An Introduction to IP Security (IPSec) Encryption

Contents:

Warning

There are severe restrictions on the export of "strong" cryptography. If you violate U.S. Federal Law, then you, not Cisco, will be held accountable! If you have any questions regarding export control, please send email to export@cisco.com.

Purpose

The purpose of this document is to introduce IPSec to users in a rapid, but concise format. You will see basic configurations of Internet Key Exchange (IKE) with pre-shared keys, IKE with a Certification Authority, and IPSec. This is not an exhaustive document. This document will, however, help you to understand the tasks and the order in which they are accomplished.

Return to top *Return to table of contents* *Go to next section*

Home	What's New	How to Buy	Login	Register	Feedback	Search	Map/Help

An Introduction to IP Security (IPSec) Encryption

Background

IPSec is the next-generation network layer crypto platform for Cisco's security platforms (Cisco IOS® Software, PIX, and so on). Originally described in RFCs 1825-1829, which are now obsolete, IPSec is currently discussed in a number of documents presented by the IETF IP Security Working Group . IPSec currently supports IP version 4 unicast packets. IPv6 and multicast support is coming later.

IPSec has the following strengths over current Cisco crypto offerings:

1. **Multivendor:** Since the IPSec framework is standardized, customers are not locked into any specific vendor's product. You will find IPSec on routers, firewalls, and client desktops (Windows, Mac, and so on).
2. **Scalability:** IPSec was designed with large enterprises in mind and therefore, it has "built-in" key management.

Note: While several Cisco platforms can use IPSec, this document is geared towards Cisco IOS Software.

Return to previous section *Return to table of contents* *Go to next section*

An Introduction to IP Security (IPSec) Encryption

Crypto Lingo (Vocabulary)

You need to know the following terms to understand IPSec, and to read the rest of this document. When you see acronyms in other portions of this document, refer to this page for definitions.

Authentication Header (AH): A security protocol that provides authentication and optional replay-detection services. AH is embedded in the data to be protected (a full IP datagram, for example). AH can be used either by itself or with Encryption Service Payload (ESP). Refer to the RFC 2402.

Authentication: One of the functions of the IPSec framework. Authentication establishes the integrity of datastream and ensures that it is not tampered with in transit. It also provides confirmation about datastream origin.

Certification Authority (CA): A third-party entity that is responsible for issuing and revoking certificates. Each device that has its own certificate and public key of the CA can authenticate every other device within a given CA's domain. This term is also applied to server software that provides these services.

Certificate: A cryptographically signed object that contains an identity and a public key associated with this identity.

Classic crypto: Cisco proprietary encryption mechanism used in Cisco IOS release 11.2. Classic crypto will be available in Cisco IOS release 11.3, however, IPSec will not be "retrofitted" to Cisco IOS release 11.2. You may also see the name "classic crypto" referred to as "Encryption Express" or "Cisco Encryption Technology" (CET) in the marketing literature.

Certificate Revocation List (CRL): A digitally signed message that lists all of the current but revoked certificates listed by a given CA. This is analogous to a book of stolen charge card numbers that allow stores to reject bad credit cards.

Crypto map: A Cisco IOS software configuration entity that performs two primary functions: (1)it selects data flows that need security processing and (2)defines the policy for these flows and the crypto peer that traffic needs to go to.

A crypto map is applied to an interface. The concept of a crypto map was introduced in classic crypto but was expanded for IPSec.

Data integrity: Data integrity mechanisms, through the use of secret-key based or public-key based algorithms, that allow the recipient of a piece of protected data to verify that the data has not been modified in transit.

Data confidentiality: Method where protected data is manipulated so that no attacker can read it. This is commonly provided by data encryption and keys that are only available to the parties involved in the communication.

Data origin authentication: A security service where the receiver can verify that protected data could have originated only from the sender. This service requires a data integrity service plus a key distribution mechanism, where a secret key is shared only between the sender and receiver.

Data Encryption Standard (DES): The DES was published in 1977 by the National Bureau of Standards and is a secret key encryption scheme based on the Lucifer algorithm from IBM. The contrast of DES is public-key. Cisco uses DES in classic crypto (40-bit and 56-bit key lengths), IPSec crypto (56-bit key), and on the PIX Firewall (56-bit key).

Diffie-Hellman: A method of establishing a shared key over an insecure medium. Diffie-Hellman is a component of Oakley.

DSS: A digital signature algorithm designed by The US National Institute of Standards and Technology (NIST) based on public key cryptography. DSS doesn't do user datagram encryption. DSS is a component in classic crypto, as well as the Redcreek IPSec card, but not in IPSec implemented in Cisco IOS software.

Encryption Service Adapter (ESA): A hardware based encryption accelerator that is used in:

- Cisco 7204 and 7206 routers
- Second-generation Versatile Interface Processor2-40s (VIP2-40s) in all Cisco 7500 series routers
- VIP2-40 in the Cisco 7000 series routers that have the Cisco 7000 series Route Switch Processor (RSP7000) and Cisco 7000 series Chassis Interface (RSP7000CI) cards installed.

IPSec does not use the ESA acceleration, but will work in a box that has an ESA card on a software-only basis.

Encapsulating Security Payload (ESP): A security protocol that provides data confidentiality and protection with optional authentication and replay-detection services. ESP completely encapsulates user data. ESP can be used either by itself or in conjunction with AH. Check out RFC 2406: IP Encapsulating Security Payload (ESP).

Hash: A one way function that takes an input message of arbitrary length and produces a fixed length digest. Cisco uses both Secure Hash Algorithm (SHA) and Message Digest 5 (MD5) hashes within our implementation of the IPSec framework (see HMAC below).

HMAC: A mechanism for message authentication using cryptographic hashes such as SHA and MD5. For an exhaustive discussion of HMAC, check out RFC 2104.

Internet Key Exchange (IKE): A hybrid protocol that uses part Oakley and part of another protocol suite called SKEME inside the Internet Security Association and Key Management Protocol (ISAKMP) framework. IKE is used to establish a shared security policy and authenticated keys for services (such as IPSec) that require keys. Before any IPSec traffic can be passed, each router/firewall/host must be able to verify the identity of its peer. This can be done by manually entering pre-shared keys into both hosts, by a CA service, or the forthcoming secure DNS (DNSSec). This is the protocol formerly known as ISAKMP/Oakley, and is defined in RFC 2409: The Internet Key Exchange (IKE). A potential point of confusion is that the acronyms "ISAKMP" and "IKE" are both used in Cisco IOS software to refer to the same thing. These two items are somewhat different, as you will see in the next definition.

Internet Security Association and Key Management Protocol (ISAKMP): A protocol framework that defines the mechanics of implementing a key exchange protocol and negotiation of a security policy. ISAKMP is defined in the Internet Security Association and Key Management Protocol (ISAKMP).

ISAKMP/Oakley: See IKE.

Message Digest 5 (MD5): A one way hashing algorithm that produces a 128-bit hash. Both MD5 and Secure Hash Algorithm (SHA) are variations on MD4, which is designed to strengthen the security of this hashing algorithm. SHA is more secure than MD4 and MD5. Cisco uses hashes for authentication within the IPSec framework.

Oakley: A key exchange protocol that defines how to acquire authenticated keying material. The basic mechanism for Oakley is the Diffie-Hellman key exchange algorithm. You can find the standard in RFC 2412: The OAKLEY Key Determination Protocol.

Perfect Forward Secrecy (PFS): PFS ensures that a given IPSec SA's key was not derived from any other secret (like some other keys). In other words, if someone were to breaks a key, PFS ensures that the attacker would not be able to derive any other key. If PFS were not enabled, someone could hypothetically break the IKE SA secret key, copy all the IPSec protected data, and then use knowledge of the IKE SA secret to compromise the IPSec SAs setup by this IKE SA. With PFS, breaking IKE would not give an attacker immediate access to IPSec. The attacker would have to break each IPSec SA individually. Cisco's IOS IPSec implementation uses PFS group 1 (D-H 768 bit) by default.

Replay-detection: A security service where the receiver can reject old or duplicate packets in order to defeat replay attacks (replay attacks rely on the attacker sending out older or duplicate packets to the receiver and the receiver thinking that the bogus traffic is legitimate). Replay-detection is done by using sequence numbers combined with authentication, and is a standard feature of IPSec.

RSA: A public key cryptographic algorithm (named after its inventors, Rivest, Shamir and Adleman) with a variable key length. RSA's main weakness is that it is significantly slow to compute compared to popular secret-key algorithms, such as DES. Cisco's IKE implementation uses a Diffie-Hellman exchange to get the secret keys. This exchange can be authenticated with RSA (or pre-shared keys). With the Diffie-Hellman exchange, the DES key never crosses the network (not even in encrypted form), which is not the case with the RSA encrypt and sign technique. RSA is not public domain, and must be licensed from RSA Data Security.

Security Association (SA): An instance of security policy and keying material applied to a data flow. Both IKE and IPSec use SAs, although SAs are independent of one another. IPSec SAs are *unidirectional* and they are unique in each security protocol. A set of SAs are needed for a protected data pipe, one per direction per protocol. For example, if you have a pipe that supports ESP between peers, one ESP SA is required for each direction. SAs are uniquely identified by destination (IPSec endpoint) address, security protocol (AH or ESP), and security parameter index (SPI).

IKE negotiates and establishes SAs on behalf of IPSec. A user can also establish IPSec SAs manually.

An IKE SA is used by IKE only, and unlike the IPSec SA, it is *bi-directional*.

Secure Hash Algorithm (SHA): A one way hash put forth by NIST. SHA is closely modeled after MD4 and produces a 160-bit digest. Because SHA produces a 160-bit digest, it is more resistant to brute-force attacks than 128-bit hashes (such as MD5), but it is slower.

Transform: A transform describes a security protocol (AH or ESP) with its corresponding algorithms. For example, ESP with the DES cipher algorithm and HMAC-SHA for authentication.

Transport Mode: An encapsulation mode for AH/ESP. Transport Mode encapsulates the upper layer payload (such as Transmission Control Protocol (TCP) or User Datagram Protocol (UDP)) of the original IP datagram. This mode can only be used when the peers are the endpoints of the communication. The contrast of Transport Mode is Tunnel Mode.

Tunnel Mode: Encapsulation of the complete IP Datagram for IPSec. Tunnel Mode is used to protect datagrams sourced from or destined to non-IPSec systems (such as in a Virtual Private Network (VPN) scenario).

Return to previous section   *Return to table of contents*   *Go to next section*

Home | What's New | How to Buy | Login | Register | Feedback | Search | Map/Help

An Introduction to IP Security (IPSec) Encryption

Configuring ISAKMP

IKE exists only to establish SAs for IPSec, but before it can do this, IKE must negotiate an SA (an ISAKMP SA) relationship with the peer. Because IKE negotiates its own policy, it is possible to configure multiple policy statements with different configuration statements, then let the two hosts come to an agreement. ISAKMP will negotiate the following:

- **An Encryption Algorithm:** this is limited to 56-bit DES only
- **A Hashing Algorithm:** MD5 or SHA
- **Authentication:** RSA signatures, RSA Encrypted nonces (random numbers"), or pre-shared keys
- **Lifetime of the SA:** in seconds

Currently, there are three methods of configuring ISAKMP:

1. Use **pre-shared keys**, which have the advantage of being simple to configure.
2. Use a **CA**, which has the advantage of being scaleable throughout the Enterprise.

The third method uses DNS Secure (DNSSec), but this document will not discuss DNSSec.

A note about access lists: IKE negotiation is done on UDP 500. IPSec uses IP protocols 50 and 51. Make sure these are permitted on any access lists you have between the peers.

1. Pre-Shared Keys

This is the "quick and dirty" method of configuring IKE. While configuring IKE is simple and you don't use a CA, it doesn't scale very well.

To configfure IKE, you need to:

- **Configure ISAKMP protection suite(s)**
- **Configure ISAKMP key**

Configure ISAKMP Protection Suite(s)

The command below creates the ISAKMP policy object. It is possible to have multiple policies, but there is only one in this example, which is:

```
dt3-45a(config)#crypto isakmp policy 1

dt3-45a(config-isakmp)#
```

With the **group** command below, you can declare what size modulus to use for Diffie-Hellman calculation. Group 1 is 768 bits long, and group 2 is 1024 bits long. Why would you use one over the other? First of all, not all vendors support group 2. Second, group 2 is also significantly more CPU intensive than group one, and therefore, you wouldn"t want to use group 2 on low-end routers like the Cisco 2500 series or less. On the other hand, group 2 is more secure than group 1. Since this example uses a Cisco 4500, we'll use group 2 (make sure the peer is also configured to use group 2). The default is group 1. If you select the default properties, the group 1 lines won't show up when you do a **write term** command.

```
dt3-45a(config-isakmp)#group 2
```

MD5 is our hashing algorithm in the line below. While implementing SHA and MD5 are both mandatory, not all peers may be configured to negotiate one or the other. The default in Cisco IOS is SHA, which is more secure than MD5.

```
dt3-45a(config-isakmp)#hash md5
```

