例を示します。この例では、30 秒のタイムアウト間隔が指定されています。このタイムアウト間隔は、SSH ネゴシエーションフェーズで使用されます。



Note 特に明記しない限り、「SSH」という用語は「SSH バージョン 1」のみを意味します。

Procedure

SSH サーバーの設定

```
Device> enable
Device# configure terminal
Device(config)# ip ssh timeout 30
Device(config)# end
```

例: SSH クライアントの呼び出し

次の例では、指定された Virtual Routing and Forwarding (VRF) インスタンスの IP アドレス 192.0.2.1 に接続するためにセキュアシェル (SSH) クライアントが呼び出されます。



Note 特に明記しない限り、「SSH」という用語は「SSH バージョン 1」のみを意味します。

Procedure

SSH クライアントを呼び出します。

```
Device> enable
Device# ssh -l user1 -vrf vrf1 192.0.2.1
```

SSH の確認



Note 特に明記しない限り、「SSH」という用語は「SSH バージョン 1」のみを意味します。

Procedure

Step 1 **show ip ssh** コマンドを使用して、SSH が有効になっていることを確認し、バージョンと設定データを表示します。

Example:

```
Device# show ip ssh
SSH Enabled - version 1.5
Authentication timeout: 120 secs; Authentication retries: 3
```

上記の出力は、SSH がデバイスで有効になっていることを示しています。

Step 2 SSH が無効になっているかどうかを確認するには、**show ip ssh** コマンドを使用します。

Example:

```
Device# show ip ssh
%SSH has not been enabled
```

上記の出力は、SSH がデバイスで有効になっていないことを示しています。

Step 3 SSH サーバー接続のステータスを確認するには、**show ssh** コマンドを使用します。

Example:

```
Device# show ssh
      Connection Version Encryption State Username
      0 1.5 3DES Session Started guest
```

この出力は、SSH を有効にしたときのデバイス上の SSH サーバー接続を示しています。

Step 4 **show ssh** コマンドを使用して、SSH サーバー接続が実行されているかどうかを確認します。

Example:

```
Device# show ssh
      %No SSH server connections running.
```

この出力は、デバイスで実行中の SSH サーバー接続がないことを示します。



CHAPTER 2

SSH バージョン 2

- セキュアシェルバージョン 2 の機能履歴, on page 13
- セキュアシェルバージョン 2 とは, on page 14
- SSHv2 の拡張機能, on page 14
- RSA キー認証, on page 15
- SSH Ed25519 暗号化, on page 15
- セキュアシェルバージョン 2 の前提条件, on page 16
- セキュアシェルバージョン 2 の制約事項, on page 17
- 設定時の注意事項, on page 17
- セキュアシェルバージョン 2 の設定方法, on page 18
- セキュアシェルバージョン 2 の設定例, on page 31

セキュアシェルバージョン 2 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情報とプラットフォームサポート情報を示します。

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	機能名と説明	サポートされるプラットフォーム
Cisco IOS XE 26.2.1ea	SSH バージョン 2: SSH バージョン 2 の機能サポートが導入されました。	Cisco C9550 シリーズ スマートスイッチ

リリース	機能名と説明	サポートされるプラットフォーム
Cisco IOS XE 17.18.1	SSH バージョン 2: SSH バージョン 2 は、非信頼ネットワークで強力な暗号化と堅牢な認証による、セキュアなリモートアクセス、コマンド実行、およびファイル転送 (SCP) を提供します。	Cisco C9350 シリーズ スマートスイッチ Cisco C9610 シリーズ スマートスイッチ

セキュアシェルバージョン 2 とは

セキュアシェルバージョン 2 (SSH) は、非信頼ネットワークを介したセキュアなリモートアクセスとコマンド実行を可能にするネットワークプロトコルです。TCP など、信頼性の高いトランスポート層に強力な認証機能と暗号化機能を提供します。SSH には、セキュアなファイル転送用のセキュアコピープロトコル (SCP) も含まれます。

SSHv2 の拡張機能

- **Virtual Routing and Forwarding (VRF)-Aware SSH:** `ssh` コマンドは、SSH クライアント側の機能に VRF 認識を追加するように拡張されました。これにより、クライアント内の VRF インスタンス名は、正しいルーティングテーブルを検索して接続を確立するための IP アドレスを提供できます。



Note VRF-Aware SSH のサポートは、ソフトウェアリリースによって異なります。

- **Diffie-Hellman (DH) グループ交換のサポート:** Cisco SSH 実装では、従来 768 ビットのモジュラスが使用されていました。DH グループ 14 (2048 ビット) およびグループ 16 (4096 ビット) 暗号化アプリケーションに対応するため、より大きなキーサイズの必要性が高まり、優先 DH グループを確立するクライアントとサーバー間のメッセージ交換が必要になっています。 `ip ssh dh min size` コマンドは、SSH サーバー上の最小モジュラスサイズを設定します。
- **デバッグの機能拡張:** `debug ip ssh` コマンドは、デバッグを簡素化するために機能拡張されました。以前は、SSH 関連のすべてのデバッグメッセージを出力していました。現在では、キーワードを使用してコマンドを設定し、メッセージを特定の情報に制限できるようになりました。

RSA キー認証

ユーザー認証: RSA ベースのユーザー認証は、各ユーザーに関連付けられている秘密キー/公開キーのペアを使用します。ユーザーはこのキーのペアをクライアントで生成し、公開キーを Cisco SSH サーバーで設定します。Cisco Confidential 認証時に、SSH ユーザーは秘密キーを使用して暗号化された署名を提供します。この署名とユーザーの公開キーは、SSH サーバーに送信されます。サーバーは指定された公開キーに対してハッシュを計算し、一致するエントリが存在するかどうかを判断します。一致が見つかった場合、公開キーを使用した RSA ベースのメッセージ検証が実行され、暗号化された署名に基づいてユーザーが認証または拒否されます。

サーバー認証: Cisco SSH クライアントは、キー交換フェーズ中に交換されたサーバーのホストキーを使用して、セッションの確立時に SSH サーバーを認証します。SSH が有効なときに作成されるこれらのキーは、SSH サーバーを識別するために使用され、認証用にクライアントで設定する必要があります。クライアントが SSH セッションを確立しようとする、キー交換メッセージの一部として、サーバーの署名を受信します。厳密なホストキーのチェックフラグがクライアント側で有効になっている場合、そのサーバーに対応するホストキーエントリがあるかどうかをクライアントが確認します。一致が見つかったら、クライアントはサーバーのホストキーを使用して署名の検証を試行します。サーバーの認証に成功すると、セッションの確立処理が続行されます。失敗すると、セッションは終了し、「Server Authentication Failed」というメッセージが表示されます。



Note

- サーバーに公開キーを保存すると、メモリを消費します。したがって、SSH サーバーで設定できる公開キーの数は、1 ユーザーに最大 2 つの公開キーを作成した場合 10 ユーザーに制限されます。
- シスコサーバーは、RSA ベースのユーザー認証をサポートしています。ただし、シスコクライアントは認証方式として公開キーを提案できません。RSA ベースの認証に対する OpenSSH クライアントからの要求を Cisco サーバーが受信した場合、サーバーは認証要求を受け入れます。
- 適切なサーバー認証の場合、サーバーの RSA 公開キーを手動で設定し、Cisco SSH クライアント側で `ip ssh stricthostkeycheck` コマンドを設定します。

SSH Ed25519 暗号化

Ed25519 は、次の機能を提供する公開キー署名システムです。

- 単一署名の高速検証
- 高速バッチ検証
- 高速署名

- 高速キー生成
- 高いセキュリティレベル
- フールプルーフのセッションキー
- 小さな署名（64 バイト）
- 小さなキー（32 バイト）

SSH およびスイッチアクセス

シェル（SSH）は、デバイスへのセキュアなリモート接続を提供し、Telnet よりも強力な暗号化と認証を提供します。このソフトウェアリリースは、SSH バージョン 2（SSHv2）をサポートします。SSH 機能は IPv4 および IPv6 で一貫しており、IPv6 トランスポート上でリモート IPv6 ノードとのセキュアで暗号化された接続を可能にします。

SNMP トラップ生成の Simple Network Management Protocol（SNMP）トラップは、ソフトウェアリリースに応じて、SSH セッションの終了時に自動生成されます。これらのトラップを有効にするには、次を実行します。

- SNMP トラップが有効になっていることを確認します。
- `debug snmp packet` コマンドを使用して SNMP デバッグを有効にし、トラップを表示します。
- `snmp-server host` コマンドを設定する場合は、SSH（または Telnet）クライアントをホストしている PC の IP アドレスを指定し、SSH サーバーへの IP 接続を確認します。トラップ情報には、送信バイト数や SSH セッションで使用されたプロトコルなどの情報が含まれます。

SSH キーボードインタラクティブ認証

SSH キーボードインタラクティブ認証機能は、SSH での汎用メッセージ認証とも呼ばれ、これによってユーザー入力のみを必要とするさまざまな認証メカニズムを実装できます。この機能は自動的に有効になります。サポートされている認証方式は次のとおりです。

- パスワード
- SecurID およびハードウェアトークン（サーバーのチャレンジに番号または文字列で応答する）
- プラグイン可能な認証モジュール（PAM）
- S/KEY（およびその他の使い捨てキー）

セキュアシェルバージョン 2 の前提条件

- ご使用のデバイスに必要なイメージがロードされていることを確認します。SSH サーバーには、ソフトウェアリリースに応じた k9（Triple Data Encryption Standard（3DES））ソフトウェアイメージが必要です。

- SSH バージョン 2 をサポートする SSH リモートデバイスを使用してシスコデバイスに接続します。
- デバイスに認証、許可、アカウントティング（AAA）を設定します。SCP は AAA に依存して正しく機能するため、AAA を設定すると SSH サーバーで SCP が有効になります。

**Note**

SSH バージョン 2 サーバーおよびクライアントは、特定のシスコソフトウェアリリースでサポートされています。SSH クライアントは、SSH バージョン 1 と SSH バージョン 2 の両方のプロトコルをサポートし、ソフトウェアリリースに応じて k9 イメージで使用できます。

セキュアシェルバージョン 2 の制約事項

- SSH のサーバーとクライアントには、Triple Data Encryption Standard（3DES）ソフトウェアイメージが必要です。
- サポートされるアプリケーションは、実行シェル、リモートコマンドの実行、Secure Copy Protocol（SCP）のみです。
- Rivest、Shamir、および Adleman（RSA）キー生成は SSH サーバーに必要です。SSH クライアントでは RSA キーの生成は必要ありません。
- RSA キーペアのサイズは、768 ビット以上である必要があります。
- サポートされていない機能は、次のとおりです。
 - ポート フォワーディング。
 - Compression

設定時の注意事項

SSH バージョン 2 を設定するには、**ip ssh version** コマンドを使用します。このコマンドを設定しない場合、デフォルトで SSH は互換モードで動作します。SSH バージョン 1 と SSH バージョン 2 両方の接続が利用できます。

**Note**

SSH バージョン 1 は、標準化されたプロトコルではありません。デバイスがこの未定義のプロトコルに戻らないようにするには、**ip ssh version** コマンドを使用してバージョン 2 を指定します。

ip ssh rsa keypair-name コマンドを使用すると、設定した Rivest、Shamir、および Adleman（RSA）キーを使用して SSH 接続を実行できます。このコマンドを使用すると、以前のホスト名とドメイン

ン名の設定なしで SSH を有効にできます。SSH は、指定されたキーペアが存在するか、または後で生成される場合に有効になります。



Note ログインバナーは SSH バージョン 2 でサポートされますが、SSH バージョン 1 ではサポートされません。

セキュアシェルバージョン 2 の設定方法

ホスト名およびドメイン名を使用した SSH バージョン 2 のデバイス設定

Procedure

- Step 1** `enable`
- Example:**
`Device> enable`
- 特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。
- Step 2** `configure terminal`
- Example:**
`Device# configure terminal`
- グローバル設定モードを開始します。
- Step 3** `hostname name`
- Example:**
`Device(config)# hostname catalyst9k`
- デバイスのホスト名を設定します。
- Step 4** `ip domain name name`
- Example:**
`catalyst9k(config)# ip domain name example.com`
- デバイスのドメイン名を設定します。
- Step 5** `crypto key generate rsa`
- Example:**
`catalyst9k(config)# crypto key generate rsa`
- ローカルおよびリモート認証用に SSH サーバをイネーブルにします。

Step 6 `ip ssh [time-out seconds | authentication-retries integer ]`

Example:

```
catalyst9k(config)# ip ssh time-out 120
```

（任意）デバイス上で SSH 制御変数を設定します。

Step 7 `ip ssh version [1 | 2 ]`

Example:

```
catalyst9k(config)# exit
```

（任意）デバイスで実行する SSH のバージョンを指定します。

Step 8 `exit`

Example:

```
catalyst9k(config)# ip ssh version 1
```

グローバル コンフィギュレーション モードを終了し、特権 EXEC モードを開始します。

- デフォルトホストに戻るには、`no hostname` コマンドを使用します。

RSA キー ペアを使用した SSH バージョン 2 のデバイス設定

Procedure

Step 1 `enable`

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2 `configure terminal`

Example:

```
Device# configure terminal
```

グローバル設定モードを開始します。

Step 3 `ip ssh rsa keypair-name keypair-name`

Example:

```
Device(config)# ip ssh rsa keypair-name sshkeys
```

SSH に使用する RSA キー ペアを指定します。

Note

シスコ デバイスには複数の RSA キー ペアを設定できます。

Step 4 `crypto key generate rsa usage-keys label key-label modulus modulus-size`

Example:

```
Device(config)# crypto key generate rsa usage-keys label sshkeys modulus 768
```

デバイスでローカルおよびリモート認証を行う SSH サーバを有効にします。

- SSH バージョン 2 では、絶対サイズは 768 ビット以上である必要があります。

Note

RSA キー ペアを削除するには、`crypto key zeroize rsa` コマンドを使用します。RSA キー ペアを削除すると、SSH サーバは自動的に無効になります。

Step 5 `ip ssh [time-out seconds | authentication-retries integer ]`

Example:

```
Device(config)# ip ssh time-out 12
```

デバイス上で SSH 制御変数を設定します。

Step 6 `ip ssh version 2`

Example:

```
Device(config)# ip ssh version 2
```

デバイスで実行する SSH のバージョンを指定します。

Step 7 `exit`

Example:

```
catalyst9k(config)# ip ssh version 1
```

グローバル コンフィギュレーション モードを終了し、特権 EXEC モードを開始します。

ユーザー認証を実行するための Cisco SSH サーバーの設定

Procedure

Step 1 `enable`

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2 `configure terminal`

Example:

```
Device# configure terminal
```

グローバル設定モードを開始します。

Step 3 `hostname name`

Example:

```
Device(config)# hostname host1
```

ホスト名を指定します。

Step 4

`ip domain name name`

Example:

```
host1(config)# ip domain name name1
```

Cisco ソフトウェアで使用するデフォルトのドメイン名を定義し、不完全なホスト名のドメインを補完します。

Step 5

`crypto key generate rsa`

Example:

```
host1(config)# crypto key generate rsa
```

RSA キー ペアを生成します。

Step 6

`ip ssh pubkey-chain`

Example:

```
host1(config)# ip ssh pubkey-chain
```

SSH サーバー上のユーザーおよびサーバー認証用に公開キーを設定し、公開キー コンフィギュレーション モードを開始します。

- サーバーに保存されている公開キーが、クライアントに保存されている公開キーまたは秘密キーのペアを使用して検証されると、ユーザー認証は成功です。

Step 7

`username username`

Example:

```
host1(conf-ssh-pubkey)# username user1
```

SSH ユーザ名を設定し、公開キー ユーザ コンフィギュレーション モードを開始します。

Step 8

`key-string`

Example:

```
host1(conf-ssh-pubkey-user)# key-string
```

リモートピアの公開キーを指定し、公開キー データ コンフィギュレーション モードを開始します。

Note

オープン SSH クライアントから（言い換えると `.ssh/id_rsa.pub` ファイルから）公開キー値を取得できます。

Step 9

`exit`

Example:

```
host1(conf-ssh-pubkey-data)# exit
```

公開キー データ コンフィギュレーション モードを終了し、公開キー サーバ コンフィギュレーション モードを開始します。

Step 10

`key-hash key-type key-name`

Example:

```
host1(conf-ssh-pubkey-data)# key-hash ssh-rsa key1
or
host1(conf-ssh-pubkey-data)# key-hash ssh-ed25519 DFEAF10390E560AEA745CCBA53E044ED
```

(任意) SSH キー タイプとバージョンを指定します。

- 秘密キー/公開キーペアの設定では、キータイプを `ssh-rsa` または `ssh-ed25519` に指定できます。
- この手順は、`key-string` コマンドが設定されている場合にのみ任意です。
- `key-string` コマンドまたは `key-hash` コマンドのいずれかを設定する必要があります。

Note

公開キースtringのハッシュを計算するには、ハッシュ処理ソフトウェアを使用します。また、別のシスコ デバイスからハッシュ値をコピーすることもできます。初めて公開キーデータを入力する場合、`key-string` コマンドを使用して公開キーデータを入力することを推奨します。

Step 11 `end`

Example:

```
host1(conf-ssh-pubkey-server)# end
```

公開キー 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。

Step 12 `configure terminal`

Example:

```
host1# configure terminal
```

グローバル設定モードを開始します。

Step 13 `ip ssh stricthostkeycheck`

Example:

```
host1(config)# ip ssh stricthostkeycheck
```

サーバ認証が実行されることを確認します。

- 障害が発生すると、接続は終了します。
- デフォルトホストに戻るには、`no hostname` コマンドを使用します。

Step 14 `end`

Example:

```
host1(config)# end
```

グローバル 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。

サーバー認証を実行するための Cisco IOS SSH クライアントの設定

Procedure

Step 1

`enable`

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2

`configure terminal`

Example:

```
Device# configure terminal
```

グローバル設定モードを開始します。

Step 3

`hostname name`

Example:

```
Device(config)# hostname host1
```

ホスト名を指定します。

Step 4

`ip domain name name`

Example:

```
host1(config)# ip domain name name1
```

Cisco ソフトウェアで使用するデフォルトのドメイン名を定義し、不完全なホスト名のドメインを補完します。

Step 5

`crypto key generate rsa`

Example:

```
host1(config)# crypto key generate rsa
```

RSA キー ペアを生成します。

Step 6

`ip ssh pubkey-chain`

Example:

```
host1(config)# ip ssh pubkey-chain
```

SSH サーバー上のユーザーおよびサーバー認証用に公開キーを設定し、公開キー コンフィギュレーション モードを開始します。

Step 7

`server server-name`

Example:

```
host1(conf-ssh-pubkey)# server server1
```

デバイスでの公開キー認証について SSH サーバを有効にし、公開キー サーバ コンフィギュレーション モードを開始します。

Step 8 `key-string`**Example:**

```
host1(conf-ssh-pubkey-server)# key-string
```

リモートピアの公開キーを指定し、公開キー データ コンフィギュレーション モードを開始します。

Note

オープン SSH クライアントから（言い換えると `.ssh/id_rsa.pub` ファイルから）公開キー値を取得できます。

Step 9 `exit`**Example:**

```
host1(conf-ssh-pubkey-data)# exit
```

公開キー データ コンフィギュレーション モードを終了し、公開キー サーバ コンフィギュレーション モードを開始します。

Step 10 `key-hash key-type key-name`**Example:**

```
host1(conf-ssh-pubkey-data)# key-hash ssh-rsa key1  
or  
host1(conf-ssh-pubkey-data)# key-hash ssh-ed25519 DFEAF10390E560AEA745CCBA53E044ED
```

（任意）SSH キー タイプとバージョンを指定します。

- 秘密キー/公開キーペアの設定では、キータイプを `ssh-rsa` または `ssh-ed25519` に指定できます。
- この手順は、`key-string` コマンドが設定されている場合にのみ任意です。
- `key-string` コマンドまたは `key-hash` コマンドのいずれかを設定する必要があります。

Note

公開キースtringのハッシュを計算するには、ハッシュ処理ソフトウェアを使用します。また、別のシスコ デバイスからハッシュ値をコピーすることもできます。初めて公開キーデータを入力する場合、`key-string` コマンドを使用して公開キーデータを入力することを推奨します。

Step 11 `end`**Example:**

```
host1(conf-ssh-pubkey-server)# end
```

公開キー 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。

Step 12 `configure terminal`**Example:**

```
host1# configure terminal
```

グローバル設定モードを開始します。

Step 13 `ip ssh stricthostkeycheck`**Example:**

```
host1(config)# ip ssh stricthostkeycheck
```

サーバ認証が実行されることを確認します。

- 障害が発生すると、接続は終了します。
- デフォルトホストに戻るには、`no hostname` コマンドを使用します。

Step 14`end`**Example:**

```
host1(config)# end
```

グローバル 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。

リモート デバイスとの暗号化セッションの開始

**Note**

接続するデバイスは、シスコソフトウェアでサポートされる暗号化アルゴリズムを備えた SSH サーバーをサポートしている必要があります。デバイスで SSH サービスを明示的に有効にする必要はありません。SSH は、アクティブに実行されていない場合でも設定できます。

Procedure**Step 1**`enable`**Example:**

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2

```
ssh [-v {1 | 2}] [-c {aes128-ctr | aes192-ctr | aes256-ctr | aes128-cbc | 3des | aes192-cbc | aes256-cbc}] [-l user-id | -l user-id : vrf-name number ip-address ip-address | -l user-id : rotary number ip-address] [-m {hmac-md5-128 | hmac-md5-96 | hmac-sha1-160 | hmac-sha1-96}] [-o numberofpasswordprompts n] [-p port-num] [ip-addr | hostname] [command | -vrf]
```

Example:

```
Device# ssh -v 2 -c aes256-ctr -m hmac-sha1-96 -l user2 10.76.82.24
```

リモート ネットワーク デバイスとの暗号化されたセッションを開始します。

セキュア シェル接続のステータスの確認

Procedure

Step 1

enable

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2

show ssh

Example:

```
Device# show ssh
```

SSH サーバ接続のステータスを表示します。

Step 3

exit

Example:

```
Device# exit
```

特権 EXEC モードを終了し、ユーザー EXEC モードに戻ります。

例

次の show ssh コマンドの出力例には、バージョン 1 およびバージョン 2 接続の複数の SSH バージョン 1 およびバージョン 2 接続のステータスが表示されています。

```
Device# show ssh
Connection Version Encryption      State      Username
          0         1.5         3DES    Session started    lab
Connection Version Mode Encryption Hmac State
Username
1      2.0 IN aes128-cbc hmac-md5 Session started    lab
1      2.0 OUT aes128-cbc hmac-md5 Session started    lab
```

次の show ssh コマンドの出力例には、バージョン 2 接続（バージョン 1 接続なし）の複数の SSH バージョン 1 およびバージョン 2 接続のステータスが表示されています。

```
Device# show ssh
Connection Version Mode Encryption Hmac State
Username
1 2.0 IN aes128-cbc hmac-md5 Session started lab
1 2.0 OUT aes128-cbc hmac-md5 Session started lab
%No SSHv1 server connections running.
```

次の show ssh コマンドの出力例には、バージョン 1 接続（バージョン 2 接続なし）の複数の SSH バージョン 1 およびバージョン 2 接続のステータスが表示されています。

```
Device# show ssh
Connection Version Encryption      State      Username
```

```
0          1.5          3DES Session started lab
%No SSHv2 server connections running.
```

セキュアシェルバージョン 2 のステータスの確認

Procedure

Step 1

enable

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2

show ip ssh

Example:

```
Device# show ip ssh
```

SSH のバージョンおよび設定データを表示します。

Step 3

exit

Example:

```
Device# exit
```

特権 EXEC モードを終了し、ユーザー EXEC モードに戻ります。

例

次の show ip ssh コマンドの出力例には、有効な SSH のバージョン、認証タイムアウト値、およびバージョン 1 およびバージョン 2 接続の認証の再試行回数が表示されています。

```
Device# show ip ssh
SSH Enabled - version 1.99
Authentication timeout: 120 secs; Authentication retries: 3
```

次の show ip ssh コマンドの出力例には、有効な SSH のバージョン、認証タイムアウト値、およびバージョン 2 接続（バージョン 1 接続なし）の認証の再試行回数が表示されています。

```
Device# show ip ssh
SSH Enabled - version 2.0
Authentication timeout: 120 secs; Authentication retries: 3
```

次の show ip ssh コマンドの出力例には、有効な SSH のバージョン、認証タイムアウト値、およびバージョン 1 接続（バージョン 2 接続なし）の認証の再試行回数が表示されています。

```
Device# show ip ssh
3d06h: %SYS-5-CONFIG_I: Configured from console by console
SSH Enabled - version 1.5
Authentication timeout: 120 secs; Authentication retries: 3
```

セキュア シェル バージョン 2 のモニタリングと維持

Procedure

Step 1

enable

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2

debug ip ssh

Example:

```
Device# debug ip ssh
```

SSH のデバッグを有効にします。

Step 3

debug snmp packet

Example:

```
Device# debug snmp packet
```

デバイスによって送受信されたすべての SNMP パケットのデバッグを有効にします。

Example

What to do next

-

例

次の debug ip ssh コマンドの出力例は、接続が SSH バージョン 2 接続であることを示します。

```
Device# debug ip ssh
00:33:55: SSH1: starting SSH control process
00:33:55: SSH1: sent protocol version id SSH-1.99-Cisco-1.25
00:33:55: SSH1: protocol version id is - SSH-2.0-OpenSSH_2.5.2p2
00:33:55: SSH2 1: send: len 280 (includes padlen 4)
00:33:55: SSH2 1: SSH2_MSG_KEXINIT sent
00:33:55: SSH2 1: ssh_receive: 536 bytes received
00:33:55: SSH2 1: input: packet len 632
00:33:55: SSH2 1: partial packet 8, need 624, maclen 0
00:33:55: SSH2 1: ssh_receive: 96 bytes received
00:33:55: SSH2 1: partial packet 8, need 624, maclen 0
00:33:55: SSH2 1: input: padlen 11
00:33:55: SSH2 1: received packet type 20
00:33:55: SSH2 1: SSH2_MSG_KEXINIT received
00:33:55: SSH2: kex: client->server aes128-cbc hmac-md5 none
00:33:55: SSH2: kex: server->client aes128-cbc hmac-md5 none
00:33:55: SSH2 1: expecting SSH2_MSG_KEXDH_INIT
```

```
00:33:55: SSH2 1: ssh_receive: 144 bytes received
00:33:55: SSH2 1: input: packet len 144
00:33:55: SSH2 1: partial packet 8, need 136, maclen 0
00:33:55: SSH2 1: input: padlen 5
00:33:55: SSH2 1: received packet type 30
00:33:55: SSH2 1: SSH2_MSG_KEXDH_INIT received
00:33:55: SSH2 1: signature length 111
00:33:55: SSH2 1: send: len 384 (includes padlen 7)
00:33:55: SSH2: kex_derive_keys complete
00:33:55: SSH2 1: send: len 16 (includes padlen 10)
00:33:55: SSH2 1: newkeys: mode 1
00:33:55: SSH2 1: SSH2_MSG_NEWKEYS sent
00:33:55: SSH2 1: waiting for SSH2_MSG_NEWKEYS
00:33:55: SSH2 1: ssh_receive: 16 bytes received
00:33:55: SSH2 1: input: packet len 16
00:33:55: SSH2 1: partial packet 8, need 8, maclen 0
00:33:55: SSH2 1: input: padlen 10
00:33:55: SSH2 1: newkeys: mode 0
00:33:55: SSH2 1: received packet type 2100:33:55: SSH2 1: SSH2_MSG_NEWKEYS received
00:33:56: SSH2 1: ssh_receive: 48 bytes received
00:33:56: SSH2 1: input: packet len 32
00:33:56: SSH2 1: partial packet 16, need 16, maclen 16
00:33:56: SSH2 1: MAC #3 ok
00:33:56: SSH2 1: input: padlen 10
00:33:56: SSH2 1: received packet type 5
00:33:56: SSH2 1: send: len 32 (includes padlen 10)
00:33:56: SSH2 1: done calc MAC out #3
00:33:56: SSH2 1: ssh_receive: 64 bytes received
00:33:56: SSH2 1: input: packet len 48
00:33:56: SSH2 1: partial packet 16, need 32, maclen 16
00:33:56: SSH2 1: MAC #4 ok
00:33:56: SSH2 1: input: padlen 9
00:33:56: SSH2 1: received packet type 50
00:33:56: SSH2 1: send: len 32 (includes padlen 13)
00:33:56: SSH2 1: done calc MAC out #4
00:34:04: SSH2 1: ssh_receive: 160 bytes received
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #5 ok
00:34:04: SSH2 1: input: padlen 13
00:34:04: SSH2 1: received packet type 50
00:34:04: SSH2 1: send: len 16 (includes padlen 10)
00:34:04: SSH2 1: done calc MAC out #5
00:34:04: SSH2 1: authentication successful for lab
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #6 ok
00:34:04: SSH2 1: input: padlen 6
00:34:04: SSH2 1: received packet type 2
00:34:04: SSH2 1: ssh_receive: 64 bytes received
00:34:04: SSH2 1: input: packet len 48
00:34:04: SSH2 1: partial packet 16, need 32, maclen 16
00:34:04: SSH2 1: MAC #7 ok
00:34:04: SSH2 1: input: padlen 19
00:34:04: SSH2 1: received packet type 90
00:34:04: SSH2 1: channel open request
00:34:04: SSH2 1: send: len 32 (includes padlen 10)
00:34:04: SSH2 1: done calc MAC out #6
00:34:04: SSH2 1: ssh_receive: 192 bytes received
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #8 ok
00:34:04: SSH2 1: input: padlen 13
00:34:04: SSH2 1: received packet type 98
```

```

00:34:04: SSH2 1: pty-req request
00:34:04: SSH2 1: setting TTY - requested: height 24, width 80; set: height 24,
width 80
00:34:04: SSH2 1: input: packet len 96
00:34:04: SSH2 1: partial packet 16, need 80, maclen 16
00:34:04: SSH2 1: MAC #9 ok
00:34:04: SSH2 1: input: padlen 11
00:34:04: SSH2 1: received packet type 98
00:34:04: SSH2 1: x11-req request
00:34:04: SSH2 1: ssh_receive: 48 bytes received
00:34:04: SSH2 1: input: packet len 32
00:34:04: SSH2 1: partial packet 16, need 16, maclen 16
00:34:04: SSH2 1: MAC #10 ok
00:34:04: SSH2 1: input: padlen 12
00:34:04: SSH2 1: received packet type 98
00:34:04: SSH2 1: shell request
00:34:04: SSH2 1: shell message received
00:34:04: SSH2 1: starting shell for vty
00:34:04: SSH2 1: send: len 48 (includes padlen 18)
00:34:04: SSH2 1: done calc MAC out #7
00:34:07: SSH2 1: ssh_receive: 48 bytes received
00:34:07: SSH2 1: input: packet len 32
00:34:07: SSH2 1: partial packet 16, need 16, maclen 16
00:34:07: SSH2 1: MAC #11 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:07: SSH2 1: send: len 32 (includes padlen 17)
00:34:07: SSH2 1: done calc MAC out #8
00:34:07: SSH2 1: ssh_receive: 48 bytes received
00:34:07: SSH2 1: input: packet len 32
00:34:07: SSH2 1: partial packet 16, need 16, maclen 16
00:34:07: SSH2 1: MAC #12 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:07: SSH2 1: send: len 32 (includes padlen 17)
00:34:07: SSH2 1: done calc MAC out #9
00:34:07: SSH2 1: ssh_receive: 48 bytes received
00:34:07: SSH2 1: input: packet len 32
00:34:07: SSH2 1: partial packet 16, need 16, maclen 16
00:34:07: SSH2 1: MAC #13 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:07: SSH2 1: send: len 32 (includes padlen 17)
00:34:07: SSH2 1: done calc MAC out #10
00:34:08: SSH2 1: ssh_receive: 48 bytes received
00:34:08: SSH2 1: input: packet len 32
00:34:08: SSH2 1: partial packet 16, need 16, maclen 16
00:34:08: SSH2 1: MAC #14 ok
00:34:08: SSH2 1: input: padlen 17
00:34:08: SSH2 1: received packet type 94
00:34:08: SSH2 1: send: len 32 (includes padlen 17)
00:34:08: SSH2 1: done calc MAC out #11
00:34:08: SSH2 1: ssh_receive: 48 bytes received
00:34:08: SSH2 1: input: packet len 32
00:34:08: SSH2 1: partial packet 16, need 16, maclen 16
00:34:08: SSH2 1: MAC #15 ok
00:34:08: SSH2 1: input: padlen 17
00:34:08: SSH2 1: received packet type 94
00:34:08: SSH2 1: send: len 32 (includes padlen 16)
00:34:08: SSH2 1: done calc MAC out #12
00:34:08: SSH2 1: send: len 48 (includes padlen 18)
00:34:08: SSH2 1: done calc MAC out #13
00:34:08: SSH2 1: send: len 16 (includes padlen 6)
00:34:08: SSH2 1: done calc MAC out #14

```

```
00:34:08: SSH2 1: send: len 16 (includes padlen 6)
00:34:08: SSH2 1: done calc MAC out #15
00:34:08: SSH1: Session terminated normally
```

セキュアシェルバージョン 2 の設定例

例: セキュア シェルバージョン 2 の設定

```
Device> enable
Device# configure terminal
Device(config)# ip ssh version 2
Device(config)# end
```

例: セキュア シェルバージョン 1 および 2 の設定

```
Device> enable
Device# configure terminal
Device(config)# no ip ssh version
Device(config)# end
```

例: リモート デバイスでの暗号化セッションの開始

```
Device> enable
Device# ssh -v 2 -c aes256-cbc -m hmac-sha1-160 -l shaship 10.76.82.24
Device# exit
```

例: SNMP トラップの設定

次の例では、SNMP トラップの設定方法を示します。トラップ通知は、SSH セッションが終了すると自動的に生成されます。この例の 10.1.1.1 は SSH クライアントの IP アドレスです。

```
Device> enable
Device# configure terminal
Device(config)# snmp-server trap link switchover
Device(config)# snmp-server host 10.1.1.1 public tty
Device(config)# end
```

例: SNMP のデバッグ

次に、**debug snmp packet** コマンドの出力例を示します。出力には、SSH セッションの SNMP トラップ情報が含まれます。

```
Device1# debug snmp packet

SNMP packet debugging is on
Device1# ssh -l lab 10.0.0.2
Password:

Device2# exit

[Connection to 10.0.0.2 closed by foreign host]
```

例: SSH のデバッグの強化

```

Device1#
*Jul 18 10:18:42.619: SNMP: Queuing packet to 10.0.0.2
*Jul 18 10:18:42.619: SNMP: V1 Trap, ent cisco, addr 10.0.0.1, gentrap 6, spectrap 1
local.9.3.1.1.2.1 = 6
tcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 4
ltcpConnEntry.5.10.0.0.1.22.10.0.0.2.55246 = 1015
ltcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 1056
ltcpConnEntry.2.10.0.0.1.22.10.0.0.2.55246 = 1392
local.9.2.1.18.2 = lab
*Jul 18 10:18:42.879: SNMP: Packet sent via UDP to 10.0.0.2

Device1#

```

例: SSH のデバッグの強化

次に、**debug ip ssh detail** コマンドの出力例を示します。出力には、SSH プロトコルとチャネル要求に関するデバッグ情報が含まれます。

```

Device# debug ip ssh detail
Cisco Confidential
00:04:22: SSH0: starting SSH control process
00:04:22: SSH0: sent protocol version id SSH-1.99-Cisco-1.25
00:04:22: SSH0: protocol version id is - SSH-1.99-Cisco-1.25
00:04:22: SSH2 0: SSH2_MSG_KEXINIT sent
00:04:22: SSH2 0: SSH2_MSG_KEXINIT received
00:04:22: SSH2:kex: client->server enc:aes128-cbc mac:hmac-sha1
00:04:22: SSH2:kex: server->client enc:aes128-cbc mac:hmac-sha1
00:04:22: SSH2 0: expecting SSH2_MSG_KEXDH_INIT
00:04:22: SSH2 0: SSH2_MSG_KEXDH_INIT received
00:04:22: SSH2: kex_derive_keys complete
00:04:22: SSH2 0: SSH2_MSG_NEWKEYS sent
00:04:22: SSH2 0: waiting for SSH2_MSG_NEWKEYS
00:04:22: SSH2 0: SSH2_MSG_NEWKEYS received
00:04:24: SSH2 0: authentication successful for lab
00:04:24: SSH2 0: channel open request
00:04:24: SSH2 0: pty-req request
00:04:24: SSH2 0: setting TTY - requested: height 24, width 80; set: height 24, width 80
00:04:24: SSH2 0: shell request
00:04:24: SSH2 0: shell message received
00:04:24: SSH2 0: starting shell for vty
00:04:38: SSH0: Session terminated normally

```

次に、**debug ip ssh packet** コマンドの出力例を示します。出力には、SSH パケットに関するデバッグ情報が含まれます。

```

Device# debug ip ssh packet

00:05:43: SSH2 0: send:packet of length 280 (length also includes padlen of 4)
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: input: total packet length of 280 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 24 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 4 bytes
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: input: total packet length of 144 bytes

```

```
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 16 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 6 bytes
00:05:43: SSH2 0: signature length 143
00:05:43: SSH2 0: send:packet of length 448 (length also includes padlen of 7)
00:05:43: SSH2 0: send:packet of length 16 (length also includes padlen of 10)
00:05:43: SSH2 0: newkeys: mode 1
00:05:43: SSH2 0: ssh_receive: 16 bytes received
00:05:43: SSH2 0: input: total packet length of 16 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 8 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 10 bytes
00:05:43: SSH2 0: newkeys: mode 0
00:05:43: SSH2 0: ssh_receive: 52 bytes received
00:05:43: SSH2 0: input: total packet length of 32 bytes
00:05:43: SSH2 0: partial packet length(block size)16 bytes,needed 16 bytes, maclen 20
00:05:43: SSH2 0: MAC compared for #3 :ok
```




CHAPTER 3

SSH 認証の X.509v3 証明書

- SSH 認証用の X.509v3 証明書の機能履歴, on page 35
- SSH 認証の X.509v3 証明書, on page 36
- SSH 認証用 X.509v3 証明書の機能, on page 36
- SSH 認証の X.509v3 証明書の前提条件, on page 37
- SSH 認証の X.509v3 証明書の制約事項, on page 37
- SSH 認証用の X.509v3 証明書の設定方法, on page 37
- ユーザー認証用のデジタル証明書を確認するための SSH サーバーの設定, on page 39
- トラストポイント認証の設定とデバイス証明書の作成, on page 41
- デジタル証明書を使用したサーバーユーザー認証の設定確認, on page 43
- SSH 認証用の X.509v3 証明書の設定例, on page 44

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情報とプラットフォームサポート情報を示します。

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	機能名と説明	サポートされるプラットフォーム
Cisco IOS XE 26.2.1ea	SSH 認証用 X.509v3 証明書: SSH 認証用 X.509v3 証明書の機能サポートが導入されました。	Cisco C9550 シリーズ スマートスイッチ

リリース	機能名と説明	サポートされるプラットフォーム
Cisco IOS XE 17.18.1	SSH 認証用 X.509v3 証明書: SSH 認証用 X.509v3 証明書により、SSH の証明書ベースのサーバーとユーザー認証が可能になり、PKI トラストポイントと X.509v3 デジタル証明書が ID 管理の強化に利用されます。	Cisco C9350 シリーズ スマートスイッチ Cisco C9610 シリーズ スマートスイッチ

SSH 認証の X.509v3 証明書

セキュアシェル（SSH）認証用の X.509v3 証明書機能は、サーバーと SSH サーバー側のユーザー認証の両方で X.509v3 デジタル証明書を使用できるようにします。このモジュールでは、これらの認証タイプの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 認証用 X.509v3 証明書の機能

デジタル証明書

X.509v3 形式（RFC5280）のデジタル証明書は、ID 管理を提供します。信頼できるルートおよび中間認証局（CA）からの署名のチェーンを介して、公開署名キーと署名者のアイデンティティ間のリンクが確立されます。公開キーインフラストラクチャ（PKI）のトラストポイントは、デジタル証明書の管理に役立ちます。トラストポイントには、CA、アイデンティティパラメータ、およびデジタル証明書に関する情報が含まれており、追跡とさまざまな証明書との関連付けができます。

サーバーとユーザー認証

デフォルトでは、証明書ベースの認証が SSH サーバーでサーバーおよびユーザーに対して有効になります。

- **サーバー認証:** Cisco IOS XE SSH サーバー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。このサーバー証明書は、`conf-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。
- **ユーザー認証:** SSH クライアントが確認のためにユーザーの証明書を SSH サーバーに送信します。SSH サーバーは、`conf-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 PKI トラストポイントを使用して、受信したユーザー証明書を確認します。

SSH 認証の X.509v3 証明書の前提条件

廃止された **ip ssh server authenticate user** コマンドは、**ip ssh server algorithm authentication** コマンドに置き換わります。古いコマンドを使用すると、次の警告メッセージが表示されます。

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

古いコマンドを無効にして、SSH サーバーが新しいコマンド **default ip ssh server authenticate user** を使用するようにします。

SSH 認証の X.509v3 証明書の制約事項

- この機能は、Cisco IOS XE セキュアシェル（SSH）サーバー側にのみ適用されます。
- SSH サーバーは、サーバーおよびユーザー認証について、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 証明書の設定方法

サーバー認証にデジタル証明書を使用するように SSH サーバーを設定するには、次の手順を実行します。

Procedure

Step 1 enable

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2 configure terminal

Example:

```
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。

Step 3 ip ssh server algorithm hostkey {x509v3-ssh-rsa [ssh-rsa] | ssh-rsa [x509v3-ssh-rsa]}

Example:

```
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、設定済みのアルゴリズムのみです。

Note

IOS SSH サーバーには、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。

- ssh-rsa: 公開キーベース認証
- x509v3-ssh-rsa: 証明書ベース認証

Step 4 ip ssh server certificate profile

Example:

```
Device(config)# ip ssh server certificate profile
```

サーバー証明書プロファイルおよびユーザー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。

Step 5 server

Example:

```
Device(conf-ssh-server-cert-profile)# server
```

サーバー証明書プロファイルを設定し、SSH サーバー証明書プロファイルのユーザー コンフィギュレーション モードを開始します。

Step 6 trustpoint sign PKI-trustpoint- name

Example:

```
Device(conf-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。

Step 7 ocsp-response include

Example:

```
Device(conf-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。

Note

デフォルトではこのコマンドの no 形式が設定されており、OCSP 応答はサーバー証明書と一緒に送信されません。

Step 8 end

Example:

```
Device(conf-ssh-server-cert-profile-server)# end
```

SSH サーバー証明書プロファイルのサーバー 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。

ユーザー認証用のデジタル証明書を確認するための SSH サーバーの設定

ユーザー認証にデジタル証明書を使用するように SSH サーバーを設定するには、次の手順を実行します。

Procedure

Step 1 enable

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2 configure terminal

Example:

```
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。

Step 3 ip ssh server algorithm authentication {publickey | keyboard | password}

Example:

```
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、設定済みのアルゴリズムのみです。

Note

- SSH サーバーには、1 つ以上の設定済みユーザー認証アルゴリズムが必要です。
- ユーザー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。
- **ip ssh server algorithm authentication** コマンドは **ip ssh server authenticate user** コマンドの代わりに使用します。

Step 4 ip ssh server algorithm publickey {x509v3-ssh-rsa [ssh-rsa] | ssh-rsa [x509v3-ssh-rsa]}

Example:

```
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。

Note

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。

- ssh-rsa: 公開キーベース認証

- x509v3-ssh-rsa: 証明書ベース認証

Step 5 ip ssh server certificate profile

Example:

```
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。

Step 6 user

Example:

```
Device(conf-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。

Step 7 trustpoint verify PKI-trustpoint-name

Example:

```
Device(conf-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。

Note

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。

Step 8 ocsp-response required

Example:

```
Device(conf-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。

Note

デフォルトではこのコマンドの no 形式が設定されており、ユーザー証明書は OCSP 応答なしで受け入れられます。

Step 9 end

Example:

```
Device(conf-ssh-server-cert-profile-user)# end
```

SSH サーバ証明書プロファイルのユーザー 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。

トラストポイント認証の設定とデバイス証明書の作成

トラストポイント認証を設定してデバイス証明書を作成するには、次の手順を実行します。

**Note**

新しく設定した PKI 証明書には、新しい RSA キーペア名を使用することをお勧めします。既存の RSA キーペア名（古い証明書に関連付けられている）を新しい PKI 証明書に再利用する場合は、次のいずれかを実行します。

- 既存の RSA キーペア名を使用して新しい RSA キーペアを再生成するのではなく、既存の RSA キーペア名を再利用します。既存の RSA キーペア名を使用して新しい RSA キーペアを再生成すると、既存の RSA キーペアに関連付けられているすべての証明書が無効になります。
- まず古い PKI 証明書設定を手動で削除してから、新しい PKI 証明書に既存の RSA キーペア名を再利用します。

Procedure

Step 1**enable****Example:**

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2**configure terminal****Example:**

```
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。

Step 3**crypto pki trustpoint name****Example:**

```
Device(config)# crypto pki trustpoint trust1
```

トラストポイントおよび設定された名前を宣言して、CA トラストポイント コンフィギュレーション モードを開始します。

Step 4**enrollment url url****Example:**

```
Device(ca-trustpoint)# enrollment url http://10.1.1.10:80
```

デバイスが証明書要求を送信する CA の URL を指定します。

Step 5**revocation-check none**

Example:

```
Device(ca-trustpoint)# revocation-check none
```

証明書の確認が無視されることを指定します。

Step 6

rsakeypair *key-label* [*keysize* [*encryption-key-size*]]

Example:

```
Device(ca-trustpoint)# rsakeypair trust1 2048
```

（任意）証明書に関連付けるキーペアを指定します。key-label 引数付きのキーペアがまだ存在しない、あるいは **auto-enroll regenerate** コマンドが発行された場合、登録時に key-label 引数付きのキーペアが生成されます。

鍵を生成するための key-size 引数を指定し、encryption-keysize 引数を指定して、個別の暗号化、署名鍵、および証明書を要求します。key-size 引数の範囲は 512 ～ 4096 です。key-size と encryption-key-size は同じサイズでなければなりません。2048 未満の長さを指定することは推奨されません。

Note

このコマンドがイネーブルでない場合に、FQDN キーペアが使用されます。

Step 7

exit

Example:

```
Device(ca-trustpoint)# exit
```

CA トラストポイントコンフィギュレーションモードを終了し、グローバルコンフィギュレーションモードに戻ります。

Step 8

crypto pki authenticate *name*

Example:

```
Device(config)# crypto pki authenticate trust1
```

CA 証明書を取得して、認証します。証明書フィンガープリントをチェックするよう求められた場合、証明書フィンガープリントをチェックします。

Note

CA 証明書がコンフィギュレーションにすでにロードされている場合、このコマンドはオプションです。

Step 9

crypto pki enroll *name*

Example:

```
Device(config)# crypto pki enroll trust1
```

証明書要求が証明書サーバーに送信され、サーバーが ID またはデバイス証明書を発行します。証明書要求にルータの FQDN および IP アドレスを含めるかどうかなどの登録情報を求められます。

Step 10

show crypto pki certificates

Example:

```
Device# show crypto pki certificates verbose trust1
```

(任意) ロールオーバー証明書などの、証明書に関する情報を表示します。

What to do next

他の登録オプションを使用して証明書をインストールする方法の詳細については、「[PKI 内での RSA キーの展開](#)」を参照してください。

デジタル証明書を使用したサーバーユーザー認証の設定確認

デジタル証明書を使用したサーバーおよびユーザー認証の設定を確認するには、次の手順を実行します。

Procedure

Step 1 enable

Example:

```
Device> enable
```

特権 EXEC モードを有効にします。プロンプトが表示されたらパスワードを入力します。

Step 2 show ip ssh

Example:

```
Device# show ip ssh
SSH Enabled - version 2.0
Authentication
methods:publickey,keyboardinteractive,
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
```

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。

SSH 認証用の X.509v3 証明書の設定例

ここでは、デジタル証明書を使用したユーザーおよびサーバー認証の例を示します。

この例では、サーバー認証用のデジタル証明書を使用するための SSH サーバーの設定方法を示します。

```
Device> enable
Device# configure terminal
Device(config)# ip ssh server algorithm hostkey x509v3-ssh-rsa
Device(config)# ip ssh server certificate profile
Device(conf-ssh-server-cert-profile)# server
Device(conf-ssh-server-cert-profile-server)# trustpoint sign trust1
Device(conf-ssh-server-cert-profile-server)# end
```

この例では、ユーザー認証用のユーザーのデジタル証明書を確認するための SSH サーバーの設定方法を示します。

```
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(conf-ssh-server-cert-profile)# user
Device(conf-ssh-server-cert-profile-user)# trustpoint verify trust2
Device(conf-ssh-server-cert-profile-user)# end
```

翻訳について

このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。