Implementing Keychain Management

This module describes how to implement keychain management on. Keychain management is a common method of authentication to configure shared secrets on all entities that exchange secrets such as keys, before establishing trust with each other. Routing protocols and network management applications on Cisco IOS XR software often use authentication to enhance security while communicating with peers.

Feature History for Implementing Keychain Management

Release	Modification
Release 3.3.0	This feature was introduced.
Release 3.4.0	• Support for the MAC authentication algorithm was added. • Support for hitless key rollover and key acceptance tolerance were added.
Release 3.5.0	Support for hitless key rollover for Open Shortest Path First (OSPF) and Intermediate System-to-Intermediate System (IS-IS) was added.

Prerequisites for Configuring Keychain Management

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Restrictions for Implementing Keychain Management

You must be aware that changing the system clock impacts the validity of the keys in the existing configuration.

Information About Implementing Keychain Management

The keychain by itself has no relevance; therefore, it must be used by an application that needs to communicate by using the keys (for authentication) with its peers. The keychain provides a secure mechanism to handle the keys and rollover based on the lifetime. Border Gateway Protocol (BGP), Open Shortest Path First (OSPF), and Intermediate System-to-Intermediate System (IS-IS) use the keychain to implement a hitless key rollover for authentication. BGP uses TCP authentication, which enables the authentication option and sends the Message Authentication Code (MAC) based on the cryptographic algorithm configured for the keychain. For information about BGP, OSPF, and IS-IS keychain configurations, see *Routing Configuration Guide for Cisco CRS Routers*

To implement keychain management, you must understand the concept of key lifetime, which is explained in the next section.

Lifetime of Key

If you are using keys as the security method, you must specify the lifetime for the keys and change the keys on a regular basis when they expire. To maintain stability, each party must be able to store and use more than one key for an application at the same time. A keychain is a sequence of keys that are collectively managed for authenticating the same peer, peer group, or both.

Keychain management groups a sequence of keys together under a keychain and associates each key in the keychain with a lifetime.

Note Any key that is configured without a lifetime is considered invalid; therefore, the key is rejected during configuration.

The lifetime of a key is defined by the following options:

- Start-time—Specifies the absolute time.
- End-time—Specifies the absolute time that is relative to the start-time or infinite time.

Each key definition within the keychain must specify a time interval for which that key is activated; for example, lifetime. Then, during a given key's lifetime, routing update packets are sent with this activated key. Keys cannot be used during time periods for which they are not activated. Therefore, we recommend that for a given keychain, key activation times overlap to avoid any period of time for which no key is activated. If a time period occurs during which no key is activated, neighbor authentication cannot occur; therefore, routing updates can fail.

Multiple keychains can be specified.

How to Implement Keychain Management

This section contains the following procedures:

Configuring a Keychain

This task configures a name for the keychain.

You can create or modify the name of the keychain.

SUMMARY STEPS

1. **configure**
2. **key chain** *key-chain-name*
3. **commit**
4. **show key chain** *key-chain-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	

Step 2	**key chain** *key-chain-name*
	Example:
	`RP/0/RP0/CPU0:router(config)# key chain isis-keys`
	`RP/0/RP0/CPU0:router(config-isis-keys)#`
		Creates a name for the keychain.
		Note Configuring only the keychain name without any key identifiers is considered a nonoperation. When you exit the configuration, the router does not prompt you to commit changes until you have configured the key identifier and at least one of the global configuration mode attributes or keychain-key configuration mode attributes (for example, lifetime or key string).

Step 3	**commit**	
Step 4	**show key chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router# show key chain isis-keys`	(Optional) Displays the name of the keychain. **Note** The *key-chain-name* argument is optional. If you do not specify a name for the *key-chain-name* argument, all the keychains are displayed.

What to do next

After completing keychain configuration, see the Configuring a Tolerance Specification to Accept Keys, on page 3 section.

Configuring a Tolerance Specification to Accept Keys

This task configures the tolerance specification to accept keys for a keychain to facilitate a hitless key rollover for applications, such as routing and management protocols.

SUMMARY STEPS

1. **configure**
2. **key chain** *key-chain-name*

3. **accept-tolerance** *value* [**infinite**]
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**key chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config)# key chain isis-keys`	Creates a name for the keychain.
Step 3	**accept-tolerance** *value* [**infinite**] **Example:** `RP/0/RP0/CPU0:router(config-isis-keys)# accept-tolerance infinite`	Configures a tolerance value to accept keys for the keychain. • Use the *value* argument to set the tolerance range in seconds. The range is from 1 to 8640000. • Use the **infinite** keyword to specify that the tolerance specification is infinite.
Step 4	**commit**	

Configuring a Key Identifier for the Keychain

This task configures a key identifier for the keychain.

You can create or modify the key for the keychain.

SUMMARY STEPS

1. **configure**
2. **key chain** *key-chain-name*
3. **key** *key-id*
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**key chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config)# key chain isis-keys`	Creates a name for the keychain.
Step 3	**key** *key-id* **Example:** `RP/0/RP0/CPU0:router(config-isis-keys)# key 8`	Creates a key for the keychain. The key ID number is translated from decimal to hexadecimal to create the command mode subprompt. • Use the *key-id* argument as a 48-bit integer.

	Command or Action	Purpose
Step 4	**commit**	

What to do next

After configuring a key identifier for the keychain, see the Configuring the Text for the Key String, on page 5 section.

Configuring the Text for the Key String

This task configures the text for the key string.

SUMMARY STEPS

1. **configure**
2. **key chain** *key-chain-name*
3. **key** *key-id*
4. **key-string** [**clear** | **password**] *key-string-text*
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**key chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config)# key chain isis-keys`	Creates a name for the keychain.