The lifetime of the SA, 500 seconds in this case, is shown in the command below. If you do not set a lifetime, it defaults to 86400 seconds, or one day. When the lifetime timer fires, the SA is renegotiated as a security measure.

```
dt3-45a(config-isakmp)#lifetime 500
```

In the command below, we manually tell IKE what key to use, therefore, we use the **pre-share** command. Two options besides the **pre-share** command are the **rsa-encr** and the **rsa-sig** commands. The **rsa-encr** command configures RSA Encrypted nonces and the **rsa-sig** command configures RSA Signature. The **rsa-encr** and the **rsa-sig** commands are addressed in the Using a CA section. For now, remember that **rsa-sig** is the default.

```
dt3-45a(config-isakmp)#authentication pre-share
```

Configure ISAKMP key

In the commands below, we tell IKE what key to use. Remember that the peer, 192.168.10.38 in this case, must have the same key "Slurpee-Machine" in its configuration.

```
dt3-45a(config-isakmp)#exit
dt3-45a(config)#crypto isakmp key Slurpee-Machine address 192.168.10.38
```

At this point, you're done with IKE configuration. For the record, the following lines are the peer's IKE configuration (the complete configurations for both routers are in the Sample Configurations section of this document):

```
crypto isakmp policy 1
 hash md5
 group 2
 authentication pre-share
crypto isakmp key Slurpee-Machine address 192.168.10.66
```

2. Using a CA

Using a CA is a complex method of configuring IKE, but because it is very scaleable in IPSec, you should use IPSec instead of classic crypto. When Cisco IOS release 11.3(3) hits the streets, there are only going to be a few CA vendors actually shipping product, so initially most configurations will be done using **pre-shared** keys. VeriSign, Entrust, Microsoft and Netscape (and probably a host of others) are working on CA products, but in this example, we will assume a VeriSign CA.

To use a CA, you need to:

- **Create RSA key-pair(s) for the router**
- **Request CA's certificate**
- **Enroll certificates for the client router**
- **Configure ISAKMP protection suite(s)**

Create RSA key-pairs for the router

The **#cry key gen rsa usage-keys** command below may be somewhat confusing. This command creates two key-pairs for RSA: one key-pair for encryption and one key-pair for digital signatures. A key-pair refers to a public key and its corresponding secret key. If you do not specify "usage-keys" at the end of the command, the router will generate only one RSA key-pair and use it for both encryption and digital signatures. Be warned, however, that this command can be used to create DSS keys as well, but DSS is a part of classic crypto, not IPSec.

```
dt3-45a(config)#cry key gen rsa usage-keys
The name for the keys will be: dt3-45a.cisco.com
```

```
%You already have RSA keys defined for dt3-45a.cisco.com.
%Do you really want to replace them? [yes/no] yes
```

Since we already had some RSA keys on this box to begin with, it asks us if we want to get rid of the existing keys. We do, so we confirm the command. Which returns the following prompt:

```
Choose the size of the key modulus in the range of 360 to 2048 for your Signature
keys.  Choosing a key modulus greater than 512 may take a few minutes.

How many bits in the modulus [512]: <return>
Generating RSA keys...
[OK]
Choose the size of the key modulus in the range of 360 to 2048 for your Encryption
keys.  Choosing a key modulus greater than 512 may take a few minutes.

How many bits in the modulus [512]: <return>
Generating RSA keys...
[OK]
dt3-45a(config)#
```

We've now created RSA key pairs with the default 512-bit modulus. That wasn't so painful, was it? Exit out of config mode now and do a **sho cry key mypubkey rsa** command, you should now see your RSA public key(s) (we never see the private key portion of the key pair). Even if you didn't have pre-existing keys, you would see the same thing above.

Reminder: save your configuration once you've generated your key-pairs!

Request CA's Certificate

Now we need to configure the router to talk to a CA. This involves several steps and eventually you will need to coordinate with your CA administrator.

