Chapter

# 5

# Deploying IPSec VPN in the Enterprise

## 5.1 Chapter Overview

In Chapters 3 and 4, the focus was on implementing a single site-to-site IPSec VPN and the different IKE peer authentication techniques. Chapter 5 is an extension of those chapters, and the emphasis is now on deploying multiple IPSec VPN sites in large enterprise networks.

When deploying an IPSec VPN in large enterprises, many factors affect the scalability, reliability, interoperability, and performance of the network. Some of these factors include network topology design, such as fully meshed and hub-and-spoke networks; network resiliency techniques, such as HSRP (hot standby router protocol); IP addressing services, such as NAT (network address translation); and performance parameters, such as fragmentation, IKE SA lifetimes, and IKE keepalives. In this chapter, the focus is on the design of scalable IPSec VPNs using the hub-and-spoke topology. Network resiliency, NAT, and performance optimization with IPSec are covered in Chapter 6.

## 5.2 Meshed Versus Hub-and-Spoke Networks

In a fully meshed IPSec network, every device in the network communicates with every other device via a unique IPSec tunnel. This becomes a scalability issue when the number of devices (or nodes) in the network increases. For instance, a 60-node network will require $n*(n - 1)/2$ tunnels (where n is the number of nodes), or 1770 tunnels to be exact!

Furthermore, with the growing number of nodes in the network, the configuration burden, as well as the complexity, becomes mammoth, and at a certain point it will not be possible to grow the size of the mesh anymore. Keeping track of so many tunnels also creates performance impacts such as heavy CPU utilization. In other words, the limiting factor in this topology is the number of tunnels that the devices can support at a reasonable CPU utilization.

Hub-and-spoke IPSec networks scale better because the hub site can expand to meet growing spoke capacity requirements. In this case, local spoke sites that require connectivity to other remote spoke sites are connected via the hub site, and this reduces the number of IPSec tunnels required for spoke-to-spoke communications. Instead of direct spoke-to-spoke communication, information is now exchanged indirectly between the spoke sites via the hub site. In this instance, a 60-node network will need only (n-1) tunnels, or 59 tunnels, this time.

The limiting factor in this topology is the significant bandwidth requirement for all the traffic that flows through the hub site, which includes all spoke-to-spoke traffic as well as spoke-to-hub traffic. In addition, not all VPN devices support spoke-to-spoke intercommunication via a hub site, and this could contribute an additional constraint. For Cisco routers, the minimum IOS software version to support the IPSec hub and spoke topology is 12.2(5).

## 5.3 Case Study 5.1: IPSec VPN in a Hub-and-Spoke Topology

### 5.3.1 Case overview and network topology

This case study demonstrates a hub-and-spoke IPSec network design that spans three routers: Brussels-R1 (spoke 1), Brussels-R2 (spoke 2), and Brussels-R3 (hub). In this instance, information is exchanged between the spoke sites by traversing through the hub. In other words, as illustrated in Figure 5.1, no direct IPSec tunnel exists between the two spoke routers. All data packets are

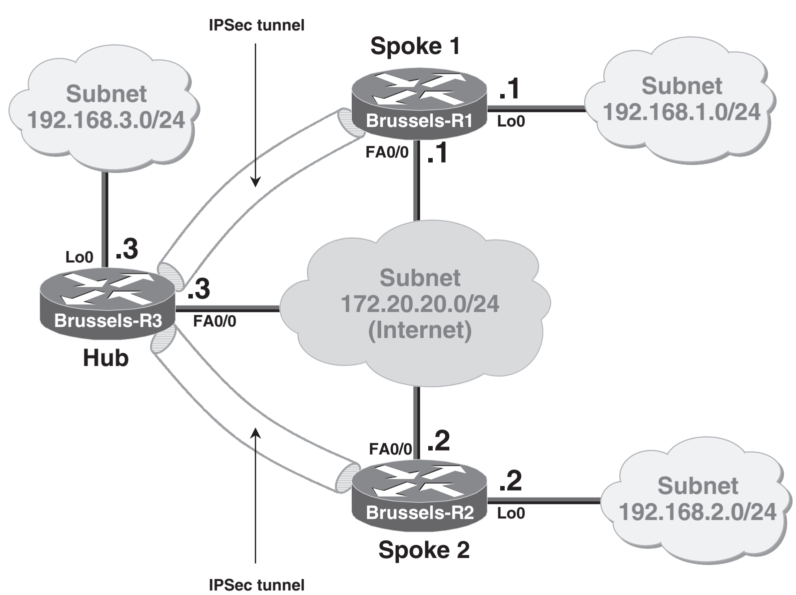


**Figure 5.1:** Hub-and-spoke topology for Case Study 5.1.

sent across the tunnel to the hub router where it redistributes them through the IPSec tunnel that is shared with the other spoke router.

In the case study, encryption is implemented from:

- Spoke 1 (subnet 192.168.1.0/24) to hub (subnet 192.168.3.0/24) and vice versa.
- Spoke 2 (subnet 192.168.2.0/24) to hub (subnet 192.168.3.0/24) and vice versa.
- Spoke 1 (subnet 192.168.1.0/24) to spoke 2 (subnet 192.168.2.0/24) and vice versa.

### 5.3.2 Hub-and-spoke IPSec configurations

Code Listings 5.1 to 5.3 are the hub-and-spoke IPSec configurations for Brussels-R1, Brussels-R2, and Brussels-R3, respectively. In the configurations, comments (in *italics*) precede certain configuration lines to explain them.

Code Listing 5.1: IPSec spoke configuration for Brussels-R1

```
hostname Brussels-R1
!
! Supersede the default policy and use pre-shared keys
! for peer authentication and hash algorithm MD5.
crypto isakmp policy 10
 hash md5
 authentication pre-share
!
! Specify the pre-shared key "key31" to be used
! for the IPSec tunnel between Brussels-R1 and Brussels-R3
crypto isakmp key key31 address 172.20.20.3
!
! Specify the ESP transform settings for IPSec,
! which is later applied to the crypto map.
crypto ipsec transform-set chapter5 esp-des esp-md5-hmac
!
! Specify the crypto map chapter5 where we define
! our hub peer Brussels-R3, transform set chapter5,
! and our crypto access list 110.
crypto map chapter5 10 ipsec isakmp
 set peer 172.20.20.3
 set transform-set chapter5
 match address 110
!
! Emulate subnet 192.168.1.0/24 with a loop-back interface.
interface Loopback0
 ip address 192.168.1.1 255.255.255.0
```

```
!
interface FastEthernet0/0
 ip address 172.20.20.1 255.255.255.0
! Apply the crypto map to an interface
! to activate crypto engine.
 crypto map chapter5
!
! Static routes are configured for
! subnet 192.168.2.0/24 and 192.168.3.0/24.
ip route 192.168.2.0 255.255.255.0 FastEthernet0/0
ip route 192.168.3.0 255.255.255.0 FastEthernet0/0
!
! This is the crypto access list that we
! referenced in crypto map chapter5.
! We are encrypting IP traffic between subnets
! 192.168.1.0/24 and 192.168.3.0/24 (between spoke 1 and hub),
! as well as between subnets 192.168.1.0/24
! and 192.168.2.0/24 (between spoke 1 and spoke 2).
access-list 110 permit ip 192.168.1.0 0.0.0.255 192.168.3.0 0.0.0.255
access-list 110 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
```

Code Listing 5.1 illustrates the IPSec spoke-end configuration for Brussels-R1 (spoke 1). Take note of the crypto access list, which is used to define the IP traffic flows that Brussels-R1 is encrypting. In this case, the traffic flows are from spoke 1 to hub and from spoke 1 to spoke 2.

Code Listing 5.2: IPSec spoke configuration for Brussels-R2

```
hostname Brussels-R2
!
! Supersede the default policy and use pre-shared keys
! for peer authentication and hash algorithm MD5.
crypto isakmp policy 10
 hash md5
 authentication pre-share
!
! Specify the pre-shared key "key32" to be used
! for the IPSec tunnel between Brussels-R2 and Brussels-R3.
crypto isakmp key key32 address 172.20.20.3
!
! Specify the ESP transform settings for IPSec,
! which is later applied to the crypto map.
crypto ipsec transform-set chapter5 esp-des esp-md5-hmac
```

```
!
! Specify the crypto map chapter5 where we define
! our hub peer Brussels-R3, transform set chapter5,
! and our crypto access list 120.
crypto map chapter5 10 ipsec-isakmp
 set peer 172.20.20.3
 set transform-set chapter5
 match address 120
!
! Emulate subnet 192.168.2.0/24 with a loop-back interface.
interface Loopback0
 ip address 192.168.2.2 255.255.255.0
!
interface FastEthernet0/0
 ip address 172.20.20.2 255.255.255.0
! Apply the crypto map to an interface
! to activate crypto engine.
 crypto map chapter5
!
! Static routes are configured for
! subnet 192.168.1.0/24 and 192.168.3.0/24.
ip route 192.168.1.0 255.255.255.0 FastEthernet0/0
ip route 192.168.3.0 255.255.255.0 FastEthernet0/0
!
! This is the crypto access list that we have
! referenced in crypto map chapter5.
! We are encrypting IP traffic between subnets 192.168.2.0/24
! and 192.168.3.0/24 (between spoke 2 and hub), as well as
! between subnets 192.168.2.0/24
! and 192.168.1.0/24 (between spoke 2 and spoke 1).
access-list 120 permit ip 192.168.2.0 0.0.0.255 192.168.3.0 0.0.0.255
access-list 120 permit ip 192.168.2.0 0.0.0.255 192.168.1.0 0.0.0.255
```

Code Listing 5.2 illustrates the IPSec spoke-end configuration for Brussels-R2 (spoke 2). Note the crypto access list, which is used to define the IP traffic flows that Brussels-R2 is encrypting. In this case, the traffic flows are from spoke 2 to hub, and from spoke 2 to spoke 1.

Code Listing 5.3: IPSec Hub Configuration for Brussels-R3

```
hostname Brussels-R3
!
! Supersede the default policy and use pre-shared keys
! for peer authentication and hash algorithm MD5.
```

```
crypto isakmp policy 10
 hash md5
 authentication pre-share
!
! Specify the pre-shared key "key32" to be used for the
! IPSec tunnel between Brussels-R3 and Brussels-R2.
crypto isakmp key key32 address 172.20.20.2
!
! Specify the pre-shared key "key31" to be used for
! the IPSec tunnel between Brussels-R3 and Brussels-R1
crypto isakmp key key31 address 172.20.20.1
!
! Specify the ESP transform settings for IPSec,
! which is later applied to the crypto map.
crypto ipsec transform-set chapter5 esp-des esp-md5-hmac
!
! Specify the first crypto map instance where we define
! our spoke peer Brussels-R1, transform set chapter5,
! and a unique crypto access list 110.
crypto map chapter5 10 ipsec-isakmp
 set peer 172.20.20.1
 set transform-set chapter5
 match address 110
!
! Specify the first crypto map instance where we define
! our spoke peer Brussels-R2, transform set chapter5,
! and a unique crypto access list 120.
crypto map chapter5 20 ipsec-isakmp
 set peer 172.20.20.2
 set transform-set chapter5
 match address 120
!
! Emulate subnet 192.168.3.0/24 with a loop-back interface.
interface Loopback0
 ip address 192.168.3.3 255.255.255.0
!
interface FastEthernet0/0
 ip address 172.20.20.3 255.255.255.0
! Apply the crypto map to an interface
! to activate crypto engine.
 crypto map chapter5
!
! Static routes are configured for
! subnet 192.168.1.0/24 and 192.168.2.0/24.
ip route 192.168.1.0 255.255.255.0 FastEthernet0/0
ip route 192.168.2.0 255.255.255.0 FastEthernet0/0
```

```
!
! In access list 110, we are encrypting IP traffic between subnets
! 192.168.3.0/24 and 192.168.1.0/24 (between hub and spoke 1),
! as well as between subnets 192.168.2.0/24 and
! 192.168.1.0/24 (between spoke 2 and spoke 1).
access-list 110 permit ip 192.168.3.0 0.0.0.255 192.168.1.0 0.0.0.255
access-list 110 permit ip 192.168.2.0 0.0.0.255 192.168.1.0 0.0.0.255
!
! In access list 120, we are encrypting IP traffic between subnets
! 192.168.3.0/24 and 192.168.2.0/24 (between hub and spoke 2),
! as well as between subnets 192.168.1.0/24
! and 192.168.2.0/24 (between spoke 1 and spoke 2).
access-list 120 permit ip 192.168.3.0 0.0.0.255 192.168.2.0 0.0.0.255
access-list 120 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
```

Code Listing 5.3 illustrates the IPSec hub-end configuration for Brussels-R3 (hub). Note the two crypto access lists, which are used to define the IP traffic flows that Brussels-R3 is encrypting. In this case, the traffic flows are from hub to spoke 1 and from spoke 2 to spoke 1 (via hub), as well as from hub to spoke 2 and from spoke 1 to spoke 2 (via hub).

### 5.3.3 Verifying crypto access lists

In Code Listings 5.4 to 5.6, we verify the crypto access lists that we have configured for the three routers by examining their respective crypto maps using the "show crypto map" command.

Code Listing 5.4: Brussels-R1 crypto map

```
Brussels-R1#show crypto map
Crypto Map "chapter5" 10 ipsec-isakmp
   Peer = 172.20.20.3
   Extended IP access list 110
       access-list 110 permit ip 192.168.1.0 0.0.0.255 192.168.3.0 0.0.0.255
       access-list 110 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
   Current peer: 172.20.20.3
   Security association lifetime: 4608000 kilobytes/3600 seconds
   PFS (Y/N): N
   Transform sets={ chapter5, }
   Interfaces using crypto map chapter5:
      FastEthernet0/0
```

From Code Listing 5.4, we can verify the crypto access list that we configured earlier for Brussels-R1 is encrypting IP traffic between subnets 192.168.1.0/24 and 192.168.3.0/24 (between spoke 1 and hub), as well as between subnets 192.168.1.0/24 and 192.168.2.0/24 (between spoke 1 and spoke 2).

From Code Listing 5.5, we can verify the crypto access list that we have configured earlier for Brussels-R2 is encrypting IP traffic between subnets 192.168.2.0/24 and 192.168.1.0/24 (between spoke 2 and spoke 1), as well as between subnets 192.168.2.0/24 and 192.168.3.0/24 (between spoke 2 and hub).

Code Listing 5.5: Brussels-R2 crypto map

```
Brussels-R2#show crypto map
Crypto Map "chapter5" 10 ipsec-isakmp
   Peer = 172.20.20.3
   Extended IP access list 120
       access-list 120 permit ip 192.168.2.0 0.0.0.255 192.168.1.0 0.0.0.255
       access-list 120 permit ip 192.168.2.0 0.0.0.255 192.168.3.0 0.0.0.255
   Current peer: 172.20.20.3
   Security association lifetime: 4608000 kilobytes/3600 seconds
   PFS (Y/N): N
   Transform sets={ chapter5, }
   Interfaces using crypto map chapter5:
      FastEthernet0/0
```

From Code Listing 5.6, we can verify the two crypto access lists that we have configured earlier for Brussels-R3. In access list 110, we are encrypting IP traffic between subnets 192.168.3.0/24 and 192.168.1.0/24 (between hub and spoke 1), as well as between subnets 192.168.2.0/24 and 192.168.1.0/24 (between spoke 2 and spoke 1). In access list 120, we are encrypting IP traffic between subnets 192.168.3.0/24 and 192.168.2.0/24 (between hub and spoke 2), as well as between subnets 192.168.1.0/24 and 192.168.2.0/24 (between spoke 1 and spoke 2).

Code Listing 5.6: Brussels-R3 Crypto Map

```
Brussels-R3#show crypto map
Crypto Map "chapter5" 10 ipsec-isakmp
   Peer = 172.20.20.1
   Extended IP access list 110
       access-list 110 permit ip 192.168.3.0 0.0.0.255 192.168.1.0 0.0.0.255
       access-list 110 permit ip 192.168.2.0 0.0.0.255 192.168.1.0 0.0.0.255
   Current peer: 172.20.20.1
   Security association lifetime: 4608000 kilobytes/3600 seconds
```

```
   PFS (Y/N): N
   Transform sets={ chapter5, }

Crypto Map "chapter5" 20 ipsec-isakmp
   Peer = 172.20.20.2
   Extended IP access list 120
       access-list 120 permit ip 192.168.3.0 0.0.0.255 192.168.2.0 0.0.0.255
       access-list 120 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
   Current peer: 172.20.20.2
   Security association lifetime: 4608000 kilobytes/3600 seconds
   PFS (Y/N): N
   Transform sets={ chapter5, }
   Interfaces using crypto map chapter5:
      FastEthernet0/0
```

### 5.3.4 Monitoring and verifying hub-and-spoke IPSec operations

Code Listing 5.7 illustrates the ping test disseminated from spoke 1 (192.168.1.1) to spoke 2 (192.168.2.2). In this instance, the test is only 60 percent successful, that is, 3 of 5 ping packets went through to spoke 2.

Code Listing 5.7: Ping test #1 from spoke 1 to spoke 2

```
Brussels-R1#ping
Protocol [ip]:
Target IP address: 192.168.2.2
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 192.168.1.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.2.2, timeout is 2 seconds:
..!!!
Success rate is 60 percent (3/5), round-trip min/avg/max = 12/13/16 ms
```

Code Listing 5.8 illustrates the results for ping test #1 conducted in Code Listing 5.7. In the command traces below, italic comments are inserted at

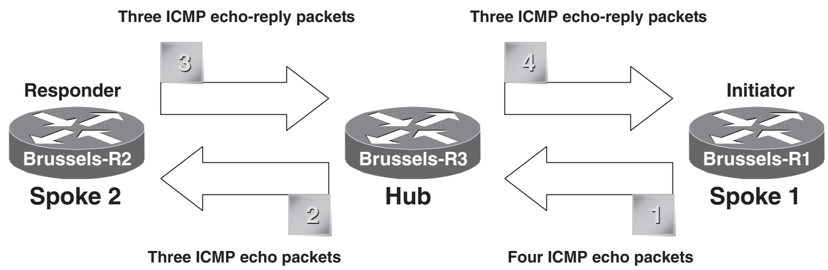


**Figure 5.2:** Traffic flow for ping test #1.

specific "checkpoints" for description purposes. We shall examine the traces from Brussels-R1, Brussels-R3, and Brussels-R2 in that order. The traffic flow for ping test #1 is illustrated in Figure 5.2.

Code Listing 5.8: Results for ping test #1

*We first examine the respective command outputs at Brussels-R1. The "show crypto isakmp sa" command at Brussels-R1 (Spoke 1) indicates that the IKE tunnel is created from spoke 1 (172.20.20.1) to hub (172.20.20.3) when the IP traffic flows from spoke 1 to spoke 2. Note that the connection ID assigned is 1.*

```
Brussels-R1#show crypto isakmp sa
dst            src            state        conn-id  slot
172.20.20.3    172.20.20.1    QM_IDLE            1     0
```

*From the "show crypto ipsec sa" command at Brussels-R1 (spoke 1), for the IPSec tunnel created between spoke 1 and hub, we can gather that four of five outgoing IP (ICMP echo) packets to spoke 2 have been encrypted (by DES) and digested (by MD5), and only three incoming IP (ICMP echo-reply) packets from spoke 2 have been decrypted (by DES) and verified (by MD5).*

```
Brussels-R1#show crypto ipsec sa
interface: FastEthernet0/0
    Crypto map tag: chapter5, local addr. 172.20.20.1
```

*Traffic flow is from local subnet 192.168.1.0/24 to remote subnet 192.168.2.0/24.*

```
  local ident (addr/mask/prot/port): (192.168.1.0/255.255.255.0/0/0)
  remote ident (addr/mask/prot/port): (192.168.2.0/255.255.255.0/0/0)
```

```
  current_peer: 172.20.20.3
    PERMIT, flags={origin_is_acl,}
```

*Four outgoing (ICMP echo) packets from spoke 1 to spoke 2 are encrypted and digested (see Fig. 5.2).*

```
  #pkts encaps: 4, #pkts encrypt: 4, #pkts digest 4
```

*Three incoming (ICMP echo-reply) packets from spoke 2 are decrypted and verified (see Fig. 5.2).*

```
  #pkts decaps: 3, #pkts decrypt: 3, #pkts verify 3
  <Output Omitted>
```

*At spoke 1, of the five packets sent, one is bad.*

```
  #send errors 1, #recv errors 0
```

*The IPSec tunnel originates from spoke 1 (172.20.20.1) and terminates at hub (172.20.20.3).*

```
   local crypto endpt.: 172.20.20.1, remote crypto endpt.: 172.20.20.3
   <Output Omitted>
```

*For the inbound IPSec SA from hub to spoke 1, the SPI is 1605659124 and the connection ID is 2000.*

```
   inbound esp sas:
   spi: 0x5FB469F4(1605659124)
     transform: esp-des esp-md5-hmac ,
     in use settings ={Tunnel, }
     slot: 0, conn id: 2000, flow_id: 1, crypto map: chapter5
     <Output Omitted>
```

*For the outbound IPSec SA from spoke 1 to hub, the SPI is 1837161637 and the connection ID is 2001.*

```
   outbound esp sas:
   spi: 0x6D80DCA5(1837161637)
     transform: esp-des esp-md5-hmac ,
     in use settings ={Tunnel, }
     slot: 0, conn id: 2001, flow_id: 2, crypto map: chapter5
     <Output Omitted>
```

*The "show crypto engine connection active" command at Brussels-R1 concurs with the "show crypto ipsec sa" command reflecting four outbound (ICMP echo) packets being encrypted (and digested) and three inbound (ICMP echo-reply) packets being decrypted and verified. The connection IDs for the inbound and outbound IPSec SAs are 2000 and 2001, respectively, which correspond to the same connection ID values illustrated earlier in the "show crypto ipsec sa" command.*

**Brussels-R1#show crypto engine connection active**

| ID | Interface | IP-Address | State | Algorithm | Encrypt | Decrypt |
|---|---|---|---|---|---|---|
| **1** | <none> | <none> | set | HMAC_MD5+DES_56_CB | 0 | 0 |
| **2000** | FastEthernet0/0 | 172.20.20.1 | set | HMAC_MD5+DES_56_CB | **0** | **3** |
| **2001** | FastEthernet0/0 | 172.20.20.1 | set | HMAC_MD5+DES_56_CB | **4** | **0** |

*Next, we examine the respective command outputs at Brussels-R3. The "show crypto isakmp sa" command at Brussels-R3 (hub) indicates that two IKE tunnels are created when the IP traffic flows from spoke 1 to spoke 2, one from spoke 1 (172.20.20.1) to hub (172.20.20.3) and the other from hub (172.20.20.3) to spoke 2 (172.20.20.2). Note that the connection IDs assigned are 1 and 2, respectively.*

**Brussels-R3#show crypto isakmp sa**

| dst | src | state | conn-id | slot |
|---|---|---|---|---|
| **172.20.20.2** | **172.20.20.3** | QM_IDLE | **2** | 0 |
| **172.20.20.3** | **172.20.20.1** | QM_IDLE | **1** | 0 |

*From the "show crypto ipsec sa" command at Brussels-R3 (hub), for the IPSec tunnel created between hub and spoke 1, we can gather that three outgoing IP (ICMP echo-reply) packets from spoke 2 to spoke 1 have been encrypted, as well as digested, and four incoming IP (ICMP echo) packets from spoke 1 to spoke 2 have been decrypted and verified. For the IPSec tunnel created between hub and spoke 2, we can gather that three of four outgoing IP (ICMP echo) packets from spoke 1 to spoke 2 have been encrypted and digested, and three incoming IP (ICMP echo-reply) packets from spoke 2 to spoke 1 have been decrypted and verified.*

**Brussels-R3#show crypto ipsec sa**

interface: FastEthernet0/0
   Crypto map tag: chapter5, **local addr. 172.20.20.3**

*Traffic flow is from local subnet 192.168.2.0/24 to remote subnet 192.168.1.0/24.*

  **local ident** (addr/mask/prot/port): (**192.168.2.0/255.255.255.0**/0/0)
  **remote ident** (addr/mask/prot/port): (**192.168.1.0/255.255.255.0**/0/0)
  **current_peer: 172.20.20.1**
    PERMIT, flags={origin_is_acl,}

*Three outgoing (ICMP echo-reply) packets from spoke 2 to spoke 1 are encrypted and digested (see Fig. 5.2).*

  #pkts encaps: 3, **#pkts encrypt: 3, #pkts digest 3**

*Four incoming (ICMP echo) packets from spoke 1 to spoke 2 are decrypted and verified (see Fig. 5.2).*

```
  #pkts decaps: 4, #pkts decrypt: 4, #pkts verify 4
  <Output Omitted>
```

*This first IPSec tunnel originates from hub (172.20.20.3) and terminates at spoke 1 (172.20.20.1).*

```
   local crypto endpt.: 172.20.20.3, remote crypto endpt.: 172.20.20.1
   <Output Omitted>
```

*For the inbound IPSec SA from spoke 1 to hub, the SPI is 1837161637 and the connection ID is 2000.*

```
   inbound esp sas:
   spi: 0x6D80DCA5(1837161637)
     transform: esp-des esp-md5-hmac ,
     in use settings ={Tunnel, }
     slot: 0, conn id: 2000, flow_id: 1, crypto map: chapter5
     <Output Omitted>
```

*For the outbound IPSec SA from hub to spoke 1, the SPI is 1605659124 and the connection ID is 2001.*

```
   outbound esp sas:
   spi: 0x5FB469F4(1605659124)
     transform: esp-des esp-md5-hmac ,
     in use settings ={Tunnel, }
     slot: 0, conn id: 2001, flow_id: 2, crypto map: chapter5
     <Output Omitted>
```

*Traffic flow is from local subnet 192.168.1.0/24 to remote subnet 192.168.2.0/24.*

```
  local ident (addr/mask/prot/port): (192.168.1.0/255.255.255.0/0/0)
  remote ident (addr/mask/prot/port): (192.168.2.0/255.255.255.0/0/0)
  current_peer: 172.20.20.2
    PERMIT, flags={origin_is_acl,}
```

*Three outgoing (ICMP echo) packets from spoke 1 to spoke 2 are encrypted and digested (see Fig. 5.2).*

```
  #pkts encaps: 3, #pkts encrypt: 3, #pkts digest 3
```

*Three incoming (ICMP echo-reply) packets from spoke 2 to spoke 1 are decrypted and verified (see Fig. 5.2).*

```
  #pkts decaps: 3, #pkts decrypt: 3, #pkts verify 3
  <Output Omitted>
```

*At the hub, of the four traversing packets from spoke 1 to spoke 2, one is bad.*

```
  #send errors 1, #recv errors 0
```

*This other IPSec tunnel originates from hub (172.20.20.3) and terminates at spoke 2 (172.20.20.2).*

```
local crypto endpt.: 172.20.20.3, remote crypto endpt.: 172.20.20.2
<Output Omitted>
```

*For the inbound IPSec SA from spoke 2 to hub, the SPI is 2981567991 and the connection ID is 2002.*

```
inbound esp sas:
spi: 0xB1B71DF7(2981567991)
  transform: esp-des esp-md5-hmac ,
  in use settings ={Tunnel, }
  slot: 0, conn id: 2002, flow_id: 3, crypto map: chapter5
  <Output Omitted>
```

*For the outbound IPSec SA from hub to spoke 2, the SPI is 2142640922, and the connection ID is 2003.*

```
outbound esp sas:
spi: 0x7FB61B1A(2142640922)
  transform: esp-des esp-md5-hmac ,
  in use settings ={Tunnel, }
  slot: 0, conn id: 2003, flow_id: 4, crypto map: chapter5
  <Output Omitted>
```

*The "show crypto engine connection active" command at Brussels-R3 concurs with the "show crypto ipsec sa" command. For the IPSec tunnel between hub and spoke 1, the command reflects three outbound (ICMP echo-reply) packets being encrypted and digested together with four inbound (ICMP echo) packets being decrypted and verified, and the connections IDs for the inbound and outbound IPSec SAs between hub and spoke 1 are 2000 and 2001, respectively. For the IPSec tunnel between hub and spoke 2, the command reflects three outbound (ICMP echo) packets being encrypted and digested together with three inbound (ICMP echo-reply) packets being decrypted and verified, and the connection IDs for the inbound and outbound IPSec SAs between hub and spoke 2 correspond with 2002 and 2003, respectively.*

```
Brussels-R3#show crypto engine connection active

  ID Interface       IP-Address    State Algorithm          Encrypt  Decrypt
   1 FastEthernet0/0 172.20.20.3   set   HMAC_MD5+DES_56_CB       0        0
   2 <none>          <none>        set   HMAC_MD5+DES_56_CB       0        0
2000 FastEthernet0/0 172.20.20.3   set   HMAC_MD5+DES_56_CB       0        4
2001 FastEthernet0/0 172.20.20.3   set   HMAC_MD5+DES_56_CB       3        0
2002 FastEthernet0/0 172.20.20.3   set   HMAC_MD5+DES_56_CB       0        3
2003 FastEthernet0/0 172.20.20.3   set   HMAC_MD5+DES_56_CB       3        0
```

*We now examine the respective command outputs at Brussels-R2. The "show crypto isakmp sa" command at Brussels-R2 (spoke 2) indicates that the IKE tunnel is created from hub (172.20.20.3) to spoke 2 (172.20.20.2) when the IP traffic flows from spoke 1 to spoke 2. Note the connection ID assigned is 1.*

```
Brussels-R2#show crypto isakmp sa
dst            src            state        conn-id    slot
172.20.20.2    172.20.20.3    QM_IDLE            1     0
```

*From the "show crypto ipsec sa" command at Brussels-R2 (spoke 2), for the IPSec tunnel created between spoke 2 and hub, we can gather that three outgoing IP (ICMP echo-reply) packets to spoke 1 have been encrypted and digested, and three incoming IP (ICMP echo) packets from spoke 1 have been decrypted and verified.*

```
Brussels-R2#show crypto ipsec sa

interface: FastEthernet0/0
  Crypto map tag: chapter5, local addr. 172.20.20.2
```

*Traffic flow is from local subnet 192.168.2.0/24 to remote subnet 192.168.1.0/24.*

```
  local ident (addr/mask/prot/port): (192.168.2.0/255.255.255.0/0/0)
  remote ident (addr/mask/prot/port): (192.168.1.0/255.255.255.0/0/0)
  current_peer: 172.20.20.3
    PERMIT, flags={origin_is_acl,}
```

*Three outgoing (ICMP echo-reply) packets from spoke 2 to spoke 1 are encrypted and digested (see Fig. 5.2).*

```
  #pkts encaps: 3, #pkts encrypt: 3, #pkts digest 3
```

*Three incoming (ICMP echo) packets from spoke 1 are decrypted and verified (see Fig. 5.2).*

```
  #pkts decaps: 3, #pkts decrypt: 3, #pkts verify 3
  <Output Omitted>
```

*The IPSec tunnel originates from spoke 2 (172.20.20.2) and terminates at hub (172.20.20.3).*

```
   local crypto endpt.: 172.20.20.2, remote crypto endpt.: 172.20.20.3
   <Output Omitted>
```

*For the inbound IPSec SA from hub to spoke 2, the SPI is 2142640922 and the connection ID is 2000.*

```
   inbound esp sas:
   spi: 0x7FB61B1A(2142640922)
```

```
      transform: esp-des esp-md5-hmac ,
      in use settings ={Tunnel, }
      slot: 0, conn id: 2000, flow_id: 1, crypto map: chapter5
      sa timing: remaining key lifetime (k/sec): (4607999/3435)
      <Output Omitted>
```

*For the outbound IPSec SA from spoke 2 to hub, the SPI is 2981567991 and the connection ID is 2001.*

```
   outbound esp sas:
   spi: 0xB1B71DF7(2981567991)
      transform: esp-des esp-md5-hmac ,
      in use settings ={Tunnel, }
      slot: 0, conn id: 2001, flow_id: 2, crypto map: chapter5
      <Output Omitted>
```

*The "show crypto engine connection active" command at Brussels-R2 concurs with the "show crypto ipsec sa" command reflecting three outbound (ICMP echo-reply) packets being encrypted and digested, and three inbound (ICMP echo) packets being decrypted and verified. The connections IDs for the inbound and outbound IPSec SAs are 2000 and 2001, respectively, which correspond to the same connection ID values illustrated earlier in the "show crypto ipsec sa" command.*

```
Brussels-R2#show crypto engine connection active

  ID Interface        IP-Address     State Algorithm           Encrypt   Decrypt
   1 FastEthernet0/0  172.20.20.3    set   HMAC_MD5+DES_56_CB        0         0
2000 FastEthernet0/0  172.20.20.3    set   HMAC_MD5+DES_56_CB        0         3
2001 FastEthernet0/0  172.20.20.3    set   HMAC_MD5+DES_56_CB        3         0
```

Code Listing 5.9 illustrates the ping test disseminated from spoke 2 (192.168.2.2) to spoke 1 (192.168.1.1). In this instance, the test is 100 percent successful, that is, 5 of 5 ping packets went through to spoke 1.

Code Listing 5.9: Ping Test #2 from spoke 2 to spoke 1

```
Brussels-R2#ping
Protocol [ip]:
Target IP address: 192.168.1.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
```

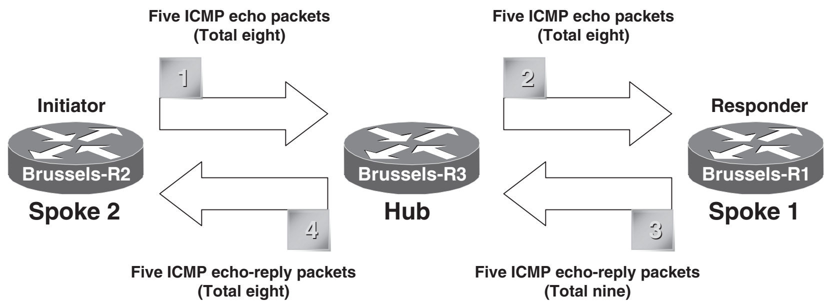


**Figure 5.3:** Traffic flow for ping test #2.

```
Source address or interface: 192.168.2.2
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/13/16 ms
```

Code Listing 5.10 illustrates the results for ping test #2 conducted in Code Listing 5.9. In the command traces below, italic comments are inserted at specific "checkpoints" for description purposes. We shall examine the traces from Brussels-R2, Brussels-R3, and Brussels-R1 in that order. The traffic flow for ping test #2 is illustrated in Figure 5.3.

Code Listing 5.10: Results for Ping Test #2

*We first examine the respective command outputs at Brussels-R2. From the "show crypto ipsec sa" command at Brussels-R2 (spoke 2), for the IPSec tunnel created between spoke 2 and hub, we can gather that five outgoing IP (ICMP echo) packets to spoke 1 have been encrypted and digested, and five incoming IP (ICMP echo-reply) packets from Spoke 1 have been decrypted and verified.*

```
Brussels-R2#show crypto ipsec sa

<Output Omitted>
```

*Traffic flow is from local subnet 192.168.2.0/24 to remote subnet 192.168.1.0/24.*

```
local ident (addr/mask/prot/port): (192.168.2.0/255.255.255.0/0/0)
remote ident (addr/mask/prot/port): (192.168.1.0/255.255.255.0/0/0)
<Output Omitted>
```

*The cumulative total number of outgoing packets from spoke 2 to spoke 1 that are encrypted and digested stands at eight. Three of them were from the previous ping test #1 (see Code Listing 5.8). The remaining five were ICMP echo packets generated by spoke 2 in ping test #2 (see Code Listing 5.9). Refer to Figure 5.3 for the traffic flow illustration.*

```
#pkts encaps: 8, #pkts encrypt: 8, #pkts digest 8
```

*The cumulative total number of incoming packets from spoke 1 to spoke 2 that are decrypted and verified stands at eight. Three of them were from the previous ping test #1 (see Code Listing 5.8). The remaining five were the response (ICMP echo-reply) packets generated by spoke 1 in ping test #2 (see Code Listing 5.9). Refer to Figure 5.3 for the traffic flow illustration.*

```
#pkts decaps: 8, #pkts decrypt: 8, #pkts verify 8
<Output Omitted>
```

*The "show crypto engine connection active" command at Brussels-R2 concurs with the "show crypto ipsec sa" command reflecting a total of eight (3 + 5) outbound packets being encrypted and digested, and a total of eight (3 + 5) inbound packets being decrypted and verified.*

```
Brussels-R2#show crypto engine connection active

  ID Interface        IP-Address     State Algorithm           Encrypt   Decrypt
   1 FastEthernet0/0 172.20.20.2     set   HMAC_MD5+DES_56_CB        0         0
2000 FastEthernet0/0 172.20.20.2     set   HMAC_MD5+DES_56_CB        0         8
2001 FastEthernet0/0 172.20.20.2     set   HMAC_MD5+DES_56_CB        8         0
```

*Next, we examine the respective command outputs at Brussels-R3. From the "show crypto ipsec sa" command at Brussels-R3 (hub), for the IPSec tunnel created between hub and spoke 1, we can gather that five outgoing IP (ICMP echo) packets from spoke 2 to spoke 1 have been encrypted and digested, and five incoming IP (ICMP echo-reply) packets from spoke 1 to spoke 2 have been decrypted and verified. For the IPSec tunnel created between hub and spoke 2, we can gather that five outgoing IP (ICMP echo-reply) packets from spoke 1 to spoke 2 have been encrypted and digested, and five incoming IP (ICMP echo) packets from spoke 2 to spoke 1 have been decrypted and verified.*

```
Brussels-R3#show crypto ipsec sa

<Output Omitted>

Traffic flow is from local subnet 192.168.2.0/24 to remote subnet
192.168.1.0/24 for the IPSec tunnel between hub and spoke 1.
  local ident (addr/mask/prot/port): (192.168.2.0/255.255.255.0/0/0)
  remote ident (addr/mask/prot/port): (192.168.1.0/255.255.255.0/0/0)
  <Output Omitted>

The cumulative total number of outgoing packets from spoke 2 to spoke 1
that are encrypted and digested stands at eight. Three of them were from
the previous ping test #1 (see Code Listing 5.8). The remaining five were
the ICMP echo packets generated by spoke 2 in ping test #2 (see Code Listing
5.9). Refer to Figure 5.3 for the traffic flow illustration.
  #pkts encaps: 8, #pkts encrypt: 8, #pkts digest 8

The cumulative total number of incoming packets from spoke 1 to spoke 2 that
are decrypted and verified stands at nine. Four of them were from the pre-
vious ping test #1 (see Code Listing 5.8). The remaining five were the response
(ICMP echo-reply) packets generated by spoke 1 in ping test #2 (see Code
Listing 5.9). Refer to Figure 5.3 for the traffic flow illustration.
  #pkts decaps: 9, #pkts decrypt: 9, #pkts verify 9
  <Output Omitted>

Traffic flow is from local subnet 192.168.1.0/24 to remote subnet
192.168.2.0/24 for the IPSec tunnel between hub and spoke 2.
 local ident (addr/mask/prot/port): (192.168.1.0/255.255.255.0/0/0)
  remote ident (addr/mask/prot/port): (192.168.2.0/255.255.255.0/0/0)
  <Output Omitted>

The cumulative total number of outgoing packets from spoke 1 to spoke 2 that
are encrypted and digested stands at eight. Three of them were from the pre-
vious ping test #1 (see Code Listing 5.8). The remaining five were the response
(ICMP echo-reply) packets generated by spoke 1 in ping test #2 (see Code
Listing 5.9). Refer to Figure 5.3 for the traffic flow illustration.
  #pkts encaps: 8, #pkts encrypt: 8, #pkts digest 8

The cumulative total number of incoming packets from spoke 2 to spoke 1
that are decrypted and verified stands at eight. Three of them were from
the previous ping test #1 (see Code Listing 5.8). The remaining five were
the ICMP echo packets generated by spoke 2 in ping test #2 (see Code Listing
5.9). Refer to Figure 5.3 for the traffic flow illustration.
  #pkts decaps: 8, #pkts decrypt: 8, #pkts verify 8
  <Output Omitted>
```

*The "show crypto engine connection active" command at Brussels-R3 concurs with the "show crypto ipsec sa" command. For the IPSec tunnel between hub and spoke 1, the command reflects eight (3 + 5) outbound (ICMP echo) packets being encrypted and digested together with nine (4 + 5) inbound (ICMP echo-reply) packets being decrypted (and verified), and the connections IDs for the inbound and outbound IPSec SAs between hub and spoke 1 are 2000 and 2001, respectively.*

*For the IPSec tunnel between hub and spoke 2, the command reflects eight (3 + 5) outbound packets being encrypted and digested together with eight (3 + 5) inbound packets being decrypted and verified, and the connections IDs for the inbound and outbound IPSec SAs between hub and spoke 2 correspond with 2002 and 2003, respectively.*

**Brussels-R3#show crypto engine connection active**

```
  ID Interface       IP-Address     State Algorithm           Encrypt   Decrypt
   1 FastEthernet0/0 172.20.20.3    set   HMAC_MD5+DES_56_CB        0         0
   2 <none>          <none>         set   HMAC_MD5+DES_56_CB        0         0
2000 FastEthernet0/0 172.20.20.3    set   HMAC_MD5+DES_56_CB        0         9
2001 FastEthernet0/0 172.20.20.3    set   HMAC_MD5+DES_56_CB        8         0
2002 FastEthernet0/0 172.20.20.3    set   HMAC_MD5+DES_56_CB        0         8
2003 FastEthernet0/0 172.20.20.3    set   HMAC_MD5+DES_56_CB        8         0
```

*We now examine the respective command outputs at Brussels-R1. From the "show crypto ipsec sa" command at Brussels-R1 (spoke 1), for the IPSec tunnel created between spoke 1 and hub, we can gather that five outgoing IP (ICMP echo-reply) packets to spoke 2 have been encrypted and digested, and five incoming IP (ICMP echo) packets from spoke 2 have been decrypted and verified.*

**Brussels-R1#show crypto ipsec sa**

<Output Omitted>

*Traffic flow is from local subnet 192.168.1.0/24 to remote subnet 192.168.2.0/24.*

```
  local ident (addr/mask/prot/port): (192.168.1.0/255.255.255.0/0/0)
  remote ident (addr/mask/prot/port): (192.168.2.0/255.255.255.0/0/0)
  <Output Omitted>
```

*The cumulative total number of outgoing packets from spoke 1 to spoke 2 that are encrypted and digested stands at nine. Four of them were from the previous ping test #1 (see Code Listing 5.8). The remaining five were the response (ICMP echo-reply) packets generated by spoke 1 in ping test #2* *(see Code Listing 5.9). Refer to Figure 5.3 for the traffic flow illustration.*

```
  #pkts encaps: 9, #pkts encrypt: 9, #pkts digest 9
```

*The cumulative total number of incoming packets from spoke 2 to spoke 1 that are decrypted and verified stands at eight. Three of them were from the previous ping test #1 (see Code Listing 5.8). The remaining five were the ICMP echo packets generated by spoke 2 in ping test #2 (see Code Listing 5.9). Refer to Figure 5.3 for the traffic flow illustration.*

```
  #pkts decaps: 8, #pkts decrypt: 8, #pkts verify 8
  <Output Omitted>
```

*The "show crypto engine connection active" command at Brussels-R2 concurs with the "show crypto ipsec sa" command reflecting a total of nine (4 + 5) outbound packets being encrypted and digested, and a total of eight (3 + 5) inbound packets being decrypted and verified.*

```
Brussels-R1#show crypto engine connection active

 ID  Interface        IP-Address    State  Algorithm            Encrypt  Decrypt
   1 <none>           <none>        set    HMAC_MD5+DES_56_CB         0        0
2000 FastEthernet0/0  172.20.20.1   set    HMAC_MD5+DES_56_CB         0        8
2001 FastEthernet0/0  172.20.20.1   set    HMAC_MD5+DES_56_CB         9        0
```

Code Listing 5.11 illustrates the ping test disseminated from spoke 1 (192.168.1.1) to hub (192.168.3.3). In this instance, the test is only 80 percent successful, that is, 4 of 5 ping packets went through to Hub.

Code Listing 5.11: Ping test #3 from spoke 1 to hub

```
Brussels-R1#ping
Protocol [ip]:
Target IP address: 192.168.3.3
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 192.168.1.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
```

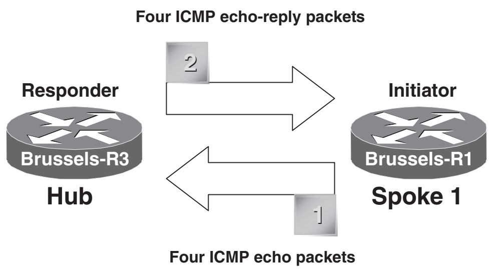


**Figure 5.4:** Traffic flow for ping test #3.

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.3.3, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 4/5/8 ms
```

Code Listing 5.12 illustrates the results for ping test #3 conducted in Code Listing 5.11. In the command traces below, italic comments are inserted at specific "checkpoints" for description purposes. We shall examine the traces for Brussels-R1 and Brussels-R3. The traffic flow for ping test #3 is illustrated in Figure 5.4.

Code Listing 5.12: Results for Ping Test #3

*We first examine the respective command outputs at Brussels-R1. From the "show crypto ipsec sa" command at Brussels-R1 (spoke 1), for the IPSec tunnel created between spoke 1 and hub, we can gather that four of five outgoing IP (ICMP echo) packets to hub have been encrypted and digested, and four incoming IP (ICMP echo-reply) packets from hub have been decrypted and verified.*

```
Brussels-R1#show crypto ipsec sa

<Output Omitted>
```

*Traffic flow is from local subnet 192.168.1.0/24 to remote subnet 192.168.3.0/24.*

```
  local ident (addr/mask/prot/port): (192.168.1.0/255.255.255.0/0/0)
  remote ident (addr/mask/prot/port): (192.168.3.0/255.255.255.0/0/0)
  <Output Omitted>
```

*Four outgoing (ICMP echo) packets from spoke 1 to hub are encrypted and digested (see Fig. 5.4).*

```
  #pkts encaps: 4, #pkts encrypt: 4, #pkts digest 4
```

*Four incoming (ICMP echo-reply) packets from hub are decrypted and verified (see Fig. 5.4).*

```
#pkts decaps: 4, #pkts decrypt: 4, #pkts verify 4
<Output Omitted>
```

*At spoke 1, of the five packets sent, one is bad.*

```
#send errors 1, #recv errors 0
```

*The IPSec tunnel originates from spoke 1 (172.20.20.1) and terminates at hub (172.20.20.3).*

```
local crypto endpt.: 172.20.20.1, remote crypto endpt.: 172.20.20.3
<Output Omitted>
```

*For the inbound IPSec SA from hub to spoke 1, the SPI is 4206479133 and the connection ID is 2002.*

```
inbound esp sas:
spi: 0xFAB9C71D(4206479133)
  transform: esp-des esp-md5-hmac ,
  in use settings ={Tunnel, }
  slot: 0, conn id: 2002, flow_id: 3, crypto map: chapter5
  <Output Omitted>
```

*For the outbound IPSec SA from spoke 1 to hub, the SPI is 2929535312 and the connection ID is 2003.*

```
outbound esp sas:
spi: 0xAE9D2950(2929535312)
  transform: esp-des esp-md5-hmac ,
  in use settings ={Tunnel, }
  slot: 0, conn id: 2003, flow_id: 4, crypto map: chapter5
  <Output Omitted>
```

*The "show crypto engine connection active" command at Brussels-R1 concurs with the "show crypto ipsec sa" command reflecting four outbound (ICMP echo) packets to hub being encrypted and digested, and four inbound (ICMP echo-reply) packets from hub being decrypted and verified. The connections IDs for the inbound and outbound IPSec SAs are 2002 and 2003, respectively, which correspond to the same connection ID values illustrated earlier in the "show crypto ipsec sa" command.*

```
Brussels-R1#show crypto engine connection active

  ID Interface        IP-Address    State Algorithm           Encrypt   Decrypt
   1 <none>           <none>        set   HMAC_MD5+DES_56_CB        0         0
2000 FastEthernet0/0  172.20.20.1   set   HMAC_MD5+DES_56_CB        0         8
2001 FastEthernet0/0  172.20.20.1   set   HMAC_MD5+DES_56_CB        9         0
```

```
2002 FastEthernet0/0 172.20.20.1   set   HMAC_MD5+DES_56_CB       0      4
2003 FastEthernet0/0 172.20.20.1   set   HMAC_MD5+DES_56_CB       4      0
```

*We now examine the respective command outputs at Brussels-R3. From the "show crypto ipsec sa" command at Brussels-R3 (hub), for the IPSec tunnel created between hub and spoke 1, we can gather that four outgoing IP (ICMP echo-reply) packets to spoke 1 have been encrypted and digested, and four incoming IP (ICMP echo) packets from spoke 1 have been decrypted and verified.*

```
Brussels-R3#show crypto ipsec sa

<Output Omitted>
```

*Traffic flow is from local subnet 192.168.3.0/24 to remote subnet 192.168.1.0/24.*

```
  local ident (addr/mask/prot/port): (192.168.3.0/255.255.255.0/0/0)
  remote ident (addr/mask/prot/port): (192.168.1.0/255.255.255.0/0/0)
  <Output Omitted>
```

*Four outgoing (ICMP echo-reply) packets from hub to spoke 1 are encrypted and digested (see Fig. 5.4).*

```
  #pkts encaps: 4, #pkts encrypt: 4, #pkts digest 4
```

*Four incoming (ICMP echo) packets from spoke 1 are decrypted and verified (see Fig. 5.4).*

```
  #pkts decaps: 4, #pkts decrypt: 4, #pkts verify 4
  <Output Omitted>
```

*The IPSec tunnel originates from hub (172.20.20.3) and terminates at spoke 1 (172.20.20.1).*

```
   local crypto endpt.: 172.20.20.3, remote crypto endpt.: 172.20.20.1
   <Output Omitted>
```

*For the inbound IPSec SA from spoke 1 to hub, the SPI is 2929535312 and the connection ID is 2004.*

```
   inbound esp sas:
   spi: 0xAE9D2950(2929535312)
     transform: esp-des esp-md5-hmac ,
     in use settings ={Tunnel, }
     slot: 0, conn id: 2004, flow_id: 5, crypto map: chapter5
     <Output Omitted>
```

*For the outbound IPSec SA from Hub to Spoke 1, the SPI is 4206479133 and the connection ID is 2005.*

```
   outbound esp sas:
   spi: 0xFAB9C71D(4206479133)
```

```
transform: esp-des esp-md5-hmac ,
in use settings ={Tunnel, }
slot: 0, conn id: 2005, flow_id: 6, crypto map: chapter5
<Output Omitted>
```

*The "show crypto engine connection active" command at Brussels-R3 concurs with the "show crypto ipsec sa" command reflecting four outbound (ICMP echo-reply) packets to spoke 1 being encrypted and digested, and four inbound (ICMP echo) packets from spoke 1 being decrypted and verified. The connections IDs for the inbound and outbound IPSec SAs are 2004 and 2005, respectively, which correspond to the same connection ID values illustrated earlier in the "show crypto ipsec sa" command.*

```
Brussels-R3#show crypto engine connection active

  ID Interface       IP-Address     State Algorithm           Encrypt   Decrypt
   1 FastEthernet0/0 172.20.20.3    set   HMAC_MD5+DES_56_CB        0         0
   2 <none>          <none>         set   HMAC_MD5+DES_56_CB        0         0
2000 FastEthernet0/0 172.20.20.3    set   HMAC_MD5+DES_56_CB        0         9
2001 FastEthernet0/0 172.20.20.3    set   HMAC_MD5+DES_56_CB        8         0
2002 FastEthernet0/0 172.20.20.3    set   HMAC_MD5+DES_56_CB        0         8
2003 FastEthernet0/0 172.20.20.3    set   HMAC_MD5+DES_56_CB        8         0
2004 FastEthernet0/0 172.20.20.3    set   HMAC_MD5+DES_56_CB        0         4
2005 FastEthernet0/0 172.20.20.3    set   HMAC_MD5+DES_56_CB        4         0
```

Code Listing 5.13 illustrates the ping test disseminated from spoke 2 (192.168.2.2) to hub (192.168.3.3). In this instance, the test is only 80 percent successful, that is, 4 of 5 ping packets went through to hub.

Code Listing 5.13: Ping test #4 from spoke 2 to hub

```
Brussels-R2#ping
Protocol [ip]:
Target IP address: 192.168.3.3
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 192.168.2.2
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
```

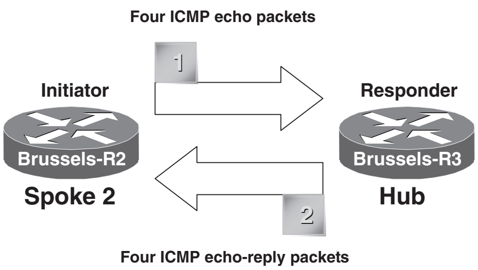


**Figure 5.5:** Traffic flow for ping test #4.

```
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.3.3, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 4/5/8 ms
```

Code Listing 5.14 illustrates the results for ping test #4 conducted in Code Listing 5.13. In the command traces below, italic comments are inserted at specific "checkpoints" for description purposes. We shall examine the traces for Brussels-R2 and Brussels-R3. The traffic flow for ping test #4 is illustrated in Figure 5.5.

Code Listing 5.14: Results for Ping Test #4

*We first examine the respective command outputs at Brussels-R2. From the "show crypto ipsec sa" command at Brussels-R2 (spoke 2), for the IPSec tunnel created between spoke 2 and hub, we can gather that four of five outgoing IP (ICMP echo) packets to hub have been encrypted and digested, and four incoming IP (ICMP echo-reply) packets from hub have been decrypted and verified.*

```
Brussels-R2#show crypto ipsec sa

<Output Omitted>
```

*Traffic flow is from local subnet 192.168.2.0/24 to remote subnet 192.168.3.0/24.*

```
  local ident (addr/mask/prot/port): (192.168.2.0/255.255.255.0/0/0)
  remote ident (addr/mask/prot/port): (192.168.3.0/255.255.255.0/0/0)
  <Output Omitted>
```

*Four outgoing (ICMP echo) packets from spoke 2 to hub are encrypted and digested (see Fig. 5.5).*

```
  #pkts encaps: 4, #pkts encrypt: 4, #pkts digest 4
```

*Four incoming (ICMP echo-reply) packets from hub are decrypted and verified (see Fig. 5.5).*

```
  #pkts decaps: 4, #pkts decrypt: 4, #pkts verify 4
  <Output Omitted>
```

*At Spoke 2, of the five packets sent, one is bad.*

```
  #send errors 1, #recv errors 0
```

*The IPSec tunnel originates from spoke 2 (172.20.20.2) and terminates at Hub (172.20.20.3).*

```
    local crypto endpt.: 172.20.20.2, remote crypto endpt.: 172.20.20.3
    <Output Omitted>
```

*For the inbound IPSec SA from hub to spoke 2, the SPI is 2266443490 and the connection ID is 2002.*

```
    inbound esp sas:
    spi: 0x87172EE2(2266443490)
      transform: esp-des esp-md5-hmac ,
      in use settings ={Tunnel, }
      slot: 0, conn id: 2002, flow_id: 3, crypto map: chapter5
      <Output Omitted>
```

*For the outbound IPSec SA from spoke 2 to hub, the SPI is 618541355 and the connection ID is 2003.*

```
    outbound esp sas:
    spi: 0x24DE312B(618541355)
      transform: esp-des esp-md5-hmac ,
      in use settings ={Tunnel, }
      slot: 0, conn id: 2003, flow_id: 4, crypto map: chapter5
      <Output Omitted>
```

*The "show crypto engine connection active" command at Brussels-R2 concurs with the "show crypto ipsec sa" command reflecting four outbound (ICMP echo) packets to hub being encrypted and digested, and four inbound (ICMP echo-reply) packets from Hub being decrypted and verified. The connections IDs for the inbound and outbound IPSec SAs are 2002 and 2003, respectively, which correspond to the same connection ID values illustrated earlier in the "show crypto ipsec sa" command.*

```
Brussels-R2#show crypto engine connection active

  ID Interface        IP-Address     State  Algorithm          Encrypt   Decrypt
   1 FastEthernet0/0  172.20.20.2    set    HMAC_MD5+DES_56_CB       0         0
2000 FastEthernet0/0  172.20.20.2    set    HMAC_MD5+DES_56_CB       0         8
2001 FastEthernet0/0  172.20.20.2    set    HMAC_MD5+DES_56_CB       8         0
```

```
2002 FastEthernet0/0 172.20.20.2    set    HMAC_MD5+DES_56_CB        0          4
2003 FastEthernet0/0 172.20.20.2    set    HMAC_MD5+DES_56_CB        4          0
```

*we now examine the respective command outputs at Brussels-R3. From the "show crypto ipsec sa" command at Brussels-R3 (hub), for the IPSec tunnel created between hub and spoke 2, we can gather that four outgoing IP (ICMP echo-reply) packets to spoke 2 have been encrypted and digested, and four incoming IP (ICMP echo) packets from spoke 2 have been decrypted and verified.*

```
Brussels-R3#show crypto ipsec sa

<Output Omitted>
```

*Traffic flow is from local subnet 192.168.3.0/24 to remote subnet 192.168.2.0/24.*

```
  local ident (addr/mask/prot/port): (192.168.3.0/255.255.255.0/0/0)
  remote ident (addr/mask/prot/port): (192.168.2.0/255.255.255.0/0/0)
  <Output Omitted>
```

*Four outgoing (ICMP echo-reply) packets from hub to spoke 2 are encrypted and digested (see Fig. 5.5).*

```
  #pkts encaps: 4, #pkts encrypt: 4, #pkts digest 4
```

*Four incoming (ICMP echo) packets from spoke 2 are decrypted and verified (see Fig. 5.5).*

```
  #pkts decaps: 4, #pkts decrypt: 4, #pkts verify 4
  <Output Omitted>
```

*The IPSec tunnel originates from hub (172.20.20.3) and terminates at spoke 2 (172.20.20.2).*

```
   local crypto endpt.: 172.20.20.3, remote crypto endpt.: 172.20.20.2
   <Output Omitted>
```

*For the inbound IPSec SA from spoke 2 to hub, the SPI is 618541355 and the connection ID is 2006.*

```
   inbound esp sas:
   spi: 0x24DE312B(618541355)
     transform: esp-des esp-md5-hmac ,
     in use settings ={Tunnel, }
     slot: 0, conn id: 2006, flow_id: 7, crypto map: chapter5
     <Output Omitted>
```

*For the outbound IPSec SA from hub to spoke 2, the SPI is 2266443490 and the connection ID is 2007.*

```
   outbound esp sas:
   spi: 0x87172EE2(2266443490)
```

```
transform: esp-des esp-md5-hmac ,
in use settings ={Tunnel, }
slot: 0, conn id: 2007, flow_id: 8, crypto map: chapter5
<Output Omitted>
```

*The "show crypto engine connection active" command at Brussels-R3 concurs with the "show crypto ipsec sa" command reflecting four outbound (ICMP echo-reply) packets to spoke 2 being encrypted and digested, and four inbound (ICMP echo) packets from spoke 2 being decrypted and verified. The connections IDs for the inbound and outbound IPSec SAs are 2006 and 2007, respectively, which correspond to the same connection ID values illustrated earlier in the "show crypto ipsec sa" command.*

```
Brussels-R3#show crypto engine connection active

  ID  Interface        IP-Address     State  Algorithm           Encrypt   Decrypt
   1  FastEthernet0/0  172.20.20.3    set    HMAC_MD5+DES_56_CB        0         0
   2  <none>           <none>         set    HMAC_MD5+DES_56_CB        0         0
2000  FastEthernet0/0  172.20.20.3    set    HMAC_MD5+DES_56_CB        0         9
2001  FastEthernet0/0  172.20.20.3    set    HMAC_MD5+DES_56_CB        8         0
2002  FastEthernet0/0  172.20.20.3    set    HMAC_MD5+DES_56_CB        0         8
2003  FastEthernet0/0  172.20.20.3    set    HMAC_MD5+DES_56_CB        8         0
2004  FastEthernet0/0  172.20.20.3    set    HMAC_MD5+DES_56_CB        0         4
2005  FastEthernet0/0  172.20.20.3    set    HMAC_MD5+DES_56_CB        4         0
2006  FastEthernet0/0  172.20.20.3    set    HMAC_MD5+DES_56_CB        0         4
2007  FastEthernet0/0  172.20.20.3    set    HMAC_MD5+DES_56_CB        4         0
```

## 5.4 Scaling IPSec VPNs

Typically, there are two ways to scale IPSec in large networks. Tunnel endpoint discovery (TED) can be used to scale partially or fully meshed networks, and the multihop crypto (or encryption) model can be used to scale existing hub-and-spoke networks. In the following sections, we briefly discuss these two IPSec scaling techniques.

### 5.4.1 Meshed networks and tunnel endpoint discovery

Tunnel endpoint discovery (see Chapter 3, Section 3.7 for details) allows IPSec to scale to large networks by reducing multiple encryptions, decreasing the setup time, and allowing for simpler configurations on participating peer routers. Each node requires only a straightforward configuration that defines the local network that the router protects and the required IPSec transforms. The limitation of the dynamic crypto map is that the initiating router cannot dynamically determine an IPSec peer; instead, only the receiving router has this ability. However, by defining a dynamic crypto map together with TED, the

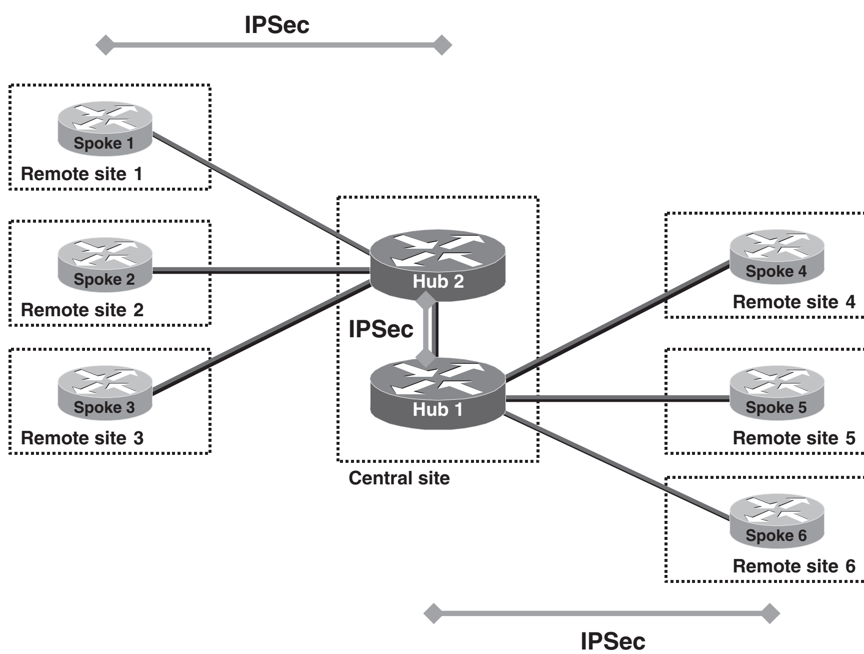


**Figure 5.6:** The Multi-hop crypto model.

initiating router can dynamically determine an IPSec peer for secure IPSec communications. In other words, TED provides automated peer discovery, which eases scalability and configuration, as well as enhances availability by automating the failover IPSec peer switch. TED mechanisms work best in partially or fully meshed networks that require spoke-to-spoke connectivity on an occasional basis.

### 5.4.2 Hub-and-spoke networks and multihop crypto

The multihop crypto (encryption) model is another alternative that can be deployed to scale IPSec VPNs. The multihop model is implemented by combining multiple hub-and-spoke IPSec networks together as illustrated in Figure 5.6. Multihop crypto allows the distribution of IPSec traffic loads across multiple central site (hub) routers and requires separate subnets and crypto maps for each interface on the central site routers.

With multihop crypto, instead of the any-to-any IPSec communication with every router, IPSec communication is limited to between the hub-and-spoke routers, thus improving the network scalability significantly. For any-to-any (fully meshed) IPSec implementation, the configurations at every router can become lengthy and complex. In the multihop crypto model, configurations are fairly simple for the (spoke) routers at the remote sites; however, the configurations for the (hub) routers at the central site can become rather complex and difficult to maintain. In other words, the multihop crypto model inherited all the pros and cons of a hub-and-spoke implementation as discussed in Section 5.2.

## 5.5 Summary

Chapter 5 gives the reader hands-on skills for implementing IPSec VPN in large enterprise networks using a hub-and-spoke topology. The chapter first gives an overview of meshed versus hub-and-spoke networks. Case Study 5.1 further substantiates the scalability of a hub-and-spoke topology with an IPSec hub-and-spoke VPN design and implementation. The concluding section of the chapter discusses how the tunnel endpoint discovery (TED) mechanism and the multihop crypto model can be used to scale existing meshed and hub-and-spoke networks, respectively.