Step 3	**key** *key-id*
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys)# key 8`
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)#`
		Creates a key for the keychain.

Step 4	**key-string** [**clear** | **password**] *key-string-text*
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)#`
	`key-string password 8`
		Specifies the text string for the key.
		• Use the **clear** keyword to specify the key string in clear text form; use the **password** keyword to specify the key in encrypted form.
		• For a string to be a valid password, it must comply with the following rules:
		•

Step 5	**commit**	

What to do next

After configuring the text for the key string, see the Configuring the Keys to Generate Authentication Digest for the Outbound Application Traffic, on page 6 section.

Determining the Valid Keys

This task determines the valid keys for local applications to authenticate the remote peers.

SUMMARY STEPS

1. **configure**
2. **key chain** *key-chain-name*
3. **key** *key-id*
4. **accept-lifetime** *start-time* [**duration** *duration-value* | **infinite** | *end-time*]
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**key chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config)# key chain isis-keys`	Creates a a name for the keychain.

Step 3	**key** *key-id*
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys)# key 8`
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)#`
		Creates a key for the keychain.

Step 4	**accept-lifetime** *start-time* [**duration** *duration-value* | **infinite** | *end-time*]
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys)# key 8`
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)#`
	`accept-lifetime 1:00:00 october 24 2005 infinite`
		(Optional) Specifies the validity of the key lifetime in terms of clock time.

Step 5	**commit**	

Configuring the Keys to Generate Authentication Digest for the Outbound Application Traffic

This task configures the keys to generate authentication digest for the outbound application traffic.

SUMMARY STEPS

1. **configure**
2. **key chain** *key-chain-name*
3. **key** *key-id*
4. **send-lifetime** *start-time* [**duration** *duration-value* | **infinite** | *end-time*]
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**key chain** *key-chain-name* **Example:** `RP/0/RP0/CPU0:router(config)# key chain isis-keys`	Creates a name for the keychain.

Step 3	**key** *key-id*
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys)# key 8`
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)#`
		Creates a key for the keychain.

Step 4	**send-lifetime** *start-time* [**duration** *duration-value*	**infinite**	*end-time*]
			Example:
			`RP/0/RP0/CPU0:router(config-isis-keys)# key 8`
			`RP/0/RP0/CPU0:router(config-isis-keys-0x8)#`
			`send-lifetime 1:00:00 october 24 2005 infinite`
				(Optional) Specifies the set time period during which an authentication key on a keychain is valid to be sent. You can specify the validity of the key lifetime in terms of clock time.
				In addition, you can specify a start-time value and one of the following values:
				• **duration** keyword (seconds)
				• **infinite** keyword
				• *end-time* argument
				If you intend to set lifetimes on keys, Network Time Protocol (NTP) or some other time synchronization method is recommended.

Step 5	**commit**	

Configuring the Cryptographic Algorithm

This task allows the keychain configuration to accept the choice of the cryptographic algorithm.

SUMMARY STEPS

1. **configure**
2. **key chain** *key-chain-name*

3. **key** *key-id*
4. **cryptographic-algorithm** [**HMAC-MD5** | **HMAC-SHA1-12** | **HMAC-SHA1-20** | **MD5** | **SHA-1**]
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	

Step 2	**key chain** *key-chain-name*
	Example:
	`RP/0/RP0/CPU0:router(config)# key chain isis-keys`
	`RP/0/RP0/CPU0:router(config-isis-keys)#`
		Creates a name for the keychain.

Step 3	**key** *key-id*
	Example:
	`RP/0/RP0/CPU0:router(config-isis-keys)# key 8`
	`RP/0/RP0/CPU0:router(config-isis-keys-0x8)#`
		Creates a key for the keychain.

Step 4	**cryptographic-algorithm** [**HMAC-MD5** | **HMAC-SHA1-12** | **HMAC-SHA1-20** | **MD5** | **SHA-1**] **Example:** `RP/0/RP0/CPU0:router(config-isis-keys-0x8)# cryptographic-algorithm MD5`	Specifies the choice of the cryptographic algorithm. You can choose from the following list of algorithms: • HMAC-MD5 • HMAC-SHA1-12 • HMAC-SHA1-20 • MD5 • SHA-1 The routing protocols each support a different set of cryptographic algorithms: • Border Gateway Protocol (BGP) supports only HMAC-MD5 and HMAC-SHA1-12. • Intermediate System-to-Intermediate System (IS-IS) supports HMAC-MD5, SHA-1, MD5. • Open Shortest Path First (OSPF) supports MD5, HMAC-MD5.
Step 5	**commit**	

Configuration Examples for Implementing Keychain Management

This section provides the following configuration example:

Configuring Keychain Management: Example

The following example shows how to configure keychain management:

```
configure
key chain isis-keys
accept-tolerance infinite
key 8
key-string mykey91abcd
cryptographic-algorithm MD5
send-lifetime 1:00:00 june 29 2006 infinite
accept-lifetime 1:00:00 june 29 2006 infinite
end

Uncommitted changes found, commit them? [yes]: yes

show key chain isis-keys

Key-chain: isis-keys/ -

accept-tolerance -- infinite
Key 8 -- text "1104000E120B520005282820"
  cryptographic-algorithm -- MD5
  Send lifetime:   01:00:00, 29 Jun 2006 - Always valid  [Valid now]
  Accept lifetime: 01:00:00, 29 Jun 2006 - Always valid [Valid now]
```

Additional References

The following sections provide references related to implementing keychain management.

Related Documents

Related Topic	Document Title
Keychain management commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Keychain Management Commands in the System Security Command Reference for Cisco CRS Routers*

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport