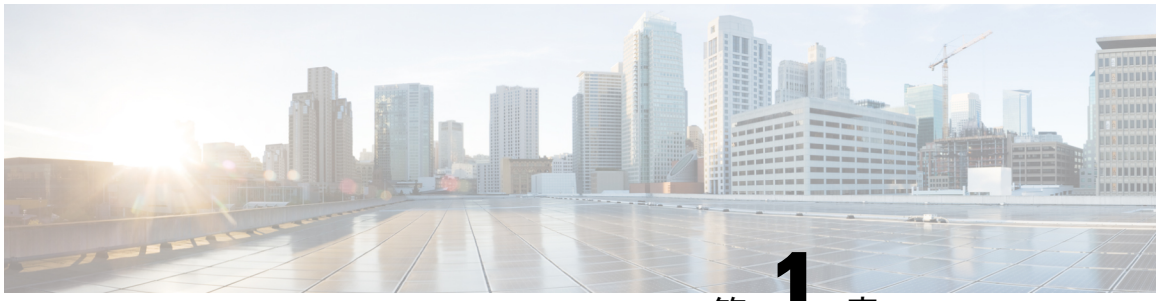
Cisco IOS XE Gibraltar 16.10.x セキュア シェル コンフィギュレーション ガイド

シスコシステムズ合同会社
〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー
http://www.cisco.com/jp
お問い合わせ先：シスコ コンタクトセンター
0120-092-255 （フリーコール、携帯・PHS含む）
電話受付時間：平日 10:00～12:00、13:00～17:00
http://www.cisco.com/jp/go/contactcenter/

目 次



第 1 章

最初にお読みください

Cisco IOS XE 16 に関する重要な情報

現行の Cisco IOS XE Release 3.7.0E（Catalyst スイッチング用）および Cisco IOS XE Release 3.17S（アクセスおよびエッジルーティング用）の 2 つのリリースは、単一バージョンのコンバージドリリース Cisco IOS XE 16 に進化（マージ）しました。これにより、スイッチングおよびルーティング ポートフォリオにおける広範なアクセス製品およびエッジ製品を 1 つのリリースでカバーします。

機能情報

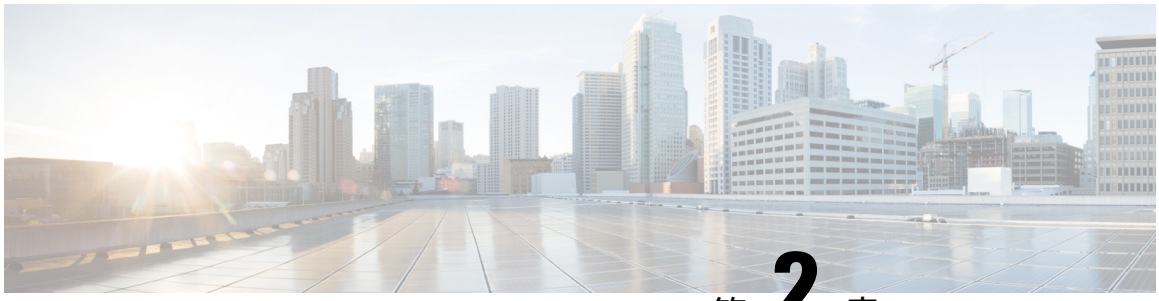
機能のサポート、プラットフォームのサポート、およびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco.com のアカウントは必要ありません。

参考資料

- 『Cisco IOS Command References, All Releases』

マニュアルの入手方法およびテクニカル サポート

- シスコからタイムリーな関連情報を受け取るには、Cisco Profile Manager でサインアップしてください。
- 重要な技術によりビジネスに必要な影響を与えるには、シスコ サービスにアクセスしてください。
- サービス リクエストを送信するには、シスコ サポートにアクセスしてください。
- 安全で検証済みのエンタープライズクラスのアプリケーション、製品、ソリューション、およびサービスを探して参照するには、Cisco Marketplace にアクセスしてください。
- 一般的なネットワーキング、トレーニング、認定関連の出版物を入手するには、Cisco Press にアクセスしてください。
- 特定の製品または製品ファミリの保証情報を探すには、Cisco Warranty Finder にアクセスしてください。



第 2 章

リバース SSH 拡張

セキュア シェル（SSH）のバージョン 1 と 2 に対してサポートされているリバース SSH 拡張機能は、SSH を有効にしなければならない端末または補助回線ごとに別々の回線を設定する必要がないようにリバース SSH を設定する代替手段を提供します。この機能は、ロータリー グループの制限も排除します。

機能情報の確認

ご使用のソフトウェア リリースでは、このモジュールで説明されるすべての機能がサポートされているとは限りません。最新の機能情報および警告については、「Bug Search Tool」およびご使用のプラットフォームおよびソフトウェア リリースのリリース ノートを参照してください。このモジュールで説明される機能に関する情報、および各機能がサポートされるリリースの一覧については、機能情報の表を参照してください。

プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

リバース SSH 拡張の前提条件

- SSH を有効にする必要があります。
- SSH クライアントとサーバで同じバージョンの SSH が動作している必要があります。

リバース SSH 拡張の制約事項

- リバース SSH の代替手段をコンソール アクセス用に設定する場合、**-l** キーワード、*userid* **:**{*number*} {*ip-address*} デリミタ、および引数が必須です。

リバース SSH 拡張に関する情報

リバース Telnet

リバース telnet を使用すると、特定のポート範囲に telnet を実行したり、端末または補助回線に接続することができます。リバース telnet は、他のシスコ デバイスのコンソールへの端末回線を複数内蔵したシスコ デバイスとの接続によく使用されていました。telnet を使用すると、特定の回線上のターミナル サーバに telnet することによって、どの場所からでも簡単にデバイス コンソールに到達できます。この telnet アプローチは、デバイスへのすべてのネットワーク接続が切断されている場合でも、そのデバイスの設定に使用できます。また、リバース telnet は、シスコ デバイスに接続されたモデムをダイヤルアウトに使用することもできます（通常は、ロータリー デバイスと一緒に使用します）。

リバース SSH

リバース telnet は SSH を使用して実現できます。リバース telnet と違って、SSH はセキュアな接続を提供します。リバース SSH 拡張機能は、SSH の設定を容易にします。この機能を使用すれば、SSH を有効にする端末または補助回線ごとに別々の回線を設定する必要がなくなります。以前のリバース SSH 設定方法では、アクセスできるポートの数が 100 に制限されていました。リバース SSH 拡張機能では、ポートの数に制限がありません。リバース SSH 設定の代替手段については、リバース SSH 拡張の設定方法 （4 ページ）を参照してください。

リバース SSH 拡張の設定方法

コンソール アクセス用のリバース SSH の設定

SSH サーバ上でリバース SSH コンソール アクセスを設定するには、次の手順を実行します。

手順の概要

1. **enable**
2. **configure terminal**
3. **line** *line-number* *ending-line-number*
4. **no exec**

5. **login authentication** *listname*
6. **transport input ssh**
7. **exit**
8. **exit**
9. **ssh -l** *userid* **:** {*number*} {*ip-address*}

手順の詳細

	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 モードをイネーブルにします。 • 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	**line** *line-number* *ending-line-number* 例： `Device# line 1 3`	設定用の回線を特定して、回線コンフィギュレーション モードに入ります。
ステップ 4	**no exec** 例： `Device(config-line)# no exec`	回線上の EXEC 処理を無効にします。
ステップ 5	**login authentication** *listname* 例： `Device(config-line)# login authentication default`	回線のログイン認証メカニズムを定義します。 （注） 認証方式はユーザ名とパスワードを使用する必要があります。
ステップ 6	**transport input ssh** 例： `Device(config-line)# transport input ssh`	デバイスの特定の回線への接続に使用されるプロトコルを定義します。 • リバース SSH 拡張機能の場合は、**ssh** キーワードを使用する必要があります。
ステップ 7	**exit** 例： `Device(config-line)# exit`	ライン コンフィギュレーション モードを終了します。
ステップ 8	**exit** 例：	グローバル コンフィギュレーション モードを終了します。

	コマンドまたはアクション	目的
	`Device(config)# exit`	
ステップ 9	**ssh -l** *userid* **:** {*number*} {*ip-address*} 例： `Device# ssh -l lab:1 router.example.com`	SSH サーバを実行しているリモート ネットワーキング デバイスにログインするときに使用されるユーザ ID を指定します。 • *userid*：ユーザ ID。 • **:**：ポート番号と端末 IP アドレスが userid 引数に続くことを示します。 • *number*：端末番号または補助回線番号。 • *ip-address*：ターミナル サーバの IP アドレス。 （注） リバース SSH の代替手段をモデム アクセス用に設定する場合は、*userid* 引数、**:rotary**{*number*}{*ip-address*} デリミタ、および引数が必須です。

モデム アクセス用のリバース SSH の設定

リバース SSH をモデム アクセス用に設定するには、後述の「手順の概要」で示す手順を実行します。

この設定では、リバース SSH がダイヤルアウト回線に使用されるモデム上で設定されます。ダイヤルアウト モデムのいずれかに到達するには、下のステップ 10 に示すように、任意の SSH クライアントを使用して SSH セッションを開始し、ロータリー デバイスから次に使用可能なモデムに到達します。

手順の概要

1. **enable**
2. **configure terminal**
3. **line** *line-number ending-line-number*
4. **no exec**
5. **login authentication** *listname*
6. **rotary** *group*
7. **transport input ssh**
8. **exit**
9. **exit**
10. **ssh -l** *userid* **:rotary** {*number*} {*ip-address*}

手順の詳細

	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 モードをイネーブルにします。 • 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	**line** *line-number ending-line-number* 例： `Device# line 1 200`	設定用の回線を特定して、回線コンフィギュレーション モードに入ります。
ステップ 4	**no exec** 例： `Device(config-line)# no exec`	回線上の EXEC 処理を無効にします。
ステップ 5	**login authentication** *listname* 例： `Device(config-line)# login authentication default`	回線のログイン認証メカニズムを定義します。 （注） 認証方式はユーザ名とパスワードを使用する必要があります。
ステップ 6	**rotary** *group* 例： `Device(config-line)# rotary 1`	1 つ以上の仮想端末回線または 1 つの補助ポート回線からなる回線グループを定義します。
ステップ 7	**transport input ssh** 例： `Device(config-line)# transport input ssh`	デバイスの特定の回線への接続に使用されるプロトコルを定義します。 • リバース SSH 拡張機能の場合は、**ssh** キーワードを使用する必要があります。
ステップ 8	**exit** 例： `Device(config-line)# exit`	ライン コンフィギュレーション モードを終了します。
ステップ 9	**exit** 例：	グローバル コンフィギュレーション モードを終了します。

	コマンドまたはアクション	目的
	`Device(config)# exit`	
ステップ 10	**ssh -l** *userid* **:rotary** {*number*} {*ip-address*} 例： `Device# ssh -l lab:rotary1 router.example.com`	SSH サーバを実行しているリモート ネットワーキングデバイスにログインするときに使用されるユーザ ID を指定します。 • *userid*：ユーザ ID。 • **:**：ポート番号と端末 IP アドレスが *userid* 引数に続くことを示します。 • *number*：端末番号または補助回線番号。 • *ip-address*：ターミナル サーバの IP アドレス。 （注） リバース SSH の代替手段をモデム アクセス用に設定する場合は、*userid* 引数、**:rotary**{*number*}{*ip-address*} デリミタ、および引数が必須です。

クライアント上でのリバース SSH のトラブルシューティング

クライアント（リモート デバイス）上でリバース SSH 設定の問題を解決するには、次の手順を実行します。

手順の概要

1. **enable**
2. **debug ip ssh client**

手順の詳細

	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 モードをイネーブルにします。 • パスワードを入力します（要求された場合）。
ステップ 2	**debug ip ssh client** 例： `Device# debug ip ssh client`	SSH クライアントに関するデバッギング メッセージを表示します。

サーバ上でのリバース SSH のトラブルシューティング

ターミナル サーバ上でリバース SSH 設定の問題を解決するには、次の手順を実行します。各ステップは、互いに独立しているため、任意の順序で設定できます。

手順の概要

1. **enable**
2. **debug ip ssh**
3. **show ssh**
4. **show line**

手順の詳細

	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 モードをイネーブルにします。 • パスワードを入力します（要求された場合）。
ステップ 2	**debug ip ssh** 例： `Device# debug ip ssh`	SSH サーバに関するデバッギング メッセージを表示します。
ステップ 3	**show ssh** 例： `Device# show ssh`	SSH サーバ接続のステータスを表示します。
ステップ 4	**show line** 例： `Device# show line`	端末回線のパラメータを表示します。

リバース SSH 拡張の設定例

リバース SSH コンソール アクセスの例

次の設定例は、リバース SSH が端末回線 1 ～ 3 のコンソール アクセス用に設定されていることを示しています。

ターミナル サーバの設定

```
line 1 3
   no exec
   login authentication default
   transport input ssh
```

クライアント設定

SSHクライアント上で設定された次のコマンドは、それぞれ、回線1、2、および3とのリバース SSH セッションを形成します。

```
ssh -l lab:1 router.example.com
ssh -l lab:2 router.example.com
ssh -l lab:3 router.example.com
```

リバース SSH モデム アクセスの例

次の設定例では、ダイヤルアウト回線の 1 ～ 200 がモデム アクセス用のロータリー グループ 1 にグループ分けされています。

```
line 1 200
   no exec
   login authentication default
   rotary 1
   transport input ssh
   exit
```

次のコマンドは、リバース SSH がロータリー グループの最初の空き回線に接続されることを表示します。

```
ssh -l lab:rotary1 router.example.com
```

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
Cisco IOS コマンド	『Cisco IOS Master Commands List, All Releases』
セキュア シェルの設定	『セキュア シェル コンフィギュレーション ガイド』
セキュリティ コマンド	『Cisco IOS Security Command Reference』

シスコのテクニカル サポート

説明	リンク
右の URL にアクセスして、シスコのテクニカル サポートを最大限に活用してください。これらのリソースは、ソフトウェアをインストールして設定したり、シスコの製品やテクノロジーに関する技術的問題を解決したりするために使用してください。この Web サイト上のツールにアクセスする際は、Cisco.com のログイン ID およびパスワードが必要です。	http://www.cisco.com/cisco/web/support/index.html

関連資料

関連項目	マニュアル タイトル
Cisco IOS コマンド	『Cisco IOS Master Commands List, All Releases』
セキュア シェルの設定	『セキュア シェル コンフィギュレーション ガイド』
セキュリティ コマンド	『Cisco IOS Security Command Reference』

標準

標準	タイトル
この機能がサポートする新しい規格または変更された規格はありません。	--

MIB

MIB	MIB のリンク
なし	選択したプラットフォーム、Cisco ソフトウェア リリース、およびフィーチャ セットの MIB を検索してダウンロードする場合は、次の URL にある Cisco MIB Locator を使用します。 http://www.cisco.com/go/mibs

RFC

RFC	タイトル
なし	--

シスコのテクニカル サポート

説明	リンク
右の URL にアクセスして、シスコのテクニカル サポートを最大限に活用してください。これらのリソースは、ソフトウェアをインストールして設定したり、シスコの製品やテクノロジーに関する技術的問題を解決したりするために使用してください。この Web サイト上のツールにアクセスする際は、Cisco.com のログイン ID およびパスワードが必要です。	http://www.cisco.com/cisco/web/support/index.html

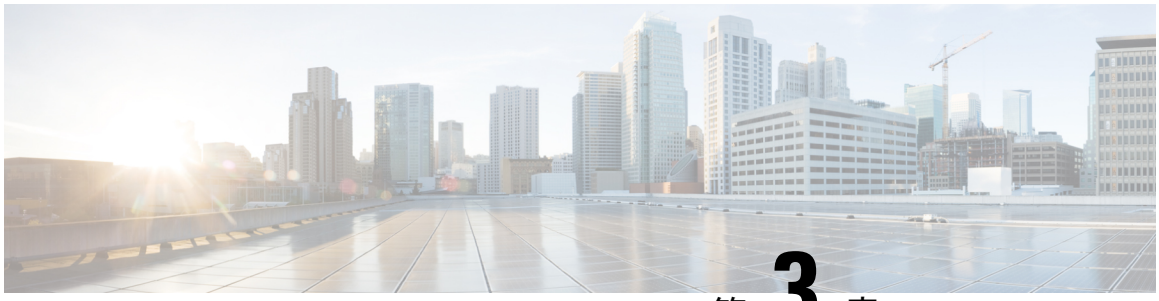
リバース SSH 拡張の機能情報

次の表に、このモジュールで説明した機能に関するリリース情報を示します。この表は、ソフトウェア リリース トレインで各機能のサポートが導入されたときのソフトウェア リリースだけを示しています。その機能は、特に断りがない限り、それ以降の一連のソフトウェア リリースでもサポートされます。

プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

表 *1*: リバース *SSH* 拡張の機能情報

機能名	リリース	機能情報
リバース SSH 拡張		セキュア シェル（SSH）のバージョン 1 と 2 に対してサポートされているリバース SSH 拡張機能は、SSH を有効にしなければならない端末または補助回線ごとに別々の回線を設定する必要がないようにリバース SSH を設定する代替手段を提供します。この機能は、ロータリー グループの制限も排除します。 次のコマンドが導入されました：**ssh**。



第 3 章

セキュア コピー

セキュア コピー（SCP）機能は、ルータ設定またはルータ イメージ ファイルをコピーするセキュアで認証された方法を提供します。SCP は、セキュア シェル（SSH）、アプリケーション、および Berkeley r ツールのセキュアな代替手段を提供するプロトコルに依存します。

セキュア コピーの前提条件

- SCP を有効にする前に、ルータ上で SSH、認証、および認可を正しく設定する必要があります。
- SCP は SSH を使用してセキュアな転送を実行するため、ルータには RSA キーのペアが必要です。

セキュアコピーのパフォーマンス向上に関する制限事項

- ウィンドウ サイズの増加は、主に SCP 操作に対してのみ使用する必要があります。
- プラットフォームのタイプによっては、ウィンドウ サイズが最大の場合に CPU 使用率が高くなることがあります。
- 万一に備えて、デフォルト サイズの 4 倍まで増やすことができます。

Secure Copy に関する情報

SCP の機能

SCP は一連の Berkeley の r-tools に基づいて設計されているため、その動作内容は、SCP が SSH のセキュリティに対応している点を除けば、Remote Copy Protocol（RCP）と類似しています。加えて、SCP は、ユーザが正しい権限レベルを持っていることをルータ上で判断できるように、認証、許可、アカウンティング（AAA）許可を設定する必要があります。

SCP を使用すると、適切な許可を得たユーザは、**copy** コマンドを使用して、Cisco IOS XE ファイル システム（IFS）内に存在する任意のファイルをルータとやり取りすることができます。許可された管理者はワークステーションからこの操作を実行することもできます。

SCP の設定方法

SCP の設定

Cisco ルータを有効にして、SCP サーバ側機能用に設定するには、次の手順を実行します。

手順の概要

1. **enable**
2. **configure terminal**
3. **aaa new-model**
4. **aaa authentication login** {**default** | *list-name*} *method1*[*method2...*]
5. **aaa authorization** {**network** | **exec** | **commands** *level* | **reverse-access** | **configuration**} {**default** | *list-name*} [*method1* [*method2...*]]
6. **username** *name* [**privilege** *level]*{ **password** *encryption-type encrypted-password*}
7. **ip scp server enable**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**enable** 例： `Router> enable`	特権 EXEC モードをイネーブルにします。 • 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： `Router# 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。

	コマンドまたはアクション	目的
ステップ 3	**aaa new-model** 例： `Router (config)# aaa new-model`	ログイン時の AAA 認証を設定します。
ステップ 4	**aaa authentication login** {**default** | *list-name*} *method1*[*method2...*] 例： `Router (config)# aaa authentication login default group tacacs+`	AAA アクセス コントロール システムをイネーブルにします。
ステップ 5	**aaa authorization** {**network** | **exec** | **commands** *level* | **reverse-access** | **configuration**} {**default** | *list-name*} [*method1* [*method2...*]] 例： `Router (config)# aaa authorization exec default group tacacs+`	ネットワークへのユーザ アクセスを制限するパラメータを設定します。 （注） **The exec** キーワードは、認可を実行してユーザが EXEC シェルの実行を許可されているかどうかを判断します。したがって、SCP を設定するときにこのキーワードを使用する必要があります。
ステップ 6	**username** *name* [**privilege** *level]*{ **password** *encryption-type encrypted-password*} 例： `Router (config)# username superuser privilege 2 password 0 superpassword`	ユーザ名をベースとした認証システムを構築します。 （注） TACACS+ や RADIUS などのネットワークベースの認証メカニズムが設定されている場合は、この手順を省略できます。
ステップ 7	**ip scp server enable** 例： `Router (config)# ip scp server enable`	SCP サーバ側機能を有効にします。

SCP の確認

SCP サーバ側機能を確認するには、次の手順を実行します。

手順の概要

1. **enable**
2. **show running-config**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**enable** 例： `Router> enable`	特権 EXEC モードをイネーブルにします。 • パスワードを入力します（要求された場合）。
ステップ 2	**show running-config** 例： `Router# show running-config`	SCP サーバ側機能を確認します。

SCP のトラブルシューティング

手順の概要

1. **enable**
2. **debug ip scp**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**enable** 例： `Router> enable`	特権 EXEC モードをイネーブルにします。 • パスワードを入力します（要求された場合）。
ステップ 2	**debug ip scp** 例： `Router# debug ip scp`	SCP 認証問題を解決します。

セキュア コピーの設定例

ローカル認証を使用した SCP サーバ側の設定例

次の例は、SCP のサーバ側機能の設定方法を示しています。この例では、ローカルに定義されたユーザ名とパスワードを使用します。

```
! AAA authentication and authorization must be configured properly for SCP to work.
aaa new-model
aaa authentication login default local
```

```
aaa authorization exec default local
username tiger privilege 15 password 0 lab
! SSH must be configured and functioning properly.
ip ssh time-out 120
ip ssh authentication-retries 3
ip scp server enable
```

ネットワークベース認証を使用した SCP サーバ側の設定例

次の例は、ネットワークベースの認証メカニズムを使用した SCP のサーバ側機能の設定方法を示しています。

```
! AAA authentication and authorization must be configured properly for SCP to work.
aaa new-model
aaa authentication login default group tacacs+
aaa authorization exec default group tacacs+
! SSH must be configured and functioning properly.
ip ssh time-out 120
ip ssh authentication-retries 3
ip scp server enable
```

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
Cisco IOS コマンド	『Cisco IOS Master Commands List, All Releases』
セキュリティ コマンド：コマンド構文の詳細、コマンド モード、コマンド履歴、デフォルト設定、使用上のガイドライン、および例	『*Cisco IOS Security Command Reference*』
セキュア シェル	セキュア シェルおよびセキュア シェル バージョン 2 サポート設定の機能モジュール。
認証と認可の設定	認証設定、認可設定、およびアカウンティング設定の機能モジュール。

標準

標準	タイトル
なし	--

MIB

MIB	MIB のリンク
なし	選択したプラットフォーム、Cisco ソフトウェア リリース、およびフィーチャ セットの MIB を検索してダウンロードする場合は、次の URL にある Cisco MIB Locator を使用します。 http://www.cisco.com/go/mibs

RFC

RFC	タイトル
この機能によりサポートされた新規 RFC または改訂 RFC はありません。またこの機能による既存 RFC のサポートに変更はありません。	--

テクニカル サポート

説明	リンク
右の URL にアクセスして、シスコのテクニカル サポートを最大限に活用してください。これらのリソースは、ソフトウェアをインストールして設定したり、シスコの製品やテクノロジーに関する技術的問題を解決したりするために使用してください。この Web サイト上のツールにアクセスする際は、Cisco.com のログイン ID およびパスワードが必要です。	http://www.cisco.com/cisco/web/support/index.html

セキュア コピーの機能情報

次の表に、このモジュールで説明した機能に関するリリース情報を示します。この表は、ソフトウェア リリース トレインで各機能のサポートが導入されたときのソフトウェア リリースだけを示しています。その機能は、特に断りがない限り、それ以降の一連のソフトウェア リリースでもサポートされます。

プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

表 *2:* セキュア コピーの機能情報

機能名	リリース	機能の設定情報
セキュア コピー	Cisco IOS XE Release 2.1	セキュア コピー（SCP）機能は、ルータ設定またはルータ イメージ ファイルをコピーするセキュアで認証された方法を提供します。SCPは、セキュア シェル（SSH）、アプリケーション、およびBerkeley r ツールのセキュアな代替手段を提供するプロトコルに依存します。 この機能は、Cisco IOS XE Release2.1 で、Cisco ASR 1000 シリーズアグリゲーション サービス ルータに導入されました。 次のコマンドが導入または変更されました：**debug ip scp**、**ip scp server enable**。

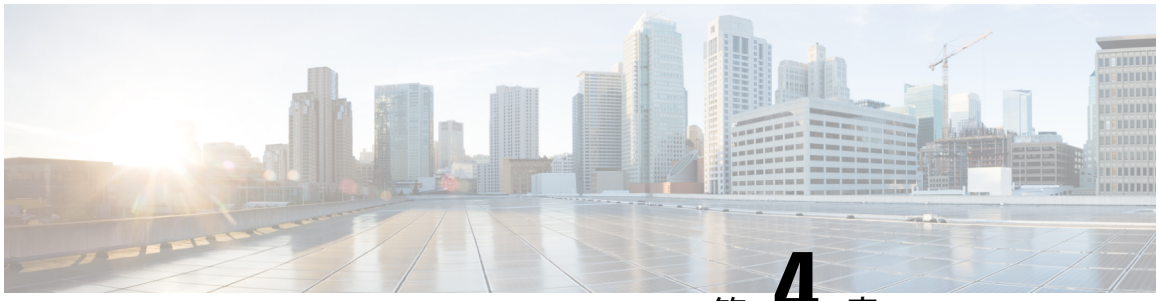
用語集

AAA ：認証、許可、およびアカウンティングセキュリティ サービスのフレームワークであり、ユーザの身元確認（認証）、リモート アクセス コントロール（許可）、課金、監査、およびレポートに使用するセキュリティ サーバ情報の収集と送信（アカウンティング）の方式を定めています。

rcp ：リモート コピーセキュリティをリモート シェル（Berkeley r ツール スイート）に依存している rcp は、ルータ イメージやスタートアップ コンフィギュレーションなどのファイルをルータとやり取りします。

SCP ：セキュア コピーセキュリティを SSH に依存している SCP サポートは、Cisco IOS XE ファイル システム内のあらゆるもののセキュアで認証されたコピーを可能にします。SCP は rcp から派生したものです。

SSH ：セキュア シェルBerkeley r ツールのセキュアな代替手段を提供するアプリケーションとプロトコル。プロトコルは標準の暗号メカニズムを使用してセッションの安全を確保します。アプリケーションは Berkeley の rexec および rsh ツールと同様に使用できます。SSH バージョン 1 は Cisco IOS XE ソフトウェアに実装されています。



第 4 章

セキュア シェル バージョン 2 サポート

セキュア シェル バージョン 2 サポート機能で、セキュア シェル（SSH）バージョン 2 を設定できます（SSH バージョン 1 サポートは、以前のシスコ ソフトウェア リリースに実装されていました）。SSH は、信頼性の高いトランスポート層の上部で実行され、強力な認証機能と暗号化機能を提供します。SSH では、信頼できる転送として定義されているのは TCP のみです。SSH で、ネットワーク上の他のコンピュータに安全にアクセスしたり、コマンドを安全に実行できます。SSH とともに提供されるセキュア コピー プロトコル（SCP）機能で、ファイルを安全に転送できます。

機能情報の確認

ご使用のソフトウェア リリースでは、このモジュールで説明されるすべての機能がサポートされているとは限りません。最新の機能情報および警告については、「Bug Search Tool」およびご使用のプラットフォームおよびソフトウェア リリースのリリース ノートを参照してください。このモジュールで説明される機能に関する情報、および各機能がサポートされるリリースの一覧については、機能情報の表を参照してください。

プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

セキュア シェル バージョン 2 サポートの前提条件

- SSH を設定する前に、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 サーバには、ご使用のリリースに応じた 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 デバイスを使用する必要があります。また、シスコ デバイスに接続する必要があります。
- SCP は、認証、認可、およびアカウンティング（AAA）によって正しく機能します。そのため、SSH サーバで Secure Copy Protocol が有効になるようにデバイスで AAA を設定する必要があります。

（注） SSH バージョン 2 サーバと SSH バージョン 2 クライアントは、ご使用のリリースに応じてシスコ ソフトウェアでサポートされます（SSH クライアントは SSH バージョン 1 プロトコルと SSH バージョン 2 プロトコルの両方を実行します。SSH クライアントは、ご使用のリリースに応じて k8 および k9 イメージの両方でサポートされます）。

ソフトウェア イメージのダウンロードに関する情報については、『*Cisco IOS Configuration Fundamentals* コンフィギュレーション ガイド』を参照してください。

セキュア シェル バージョン 2 サポートの制約事項

- セキュア シェル（SSH）サーバと SSH クライアントは、Triple Data Encryption Standard（3DES）ソフトウェア イメージでサポートされます。
- サポートされるアプリケーションは、実行シェル、remote 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 サーバ側の要件です。SSH クライアントとして動作するデバイスは、RSA キーを生成する必要がありません。
- 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。
- 次の機能はサポートされていません。
  - ポート フォワーディング。
  - Compression

セキュア シェル バージョン 2 サポートに関する情報

SSH バージョン 2

セキュア シェル バージョン 2 サポート機能で、SSH バージョン 2 を設定できます。

SSH バージョン 2 サーバの設定は、SSH バージョン 1 の設定と同様です。**ip ssh version** コマンドは、設定する SSH バージョンを定義します。このコマンドを設定しない場合、デフォルトで SSH は互換モードで実行されます。バージョン 1 とバージョン 2 両方の接続が利用できます。

（注） SSH バージョン 1 は、標準として定義されていないプロトコルです。未定義のプロトコル（バージョン 1）にデバイスがフォールバックし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 を指定する必要があります。

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。すでに、SSH は生成済みの最初の RSA キーにリンクされています（つまり、最初の RSA キー ペアが生成された時点で SSH はイネーブルになっています）。この動作は存在していますが、**ip ssh rsa keypair-name** コマンドを使用してこの動作を行わないようにすることができます。**ip ssh rsa keypair-name** コマンドをキー ペアの名前を指定して設定すると、SSH は、キー ペアが存在する場合に有効になるか、キー ペアを後で作成する場合は後から有効になります。このコマンドを使用して SSH をイネーブルにする場合、Cisco ソフトウェアの SSH バージョン 1 では必要な、ホスト名とドメイン名を設定を設定する必要はありません。

（注） 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、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 の機能拡張

SSH バージョン 2 の機能拡張には、Virtual Routing and Forwarding（VRF）-Aware SSH、SSH デバッグ機能拡張、および Diffie-Hellman（DH）グループ交換のサポートなどの追加機能がいくつか含まれています。

（注） VRF-Aware SSH 機能は、ご使用のリリースに応じてサポートされます。

Cisco SSH 実装では従来、768 ビット絶対値が使用されていましたが、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 サーバ上のモジュラス サイズを設定します。これに加え、**ssh** コマンドが拡張され、SSH クライアント側のクライアントの VRF インスタンス名を IP アドレスとともに使用して、正しいルーティング テーブルを検索し、接続を確立する機能に、VRF 認識が追加されました。

SSH debug コマンドが修正され、デバッグが拡張されました。**debug ip ssh** コマンドは、デバッグ プロセスを簡素化するために拡張されました。デバッグ プロセスを簡素化する前、このコマンドでは、明確に必要かどうかに関係なく SSH に関連するすべてのデバッグ メッセージが印刷されました。この動作は依然として存在しますが、 **debug ip ssh** コマンドをキーワードを指定して設定した場合、メッセージはキーワードで指定した情報に制限されます。

セキュア シェル バージョン 2 の RSA キーに関する機能拡張

Cisco SSH バージョン 2 は、キーボード インタラクティブ認証方式およびパスワード ベースの認証方式をサポートしています。RSA キーの SSH バージョン 2 拡張機能は、クライアントとサーバ向けの RSA ベースの公開キー認証もサポートしています。

ユーザ認証：RSA ベースのユーザ認証では、各ユーザに関連付けられている秘密キー/公開キーのペアを認証に使用します。ユーザは秘密キー/公開キーのペアをクライアントで生成し、公開キーを Cisco SSH サーバで設定して、認証を完了します。

クレデンシャルの確立を試行する SSH ユーザは、秘密キーを使用して暗号化された署名を提示します。署名とユーザの公開キーは、認証のために SSH サーバに送信されます。SSH サーバでは、ユーザから提示された公開キーに対してハッシュを計算します。ハッシュは、サーバに一致するエントリがあるかどうかを判断するために使用されます。一致が見つかった場合、RSA ベースのメッセージ検証が公開キーを使用して実行されます。その結果、暗号化されたシグニチャに基づいて、ユーザのアクセスは認証されるか拒否されます。

サーバ認証：SSH セッションの確立中に、Cisco SSH クライアントは、キー交換フェーズ中に使用できるサーバ ホスト キーを使用して、SSH サーバを認証します。SSH サーバ キーは、SSH サーバの識別に使用されます。これらのキーは SSH がイネーブルになるときに作成され、クライアント側で設定する必要があります。

サーバ認証の場合、Cisco SSH クライアントが各サーバにホスト キーを割り当てる必要があります。クライアントがサーバとの間で SSH セッションを確立しようとすると、クライアントはキー交換メッセージの一部として、サーバ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イネーブルの場合、そのサーバに対応するホスト キー エントリがあるかどうかがクライアントで確認されます。一致が見つかると、クライアントはサーバ ホスト キーを使用して署名の検証を試行します。サーバの認証に成功すると、セッションの確立処理は続行します。失敗すると、処理は終了し、「Server Authentication Failed」というメッセージが表示されます。

（注） 公開キーをサーバで格納する際、メモリを使用します。したがって、SSH サーバで設定できる公開キーの数は、1 ユーザに最大 2 つの公開キーを作成した場合 10 ユーザ分に限られます。

（注） シスコ サーバは RSA ベースのユーザ認証をサポートしていますが、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オープンな SSH クライアントからの要求を Cisco サーバが受信した場合、サーバは認証要求を受け入れます。

（注） サーバ認証の場合、サーバの 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。

SNMP トラップ生成

ご使用のリリースに応じて、簡易ネットワーク管理プロトコル（SNMP）トラップは、トラップが有効で SNMP デバッグがオンになっている場合、SSH セッションが終了した際に自動的に生成されます。SNMP トラップの有効化に関する情報については、『*SNMP Configuration Guide*』の「Configuring SNMP Support」モジュールを参照してください。

（注） **snmp-server host** コマンドを設定する場合、IP アドレスは、SSH（telnet）クライアントがあり、SSH サーバへの IP 接続が可能な PC のアドレスにする必要があります。

また、**debug snmp packet** コマンドを使用して SNMP デバッグを有効にし、トラップを表示する必要があります。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、異なる種類の認証メカニズムを実装するために使用できる方式です。基本的に、現在サポートされている、ユーザの入力のみが必要な認証方式はすべて、この機能で実行することができます。この機能は自動的にイネーブルになります。

次の方式がサポートされています。

- Password
- サーバが送信するチャレンジに応答する番号またはストリングを印刷する SecurID およびハードウェア トークン
- 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）

自動的に有効化された SSH キーボード インタラクティブ認証機能のさまざまなシナリオの例については、「例：SSH キーボード インタラクティブ認証 （42 ページ）」を参照してください。

セキュア シェル バージョン 2 サポートの設定方法

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

手順の概要

1. **enable**
2. **configure terminal**
3. **hostname** *name*
4. **ip domain-name** *name*
5. **crypto key generate rsa**
6. **ip ssh** [**time-out** *seconds* | **authentication-retries** *integer*]
7. **ip ssh version** [**1** | **2**]
8. **exit**

手順の詳細

	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 モードをイネーブルにします。 • 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	**hostname** *name* 例： `Device(config)# hostname cisco7200`	デバイスのホスト名を設定します。
ステップ 4	**ip domain-name** *name* 例： `cisco7200(config)# ip domain-name example.com`	デバイスのドメイン名を設定します。

	コマンドまたはアクション	目的
ステップ 5	**crypto key generate rsa** 例： `cisco7200(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。
ステップ 6	**ip ssh** [**time-out** *seconds* | **authentication-retries** *integer*] 例： `cisco7200(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。
ステップ 7	**ip ssh version** [**1** | **2**] 例： `cisco7200(config)# ip ssh version 1`	（任意）デバイスで実行する SSH のバージョンを指定します。
ステップ 8	**exit** 例： `cisco7200(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

ステップ 1 **enable**

例：

```
Device> enable
```

特権 EXEC モードをイネーブルにします。

- 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 rsa keypair-name** *keypair-name*

例：

```
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。

（注） シスコ デバイスには複数の RSA キー ペアを設定できます。

ステップ 4 **crypto key generate rsa usage-keys label** *key-label* **modulus** *modulus-size*

例：

```
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。

（注） 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。

ステップ 5 **ip ssh** [**time-out** *seconds* | **authentication-retries** *integer*]

例：

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

デバイス上で SSH 制御変数を設定します。

ステップ 6 **ip ssh version 2**

例：

```
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。

ステップ 7 **exit**

例：

```
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 モードを開始します。

RSA ベースのユーザ認証を実行するための Cisco SSH サーバの設定

ステップ 1 **enable**

例：

```
Device> enable
```

特権 EXEC モードをイネーブルにします。

- 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 **hostname** *name*

例：

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

ホスト名を指定します。

ステップ 4 **ip domain-name** *name*

例：

```
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。

ステップ 5 **crypto key generate rsa**

例：

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

RSA キー ペアを生成します。

ステップ 6 **ip ssh pubkey-chain**

例：

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

SSH サーバ上のユーザおよびサーバ認証用に SSH-RSA 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。

- サーバに保存されている RSA 公開キーが、クライアントに保存されている公開キーと秘密キーのペアを使用して検証されると、ユーザ認証は成功です。

ステップ 7 **username** *username*

例：

```
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。

ステップ 8 **key-string**

例：

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

リモート ピアの RSA 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。

（注） 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。

ステップ 9 **key-hash** *key-type key-name*

例：

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

（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 にする必要があります。
- **key-string** コマンドが設定されている場合に限りこの手順は任意です。
- **key-string** コマンドと **key-hash** コマンドのいずれかを設定する必要があります。

（注） 公開キー ストリング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。

ステップ 10 **end**

例：

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

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、特権 EXEC モードに戻ります。

- 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 ベースのサーバ認証を実行するための Cisco IOS SSH サーバの設定

ステップ 1 **enable**

例：

```
Device> enable
```

特権 EXEC モードをイネーブルにします。

- 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 **hostname** *name*

例：

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

ホスト名を指定します。

ステップ 4 **ip domain-name** *name*

例：

```
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。

ステップ 5 **crypto key generate rsa**

例：

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

RSA キー ペアを生成します。

ステップ 6 **ip ssh pubkey-chain**

例：

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

SSH サーバ上のユーザおよびサーバ認証用に SSH-RSA 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。

ステップ 7 **server** *server-name*

例：

```
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。

ステップ 8 **key-string**

例：

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

リモート ピアの RSA 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。

（注） 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。

ステップ 9 **exit**

例：

```
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。

ステップ 10 **key-hash** *key-type key-name*

例：

```
host1(conf-ssh-pubkey-server)# key-hash ssh-rsa key1
```

（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 にする必要があります。
- **key-string** コマンドが設定されている場合に限りこの手順は任意です。
- **key-string** コマンドと **key-hash** コマンドのいずれかを設定する必要があります。

（注） 公開キー ストリング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。

ステップ 11 **end**

例：

```
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。

ステップ 12 **configure terminal**

例：

```
host1# 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 13 **ip ssh stricthostkeycheck**

例：

```
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。

リモート デバイスとの暗号化セッションの開始

（注） 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セキュア シェル（SSH）サーバをサポートしている必要があります。また、デバイスを有効にする必要はありません。SSH はディセーブル モードで実行できます。

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**]

例：

```
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。

トラブルシューティングのヒント

ip ssh version コマンドは、SSH の設定のトラブルシューティングに使用できます。バージョンを変更することによって、問題がある SSH バージョンを特定できます。

SSH サーバでのセキュア コピー プロトコルの有効化

（注） 次のタスクでは、SCP のサーバ側機能を設定します。このタスクは、デバイスでリモートのワークステーションからファイルを安全にコピーできる一般的な設定を示しています。

ステップ 1 **enable**

例：

```
Device> enable
```

特権 EXEC モードをイネーブルにします。

- 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 **aaa new-model**

例：

```
Device(config)# aaa new-model
```

AAA アクセス コントロール モデルをイネーブルにします。

ステップ 4 **aaa authentication login default local**

例：

```
Device(config)# aaa authentication login default local
```

認証時にローカルのユーザ名データベースを使用するように、ログイン時の AAA 認証を設定します。

ステップ 5 **aaa authorization exec defaultlocal**

例：

```
Device(config)# aaa authorization exec default local
```

ユーザ アクセスを制限するパラメータをネットワークに設定します。認証を実行し、ユーザ ID で EXEC シェルの実行を許可するかどうかを定義します。その後、システムで認証にローカル データベースを使用する必要があることを指定します。

ステップ 6 **username***name* **privilege** *privilege-level* **password** *password*

例：

```
Device(config)# username samplename privilege 15 password password1
```

ユーザ名ベースの認証システムを確立し、ユーザ名、権限レベル、および非暗号化パスワードを指定します。

（注） *privilege-level* 引数の最小値は 15 です。権限レベルが 15 未満の場合、接続が切断されます。

ステップ 7 **ip ssh time-out***seconds*

例：

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

デバイスが SSH クライアントの応答を待つ時間間隔を、秒単位で設定します。

ステップ 8 **ip ssh authentication-retries** 整数

例：

```
Device(config)# ip ssh authentication-retries 3
```

インターフェイスのリセット後、認証を試行する回数を設定します。

ステップ 9 **ip scpserverenable**

例：

```
Device(config)# ip scp server enable
```

デバイスで、リモート ワークステーションから安全にファイルをコピーできるようにします。

ステップ 10 **exit**

例：

```
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。

ステップ 11 **debug ip scp**

例：

```
Device# debug ip scp
```

（任意）SCP 認証の問題に関する診断情報を提供します。

セキュア シェル接続のステータスの確認

ステップ 1 **enable**

例：

```
Device> enable
```

特権 EXEC モードをイネーブルにします。

- パスワードを入力します（要求された場合）。

ステップ 2 **show ssh**

例：

```
Device# show ssh
```

SSH サーバ接続のステータスを表示します。

ステップ 3 **exit**

例：

```
Device# exit
```

特権 EXEC モードを終了し、ユーザ 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 2 およびバージョン 1 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 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 Encryption      State                   Username
 0              1.5     3DES            Session started         lab
%No SSHv2 server connections running.
---------------------------------------------------------------------
```

セキュア シェル ステータスの確認

ステップ 1 **enable**

例：

```
Device> enable
```

特権 EXEC モードをイネーブルにします。

- パスワードを入力します（要求された場合）。

ステップ 2　show ip ssh

例：

```
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。

ステップ 3　exit

例：

```
Device# exit
```

特権 EXEC モードを終了し、ユーザ 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

ステップ 1 **enable**

例：

```
Device> enable
```

特権 EXEC モードをイネーブルにします。

- パスワードを入力します（要求された場合）。

ステップ 2 **debug ip ssh**

例：

```
Device# debug ip ssh
```

SSH のデバッグを有効にします。

ステップ 3 **debug snmp packet**

例：

```
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。

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 サポートの設定例

例：セキュア シェル バージョン 1 の設定

```
Device# configure terminal
Device(config)# ip ssh version 1ip ssh version 2
```

例：セキュア シェル バージョン 2 の設定

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

例：セキュア シェル バージョン 1 および 2 の設定

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

例：リモート デバイスでの暗号化セッションの開始

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

例：サーバサイド SCP の設定

次の例では、SCP のサーバ側機能の設定方法を示します。この例では、デバイスでの AAA 認証および認可も設定しています。この例では、ローカルに定義されたユーザ名とパスワードを使用します。

```
Device# configure terminal
Device(config)# aaa new-model
Device(config)# aaa authentication login default local
Device(config)# aaa authorization exec default local
Device(config)# username samplename privilege 15 password password1
Device(config)# ip ssh time-out 120
```

```
Device(config)# ip ssh authentication-retries 3
Device(config)# ip scp server enable
```

例：SNMP トラップの設定

次の例では、設定済みの SNMP トラップを示します。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 a、b、c、d は SSH クライアントの IP アドレスです。SNMP トラップ デバッグ出力の例については、「例：SNMP のデバッグ （44 ページ）」のセクションを参照してください。

```
snmp-server
snmp-server host a.b.c.d public tty
```

例：SSH キーボード インタラクティブ認証

例：クライアント側のデバッグの有効化

次の例では、クライアント側のデバッグがオンになっており、プロンプトの最大数が 6（SSH キーボード インタラクティブ認証方式のために 3 つ、パスワード認証方式のために 3 つ）になっています。

```
Password:
Password:
Password:
Password:
Password:
Password: cisco123
Last login: Tue Dec 6 13:15:21 2005 from 10.76.248.213
user1@courier:~> exit
logout
[Connection to 10.76.248.200 closed by foreign host]
Device1# debug ip ssh client

SSH Client debugging is on

Device1# ssh -l lab 10.1.1.3

Password:
*Nov 17 12:50:53.199: SSH0: sent protocol version id SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.199: SSH CLIENT0: protocol version id is - SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.199: SSH CLIENT0: sent protocol version id SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.199: SSH CLIENT0: protocol version exchange successful
*Nov 17 12:50:53.203: SSH0: protocol version id is - SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.335: SSH CLIENT0: key exchange successful and encryption on
*Nov 17 12:50:53.335: SSH2 CLIENT 0: using method keyboard-interactive
Password:
Password:
Password:
*Nov 17 12:51:01.887: SSH2 CLIENT 0: using method password authentication
Password:
Password: lab
Device2>

*Nov 17 12:51:11.407: SSH2 CLIENT 0: SSH2_MSG_USERAUTH_SUCCESS message received
```

```
*Nov 17 12:51:11.407: SSH CLIENT0: user authenticated
*Nov 17 12:51:11.407: SSH2 CLIENT 0: pty-req request sent
*Nov 17 12:51:11.411: SSH2 CLIENT 0: shell request sent
*Nov 17 12:51:11.411: SSH CLIENT0: session open
```

例：ブランク パスワードの変更による ChPass の有効化

次の例では、ChPass 機能が有効になっており、SSH キーボード インタラクティブ認証方式を使用してブランク パスワードが変更されています。TACACS+ アクセス コントロール サーバ（ACS）は、バックエンド AAA サーバとして使用されています。

```
Device1# ssh -l cisco 10.1.1.3

Password:
Old Password: cisco
New Password: cisco123
Re-enter New password: cisco123

Device2> exit

[Connection to 10.1.1.3 closed by foreign host]
```

例：ChPass の有効化および初回ログインでのパスワード変更

次の例では、ChPass 機能が有効になっており、TACACS+ ACS はバックエンド サーバとして使用されています。パスワードは、SSH キーボード インタラクティブ 認証方式を使用して最初のログインで変更されています。

```
Device1# ssh -l cisco 10.1.1.3

Password: cisco
Your password has expired.
Enter a new one now.
New Password: cisco123
Re-enter New password: cisco123

Device2> exit

[Connection to 10.1.1.3 closed by foreign host]

Device1# ssh -l cisco 10.1.1.3

Password:cisco1
Your password has expired.
Enter a new one now.
New Password: cisco
Re-enter New password: cisco12
The New and Re-entered passwords have to be the same.
Try again.
New Password: cisco
Re-enter New password: cisco

Device2>
```

例：ChPass の有効化および 3 回ログインした後のパスワードの失効

次の例では、ChPass 機能が有効になっており、TACACS+ ACS はバックエンド AAA サーバとして使用されています。パスワードは、SSH キーボード インタラクティブ認証方式を使用して 3 回ログインした後に期限切れになります。

```
Device# ssh -l cisco. 10.1.1.3

Password: cisco

Device2> exit

[Connection to 10.1.1.3 closed by foreign host]

Device1# ssh -l cisco 10.1.1.3

Password: cisco

Device2> exit

Device1# ssh -l cisco 10.1.1.3

Password: cisco

Device2> exit

[Connection to 10.1.1.3 closed by foreign host]

Device1# ssh -l cisco 10.1.1.3

Password: cisco
Your password has expired.
Enter a new one now.
New Password: cisco123
Re-enter New password: cisco123

Device2>
```

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]
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

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
```

セキュア シェル バージョン 2 サポートの追加情報

関連資料

関連項目	マニュアル タイトル
Cisco IOS コマンド	『Cisco IOS Master Command List, All Releases』
AAA ホスト名およびホスト ドメインの設定タスク セキュア シェルの設定タスク	『*Security Configuration Guide*：*Securing User Services*』
ソフトウェア イメージのダウンロード 設定の基礎	『*Cisco IOS Configuration Fundamentals Configuration Guide*』
IPsec の設定作業	『*Security Configuration Guide*：*Secure Connectivity*』
SNMP トラップの設定タスク	『*SNMP Configuration Guide*』

標準

標準	タイトル
IETF Secure Shell Version 2 Draft 規格	Internet Engineering Task Force の Web サイト

シスコのテクニカル サポート

説明	リンク
右の URL にアクセスして、シスコのテクニカル サポートを最大限に活用してください。これらのリソースは、ソフトウェアをインストールして設定したり、シスコの製品やテクノロジーに関する技術的問題を解決したりするために使用してください。この Web サイト上のツールにアクセスする際は、Cisco.com のログイン ID およびパスワードが必要です。	http://www.cisco.com/cisco/web/support/index.html

セキュア シェル バージョン 2 サポートの機能情報

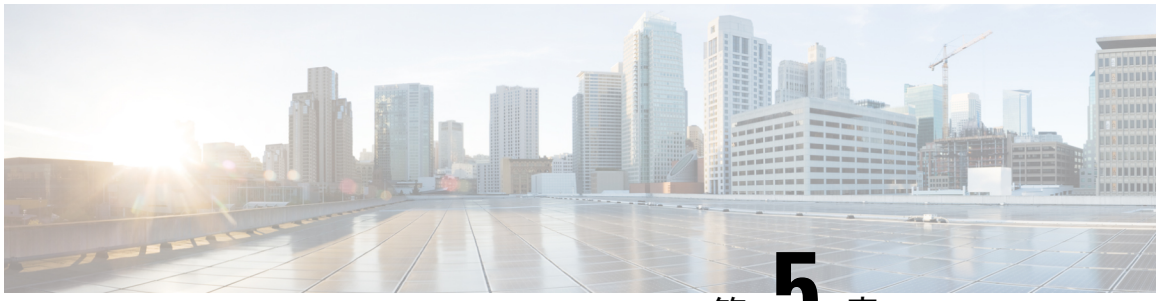
次の表に、このモジュールで説明した機能に関するリリース情報を示します。この表は、ソフトウェア リリース トレインで各機能のサポートが導入されたときのソフトウェア リリースだけを示しています。その機能は、特に断りがない限り、それ以降の一連のソフトウェア リリースでもサポートされます。

プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

表 *3:* セキュア シェル バージョン *2* サポートの機能情報

機能名	リリース	機能情報
セキュア シェル バージョン 2 サポート		セキュア シェル バージョン 2 サポート機能を使用して、セキュア シェル（SSH）バージョン 2 を設定できます（SSH バージョン 1 のサポートは、以前の Cisco IOS ソフトウェア リリースで実装されていました）。SSH は、信頼性の高いトランスポート層の上部で実行され、強力な認証機能と暗号化機能を提供します。SSH バージョン 2 は、AES カウンタベース暗号化モードもサポートします。 次のコマンドが導入または変更されました：**debug ip ssh**、**ip ssh min dh size**、**ip ssh rsa keypair-name**、**ip ssh version**、**ssh**。
セキュア シェル バージョン 2 クライアントおよびサーバ サポート		Cisco IOS イメージが、SSH セッション終了時に SNMP トラップを自動的に生成するよう更新されました。

機能名	リリース	機能情報
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、異なる種類の認証メカニズムを実装するために使用できる方式です。基本的に、現在サポートされている、ユーザの入力のみが必要な認証方式はすべて、この機能で実行することができます。
セキュアシェルバージョン 2 の機能拡張		セキュア シェル バージョン 2 の機能拡張には、VRF aware SSH、SSH デバッグ機能拡張、および DH グループ 14 および 16 交換のサポートなどの、追加機能がいくつか含まれています。 次のコマンドが導入または変更されました：**debug ip ssh**、**ip ssh dh min size**。
セキュアシェルバージョン 2 の RSA キーに関する機能拡張		RSA キーのセキュア シェル バージョン 2 機能拡張には、SSH 向け RSA キー ベースのユーザ認証や、SSH サーバ ホスト キーの保存や検証のサポートなどの、追加機能がいくつか含まれています。 次のコマンドが導入または変更されました：**ip ssh pubkey-chain**、**ip ssh stricthostkeycheck**。



第 5 章

セキュア シェル：ユーザ認証方式の設定

セキュア シェル：ユーザ認証方式の設定機能によって、セキュア シェル（SSH）サーバで使用可能なユーザ認証方式を設定できます。

機能情報の確認

ご使用のソフトウェア リリースでは、このモジュールで説明されるすべての機能がサポートされているとは限りません。最新の機能情報および警告については、「Bug Search Tool」およびご使用のプラットフォームおよびソフトウェア リリースのリリース ノートを参照してください。このモジュールで説明される機能に関する情報、および各機能がサポートされるリリースの一覧については、機能情報の表を参照してください。

プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

セキュア シェルの制約事項：ユーザ認証方式の設定

セキュア シェル（SSH）サーバと SSH クライアントは、データ暗号化ソフトウェア（DES）（56 ビット）および 3DES（168 ビット）イメージでのみサポートされます。

セキュア シェルに関する情報：ユーザ認証方式の設定

セキュア シェル ユーザ認証の概要

セキュア シェル（SSH）を使用することによって、SSH クライアントはシスコデバイス（Cisco IOS SSH サーバ）に対してセキュアで暗号化された接続を確立できます。SSH クライアントは SSH プロトコルを使用して、デバイス認証と暗号化を実行します。

SSH サーバは、3 種類のユーザ認証方式をサポートし、これらの認証方式を事前に定義された次の順序で SSH クライアントに送信します。

- 公開キー認証方式
- キーボード インタラクティブ認証方式
- パスワード認証方式

デフォルトでは、すべてのユーザ認証方式が有効になっています。無効な方式が SSH ユーザ認証プロトコルでネゴシエートされないように特定のユーザ認証を無効にするには、**no ip ssh server authenticate user** {**publickey** | **keyboard** | **pasword**} コマンドを使用します。この機能によって、SSH サーバは、事前に定義された順序とは異なる順序で希望のユーザ認証方式を指定できます。**ip ssh server authenticate user** {**publickey** | **keyboard** | **pasword**} コマンドを使用すると、無効になっているユーザ認証方式を有効にできます。

RFC 4252（セキュア シェル（SSH）認証プロトコル）のとおり、公開キー認証方式は必須です。この機能によって、SSH サーバで RFC の動作をオーバーライドして、公開キー認証を含む任意の SSH ユーザ認証方式を無効にすることができます。

たとえば、SSH サーバでパスワード認証方式を希望する場合、SSH サーバで公開キー認証方式とキーボード インタラクティブ認証方式を無効にすることができます。

セキュアシェルの設定方法：ユーザ認証方式の設定方法

SSH サーバのユーザ認証の設定

このタスクを実行して、セキュア シェル（SSH）サーバでのユーザ認証方式を設定します。

手順の概要

1. **enable**
2. **configure terminal**
3. **no ip ssh server authenticate user** {**publickey** | **keyboard** | **pasword**}
4. **ip ssh server authenticate user** {**publickey** | **keyboard** | **pasword**}
5. **default ip ssh server authenticate user**

6. **end**

手順の詳細

	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 モードをイネーブルにします。 • 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	**no ip ssh server authenticate user** {**publickey** | **keyboard** | **pasword**}
	例：
	`Device(config)# no ip ssh server authenticate user publickey`
	`%SSH:Publickey disabled.Overriding RFC`
		セキュア シェル（SSH）サーバでユーザ認証方式を無効にします。
		（注） **no ip ssh server authenticate user publickey** コマンドを使用して公開キー認証を無効にすると、警告メッセージが表示されます。このコマンドは、公開キー認証が必須であることが明記されている RFC 4252（セキュア シェル（SSH）認証プロトコル）の動作をオーバーライドします。

ステップ 4	**ip ssh server authenticate user** {**publickey** | **keyboard** | **pasword**} 例： `Device(config)# ip ssh server authenticate user publickey`	SSH サーバで無効になっているユーザ認証方法を有効にします。
ステップ 5	**default ip ssh server authenticate user** 例： `Device(config)# default ip ssh server authenticate user`	すべてのユーザ認証方式が事前に定義された順序で有効になっているデフォルトの動作に戻ります。
ステップ 6	**end** 例： `Device(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。

トラブルシューティングのヒント

- **no ip ssh server authenticate user publickey** コマンドを使用して公開キーベースの認証方式を無効にすると、公開キー認証が必須の RFC 4252（セキュア シェル（SSH）認証プロトコル）の動作がオーバーライドされ、次の警告メッセージが表示されます。

  ```
  %SSH:Publickey disabled.Overriding RFC
  ```

- 3 つすべての認証方式が無効になっている場合、次の警告メッセージが表示されます。

  ```
  %SSH:No auth method configured.Incoming connection will be dropped
  ```

- 3 つすべての認証方式が SSH サーバで無効になっているときに SSH クライアントから SSH セッション要求を受信した場合、接続要求は SSH サーバでドロップされ、次の形式でシステム ログ メッセージが表示されます。

  ```
  %SSH-3-NO_USERAUTH: No auth method configured for SSH Server. Incoming connection
  from <ip address> (tty = <ttynum>) dropped
  ```

SSH サーバのユーザ認証の確認

手順の概要

1. **enable**
2. **show ip ssh**

手順の詳細

ステップ 1 **enable**

特権 EXEC モードをイネーブルにします。

- 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**

セキュア シェル（SSH）のバージョンおよび設定データを表示します。

例：

次の **show ip ssh** コマンドの出力例では、3 つすべてのユーザ認証方式が SSH サーバで有効になっていることを確認します。

```
Device# show ip ssh

Authentication methods:publickey,keyboard-interactive,password
```

次の **show ip ssh** コマンドの出力例では、3 つすべてのユーザ認証方式が SSH サーバで無効になっていることを確認します。

```
Device# show ip ssh

Authentication methods:NONE
```

セキュア シェルの設定例：ユーザ認証方式の設定

例：ユーザ認証方式の無効化

次の例では、公開キーベースの認証方式およびキーボードベースの認証方式を無効にし、パスワードベースの認証方式を使用して SSH クライアントが SSH サーバに接続できるようにする方法を示します。

```
Device> enable
Device# configure terminal
Device(config)# no ip ssh server authenticate user publickey
%SSH:Publickey disabled.Overriding RFC
Device(config)# no ip ssh server authenticate user keyboard
Device(config)# exit
```

例：ユーザ認証方式の有効化

次の例では、公開キーベースの認証方式およびキーボードベースの認証方式を有効にする方法を示します。

```
Device> enable
Device# configure terminal
Device(config)# ip ssh server authenticate user publickey
Device(config)# ip ssh server authenticate user keyboard
Device(config)# exit
```

例：デフォルトのユーザ認証方式の設定

次の例では、3 つすべてのユーザ認証方式が事前に定義された順序で有効になっているデフォルトの動作に戻す方法を示します。

```
Device> enable
Device# configure terminal
```

```
Device(config)# default ip ssh server authenticate user
Device(config)# exit
```

セキュア シェルの追加情報：ユーザ認証方式の設定

関連資料

関連項目	マニュアル タイトル
Cisco IOS コマンド	Cisco IOS Master Command List, All Releases
セキュリティ コマンド	• 『Cisco IOS Security Command Reference: Commands A to C』 • 『Cisco IOS Security Command Reference: Commands D to L』 • 『Cisco IOS Security Command Reference: Commands M to R』 • 『Cisco IOS Security Command Reference: Commands S to Z』
SSH の設定	『セキュア シェル コンフィギュレーション ガイド』

標準および RFC

標準/RFC	タイトル
RFC 4252	『セキュア シェル（*SSH*）認証プロトコル』
RFC 4253	『セキュア シェル（*SSH*）トランスポート層プロトコル』

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

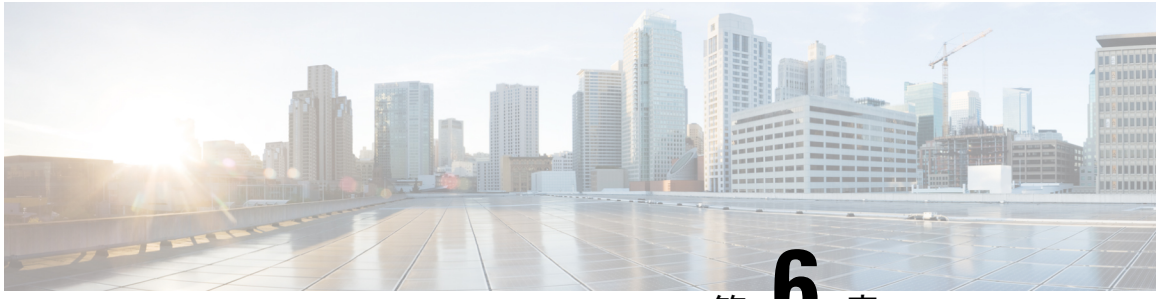
セキュア シェルの機能情報：ユーザ認証方式の設定

次の表に、このモジュールで説明した機能に関するリリース情報を示します。この表は、ソフトウェア リリース トレインで各機能のサポートが導入されたときのソフトウェア リリースだけを示しています。その機能は、特に断りがない限り、それ以降の一連のソフトウェア リリースでもサポートされます。

プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

表 *4*：セキュア シェルの機能情報：ユーザ認証方式の設定

機能名	リリース	機能情報
セキュア シェル：ユーザ認証方式の設定	Cisco IOS XE Release 3.10S	セキュア シェル：ユーザ認証方式の設定機能によって、セキュア シェル（SSH）サーバで使用可能なユーザ認証方式を設定できます。 次のコマンドが導入されました：**ip ssh server authenticate user**。 この機能は、Cisco IOS XE Release3.10 で、Cisco ASR 1000 シリーズ アグリゲーション サービス ルータに導入されました。



第 6 章

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、セキュアシェル（SSH）サーバ側でユーザ認証を使用します。

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。

機能情報の確認

ご使用のソフトウェア リリースでは、このモジュールで説明されるすべての機能がサポートされているとは限りません。最新の機能情報および警告については、「Bug Search Tool」およびご使用のプラットフォームおよびソフトウェア リリースのリリース ノートを参照してください。このモジュールで説明される機能に関する情報、および各機能がサポートされるリリースの一覧については、機能情報の表を参照してください。

プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

SSH 認証の X.509v3 証明書 の前提条件

- SSH 認証の X.509v3 証明書機能では、**ip ssh server authenticate user** コマンドの代わりに **ip ssh server algorithm authentication** コマンドが導入されます。**ip ssh server authenticate user** 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 CLI ineffective.
```

  - **default ip ssh server authenticate user** コマンドを使用して、**ip ssh server authenticate user** コマンドを無効にします。その後、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。

SSH 認証の X.509v3 証明書 の制約事項

- SSH 認証の X.509v3 証明書 機能の実装は、IOS セキュア シェル（SSH）側にのみ適用できます。
- IOS SSH サーバは、IOS SSH サーバ側の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

デジタル証明書

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.509v3 形式（RFC5280）のデジタル証明書は、アイデンティティの管理を実行するために使用されます。信頼できるルート証明機関とその中間証明機関による署名の連鎖によって、指定の公開署名キーと指定のデジタル アイデンティティがバインドされます。

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。証明書とトラストポイントを関連付けることによって、証明書を追跡できます。トラストポイントには、認証局（CA）、さまざまなアイデンティティ 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います。複数のトラストポイントを作成して、異なる証明書に関連付けることができます。

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、IOS セキュア シェル（SSH）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。

SSH 認証用の X.509v3 証明書の設定方法

サーバ認証にデジタル証明書を使用するための IOS SSH サーバの設定

手順の概要

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. **ocsp-response include**
8. **end**

手順の詳細

	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 モードをイネーブルにします。 • 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：	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。

	コマンドまたはアクション	目的
	`Device(config)# ip ssh server algorithm hostkey x509v3-ssh-rsa`	（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。 • ssh-rsa：公開キーベース認証 • x509v3-ssh-rsa：証明書ベース認証
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。
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 応答はサーバ証明書と一緒には送信されません。
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 モードを開始します。

ユーザ認証用のユーザのデジタル証明書を確認するための IOS SSH サーバの設定

手順の概要

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**

手順の詳細

	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 モードをイネーブルにします。 • 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、設定済みのアルゴリズムのみです。 （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。 （注） 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。 （注） **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。
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。 • ssh-rsa：公開キーベース認証 • x509v3-ssh-rsa：証明書ベース認証

	コマンドまたはアクション	目的
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。 （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） デフォルトではこのコマンドの「no」形式が設定されており、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 モードを開始します。

デジタル証明書を使用したサーバおよびユーザ認証の設定の確認

手順の概要

1. **enable**
2. **show ip ssh**

手順の詳細

ステップ 1 enable

特権 EXEC モードをイネーブルにします。

- 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
```

SSH 認証用の X.509v3 証明書の設定例

例：サーバ認証にデジタル証明書を使用するための IOS SSH サーバの設定

```
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：ユーザ認証用のユーザのデジタル証明書を確認するための IOS SSH サーバの設定

```
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
```

に関する追加情報 SSH 認証の X.509v3 証明書

関連資料

関連項目	マニュアル タイトル
Cisco IOS コマンド	Cisco IOS Master Command List, All Releases
セキュリティ コマンド	• 『Cisco IOS Security Command Reference: Commands A to C』 • 『Cisco IOS Security Command Reference: Commands D to L』 • 『Cisco IOS Security Command Reference: Commands M to R』 • 『Cisco IOS Security Command Reference: Commands S to Z』
SSH 認証	『セキュア シェル コンフィギュレーション ガイド』の「セキュア シェル：ユーザ認証方式の設定」の章
公開キー インフラストラクチャ（PKI）のトラストポイント	『*Public Key Infrastructure Configuration Guide*』の「Configuring and Managing a Cisco IOS Certificate Server for PKI Deployment」の章

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/cisco/web/support/index.html

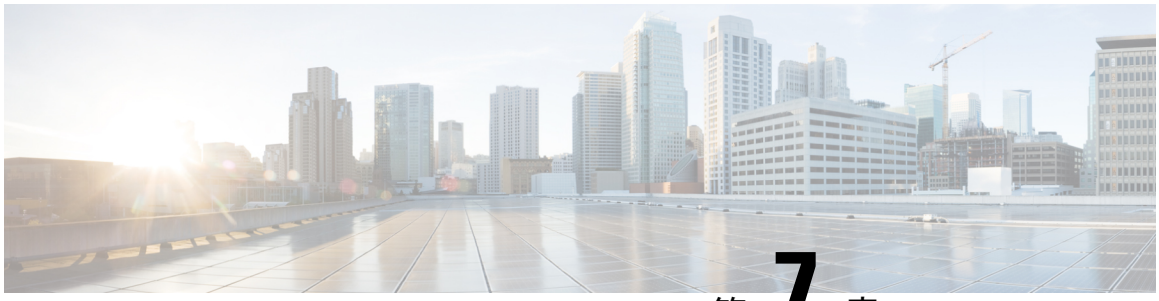
SSH 認証の X.509v3 証明書 の機能情報

次の表に、このモジュールで説明した機能に関するリリース情報を示します。この表は、ソフトウェア リリース トレインで各機能のサポートが導入されたときのソフトウェア リリースだけを示しています。その機能は、特に断りがない限り、それ以降の一連のソフトウェア リリースでもサポートされます。

プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

表 *5 : SSH* 認証の *X.509v3* 証明書 の機能情報

機能名	リリース	機能情報
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、セキュア シェル（SSH）サーバ側でユーザ認証を使用します。 次のコマンドが導入または変更されました：**ip ssh server algorithm hostkey**、**ip ssh server algorithm authentication**、**ip ssh server certificate profile**。



第 7 章

コモン クライテリア認定用の SSH アルゴリズム

コモン クライテリア認定用の SSH アルゴリズム 機能によって、コモン クライテリア認定を取得したアルゴリズムのリストおよび順序が提供されます。このモジュールでは、認定されたアルゴリズムのリストに基づいて SSH 接続を制限できるように、セキュア シェル（SSH）サーバおよびクライアントの暗号化、メッセージ認証コード（MAC）、およびホスト キー アルゴリズムの設定方法について説明します。

機能情報の確認

ご使用のソフトウェア リリースでは、このモジュールで説明されるすべての機能がサポートされているとは限りません。最新の機能情報および警告については、「Bug Search Tool」およびご使用のプラットフォームおよびソフトウェア リリースのリリース ノートを参照してください。このモジュールで説明される機能に関する情報、および各機能がサポートされるリリースの一覧については、機能情報の表を参照してください。

プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

コモン クライテリア認定用の SSH アルゴリズムの詳細

コモン クライテリア認定用の SSH アルゴリズム

セキュア シェル（SSH）設定によって、Cisco IOS SSH サーバおよびクライアントは、許可リストから設定されたアルゴリズムのネゴシエーションのみを許可することができます。リモートパーティが許可リストに含まれていないアルゴリズムのみを使用してネゴシエートしようとすると、要求は拒否され、セッションは確立されません。

Cisco IOS SSH サーバ アルゴリズム

Cisco IOS セキュア シェル（SSH）サーバは、次の順序で暗号化アルゴリズム（Advanced Encryption Standard カウンタ モード [AES-CTR]、AES 暗号ブロック連鎖 [AES-CBC]、Triple Data Encryption Standard [3DES]）をサポートします。

1. aes128-ctr
2. aes192-ctr
3. aes256-ctr
4. aes128-cbc
5. 3des-cbc
6. aes192-cbc
7. aes256-cbc

Cisco IOS SSH サーバは、次の順序でメッセージ認証コード（MAC）アルゴリズムをサポートします。

1. hmac-sha1
2. hmac-sha1-96

Cisco IOS SSH サーバは、次の順序でホスト キー アルゴリズムをサポートします。

1. x509v3-ssh-rsa
2. ssh-rsa

Cisco IOS SSH クライアント アルゴリズム

Cisco IOS セキュア シェル（SSH）クライアントは、次の順序で暗号化アルゴリズム（Advanced Encryption Standard カウンタ モード [AES-CTR]、AES 暗号ブロック連鎖 [AES-CBC]、Triple Data Encryption Standard [3DES]）をサポートします。

1. aes128-ctr

2. aes192-ctr
3. aes256-ctr
4. aes128-cbc
5. 3des-cbc
6. aes192-cbc
7. aes256-cbc

Cisco IOS SSH クライアントは、次の順序でメッセージ認証コード（MAC）アルゴリズムをサポートします。

1. hmac-sha1
2. hmac-sha1-96

Cisco IOS SSH クライアントがサポートするホスト キー アルゴリズムは 1 つのみで、CLI 設定は必要ありません。

- ssh-rsa

コモン クライテリア認定用の SSH アルゴリズム の設定方法

Cisco IOS SSH サーバおよびクライアントの暗号キー アルゴリズムの設定

手順の概要

1. **enable**
2. **configure terminal**
3. **ip ssh** {**server** | **client**} **algorithm encryption** {**aes128-ctr** | **aes192-ctr** | **aes256-ctr** | **aes128-cbc** | **3des-cbc** | **aes192-cbc** | **aes256-cbc**}
4. **end**

手順の詳細

	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 モードをイネーブルにします。 • パスワードを入力します（要求された場合）。

	コマンドまたはアクション	目的
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** | **client**} **algorithm encryption** {**aes128-ctr** | **aes192-ctr** | **aes256-ctr** | **aes128-cbc** | **3des-cbc** | **aes192-cbc** | **aes256-cbc**}
	例：
	`Device(config)# ip ssh server algorithm encryption aes128-ctr aes192-ctr aes256-ctr aes128-cbc 3des-cbc aes192-cbc aes256-cbc`
	`Device(config)# ip ssh client algorithm encryption aes128-ctr aes192-ctr aes256-ctr aes128-cbc 3des-cbc aes192-cbc aes256-cbc`
		SSH サーバおよびクライアントでの暗号化アルゴリズムの順序を定義します。この順序は、アルゴリズムのネゴシエーション時に指定されます。
		（注） Cisco IOS SSH サーバおよびクライアントには、1 つ以上の設定済み暗号化アルゴリズムが必要です。
		（注） 以前設定したアルゴリズムのリストから 1 つのアルゴリズムを無効にするには、このコマンドの **no** 形式を使用します。複数のアルゴリズムを無効にするには、このコマンドの **no** 形式を異なるアルゴリズム名で複数回使用します。
		（注） デフォルト設定では、次に示すようにこのコマンドのデフォルト形式を使用します。
		`Device(config)# ip ssh server algorithm encryption aes128-ctr aes192-ctr aes256-ctr aes128-cbc 3des-cbc aes192-cbc aes256-cbc`

ステップ 4	**end** 例： `Device(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。

トラブルシューティングのヒント

設定で最後の暗号化アルゴリズムを無効にしようとすると、次のメッセージが表示されてコマンドが拒否されます。

```
% SSH command rejected: All encryption algorithms cannot be disabled
```

Cisco IOS SSH サーバおよびクライアントの MAC アルゴリズムの設定

手順の概要

1. **enable**
2. **configure terminal**
3. **ip ssh** {**server** | **client**} **algorithm mac** {**hmac-sha1** | **hmac-sha1-96**}
4. **end**

手順の詳細

	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 モードをイネーブルにします。 • 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**	**client**} **algorithm mac** {**hmac-sha1**	**hmac-sha1-96**}
			例：
			`Device(config)# ip ssh server algorithm mac hmac-sha1 hmac-sha1-96`
			`Device(config)# ip ssh client algorithm mac hmac-sha1 hmac-sha1-96`
				SSH サーバおよびクライアントでの MAC（メッセージ認証コード）アルゴリズムの順序を定義します。この順序は、アルゴリズムのネゴシエーション時に指定されます。
				（注） Cisco IOS SSH サーバおよびクライアントには、1 つ以上の設定済みハッシュ メッセージ認証コード（HMAC）アルゴリズムが必要です。
				（注） 以前設定したアルゴリズムのリストから 1 つのアルゴリズムを無効にするには、このコマンドの **no** 形式を使用します。複数のアルゴリズムを無効にするには、このコマンドの **no** 形式を異なるアルゴリズム名で複数回使用します。
				（注） デフォルト設定では、次に示すようにこのコマンドのデフォルト形式を使用します。
				`Device(config)# ip ssh server algorithm mac hmac-sha1 hmac-sha1-96`

	コマンドまたはアクション	目的
ステップ 4	**end** 例： `Device(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。

トラブルシューティングのヒント

設定で最後の MAC アルゴリズムを無効にしようとすると、次のメッセージが表示されてコマンドが拒否されます。

```
% SSH command rejected: All mac algorithms cannot be disabled
```

Cisco IOS SSH サーバのホスト キー アルゴリズムの設定

手順の概要

1. **enable**
2. **configure terminal**
3. **ip ssh server algorithm hostkey** {**x509v3-ssh-rsa** | **ssh-rsa**}
4. **end**

手順の詳細

	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 モードをイネーブルにします。 • 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**} 例： `Device(config)# ip ssh server algorithm hostkey x509v3-ssh-rsa ssh-rsa`	ホストキーアルゴリズムの順序を定義します。Cisco IOS 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。

	コマンドまたはアクション	目的
		（注） Cisco 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：X.509v3 証明書ベース認証 • ssh-rsa：公開キーベース認証 （注） 以前設定したアルゴリズムのリストから 1 つのアルゴリズムを無効にするには、このコマンドの **no** 形式を使用します。複数のアルゴリズムを無効にするには、このコマンドの **no** 形式を異なるアルゴリズム名で複数回使用します。 （注） デフォルト設定では、次に示すようにこのコマンドのデフォルト形式を使用します。 `Device(config)# ip ssh server algorithm hostkey x509v3-ssh-rsa ssh-rsa`
ステップ 4	**end** 例： `Device(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。

トラブルシューティングのヒント

設定で最後のホスト キー アルゴリズムを無効にしようとすると、次のメッセージが表示されてコマンドが拒否されます。

```
% SSH command rejected: All hostkey algorithms cannot be disabled
```

コモン クライテリア認定用の SSH アルゴリズム の確認

手順の概要

1. **enable**
2. **show ip ssh**

手順の詳細

ステップ 1 enable

特権 EXEC モードをイネーブルにします。

- 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

設定済みのセキュア シェル（SSH）暗号化、ホスト キー、およびメッセージ認証コード（MAC）アルゴリズムを表示します。

例：

次の **show ip ssh** コマンドの出力例は、デフォルトの順序で設定された暗号化アルゴリズムを示しています。

```
Device# show ip ssh

Encryption Algorithms: aes128-ctr, aes192-ctr, aes256-ctr, aes128-cbc, 3des-cbc, aes192-cbc, aes256-cbc
```

次の **show ip ssh** コマンドの出力例は、デフォルトの順序で設定された MAC アルゴリズムを示しています。

```
Device# show ip ssh

MAC Algorithms: hmac-sha1 hmac-sha1-96
```

次の **show ip ssh** コマンドの出力例は、デフォルトの順序で設定されたホスト キー アルゴリズムを示しています。

```
Device# show ip ssh

Hostkey Algorithms: x509v3-ssh-rsa, ssh-rsa
```

設定例 コモン クライテリア認定用の SSH アルゴリズム

例：Cisco IOS SSH サーバの暗号キー アルゴリズムの設定

```
Device> enable
Device# configure terminal
Device(config)# ip ssh server algorithm encryption aes128-ctr aes192-ctr aes256-ctr
aes128-cbc 3des-cbc aes192-cbc aes256-cbc
Device(config)# end
```

例：Cisco IOS SSH クライアントの暗号キー アルゴリズムの設定

```
Device> enable
Device# configure terminal
Device(config)# ip ssh client algorithm encryption aes128-ctr aes192-ctr aes256-ctr
aes128-cbc 3des-cbc aes192-cbc aes256-cbc
Device(config)# end
```

例：Cisco IOS SSH サーバの MAC アルゴリズムの設定

```
Device> enable
Device# configure terminal
Device(config)# ip ssh server algorithm mac hmac-sha1 hmac-sha1-96
Device(config)# end
```

例：Cisco IOS SSH サーバ用のキー交換 DH グループの設定

```
Device> enable
Device# configure terminal
Device(config)# ip ssh server algorithm kex diffie-hellman-group-exchange-sha1
Device(config)# end
```

```
Device> enable
Device# configure terminal
Device(config)# ip ssh server algorithm kex diffie-hellman-group14-sha1
Device(config)# end
```

例：Cisco IOS SSH サーバのホスト キー アルゴリズムの設定

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

に関する追加情報 コモン クライテリア認定用の SSH アルゴリズム

関連資料

関連項目	マニュアル タイトル
Cisco IOS コマンド	Cisco IOS Master Command List, All Releases
セキュリティ コマンド	• 『Cisco IOS Security Command Reference: Commands A to C』 • 『Cisco IOS Security Command Reference: Commands D to L』 • 『Cisco IOS Security Command Reference: Commands M to R』 • 『Cisco IOS Security Command Reference: Commands S to Z』
SSH 認証	『セキュア シェル コンフィギュレーション ガイド』の「セキュア シェル：ユーザ認証方式の設定」の章
サーバおよびユーザ認証での X.509v3 デジタル証明書	『セキュア シェル コンフィギュレーション ガイド』の「SSH 認証の X.509v3 証明書」の章

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/cisco/web/support/index.html

コモン クライテリア認定用の SSH アルゴリズム の機能情報

次の表に、このモジュールで説明した機能に関するリリース情報を示します。この表は、ソフトウェア リリース トレインで各機能のサポートが導入されたときのソフトウェア リリースだけを示しています。その機能は、特に断りがない限り、それ以降の一連のソフトウェア リリースでもサポートされます。

プラットフォームのサポートおよびシスコ ソフトウェア イメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

表 *6:* コモン クライテリア認定用の *SSH* アルゴリズム の機能情報

機能名	リリース	機能情報
コモン クライテリア認定用の SSH アルゴリズム		コモンクライテリア認定用のSSHアルゴリズム機能によって、コモンクライテリア認定を取得したアルゴリズムのリストおよび順序が提供されます。このモジュールでは、認定されたアルゴリズムのリストに基づいてSSH接続を制限できるように、セキュア シェル（SSH）サーバおよびクライアントの暗号化、メッセージ認証コード（MAC）、およびホスト キー アルゴリズムの設定方法について説明します。 この機能により、次のコマンドが導入されました：**ip ssh** {**server** | **client**} **algorithm encryption**、**ip ssh** {**server** | **client**} **algorithm mac**。