In the config lines below, we are adding a domain name to the router, creating a hostname **ciscoca-ultra**, and telling the router what its IP address is going to be (and the name servers, of course). You should have either hostnames defined for the CA or DNS working on the box. We recommend that you have DNS working on the box.

```
dt3-45a(config)#ip host ciscoca-ultra 171.69.54.46
dt3-45a(config)#ip domain-name cisco.com
dt3-45a(config)#ip name-server 171.692.132
dt3-45a(config)#ip name-server 198.92.30.32
```

Now we start to configure the CA parameters. **verisign-ca** is just an arbitrary name.

```
dt3-45a(config)#cry ca identity verisign-ca
dt3-45a(ca-identity)#
```

Below, Cisco's enrollment protocol uses HTTP to talk to the CA. The **dt3-45a(ca-identity)#enrollment url http://ciscoca-ultra** command tells the router to go to the specified URL to interact with the CA. The **dt3-45a(ca-identity)#crypto ca authenticate verisign-ca** command instructs the router to fetch the CA's own certificate. Before you can enroll in the CA, you need to make sure you're talking to the real CA. Verify the CA's certificate with the CA administrator to ensure authenticity.

```
dt3-45a(ca-identity)#enrollment url http://ciscoca-ultra
dt3-45a(ca-identity)#ex
dt3-45a(ca-identity)#crypto ca authenticate verisign-ca
```

Enroll Certificates for the Client Router

Issue the command to begin enrollment with the CA, as we've done below. There are several steps to this. First, we have to verify the identity of the CA, then the CA has to verify the identity of the router. Should we ever need to revoke our certificate before it expires (if you renumber your router's interfaces or if you believe that your certificate may be compromised), you will need to provide a password to the CA administrator. Enter that, as is illustrated below. After entering our password, the router will continue on.

```
dt3-45a(config)#cry ca enroll verisign-ca
%Start certificate enrollment ..
%Create a challenge password. You will need to verbally provide this password
to the CA Administrator in order to revoke your certificate.  For security reasons
your password will not be saved in the configuration.  Please make a note of it.

Password:
Re-enter password:
```

You should now see the fingerprint(s) from the CA. Verify that the fingerprint(s) are correct with the CA administrator. In addition, if you do a
sho cry ca cert command, you should see the CA certificate(s), in addition to your own certificates. The CA certificates should be listed as "pending" at this time.

```
% The subject name for the keys will be: dt3-45a.cisco.com
% Include the router serial number in the subject name? [yes/no]: yes
% The serial number in the certificate will be: 01204044
% Include an IP address in the subject name? [yes/no]: yes
Interface: Ethernet 0
Request certificate from CA? [yes/no]: yes
```

Now, contact the CA administrator because this person will want to confirm the host's identity before issuing a certificate. Once the CA issues the certificate, the status of our certificate will change from "pending" to "available." With this, CA enrollment is complete. You are not done yet, however. You still need to configure ISAKMP policy object(s).

Configure ISAKMP Protection Suite(s)

We are using the **rsa-sig** default below. You can have multiple protection suite(s), but there is only one in this example. In the event of multipleprotection suites, the policies will be presented to the peer in numerical order and the peer will negotiate which one to use. We should do this if we know that all of our peers do not support certain features (which features?). The router will not attempt to negotiate things that don't make sense. For example, if we configure your policy for **rsa-sig** and you have no certificate, the router will not negotiate this.

```
dt3-45a(config)#crypto isakmp policy 1
dt3-45a(config-isakmp)#hash md5
dt3-45a(config-isakmp)#lifetime 4000
dt3-45a(config-isakmp)#ex
```

Return to previous section *Return to table of contents* *Go to next section*

Home | What's New | How to Buy | Login | Register | Feedback | Search | Map/Help

An Introduction to IP Security (IPSec) Encryption

Configuring IPSec

Whether you use **pre-shared** keys or configure a CA, once you setup IKE, you still have to setup IPSec. Regardless of which IKE method we use, the configuration steps for IPSec are the same.

To configure IPSec, we need to:

- **Create extended ACL**
- **Create IPSec transform(s)**
- **Create crypto map**
- **Apply crypto map to interface**

Create Extended ACL

The command below is a very simple ACL that allows the routers to talk to one another (a Telnet from one router to the next, for example).

```
dt3-45a(config)# access-list 101 permit ip host 192.168.10.38 host 192.168.10.66
```

A more realistic ACL looks like the command below. This command is an ordinary extended ACL, where 192.168.3.0 is a subnet behind the router in question, and 10.3.2.0 is a subnet somewhere behind the peer router. Remember that **permit** means encrypt and **deny** means don't encrypt.

```
dt3-45a(config)#access-list 101 permit ip 192.168.3.0 0.0.0.255 10.3.2.0 0.0.0.255
```

Create IPSec Transform(s)

Create three transform sets, as we've done in the lines below. The first one uses ESP only, the second one uses AH combined with ESP, and the last one uses only AH. During IPSec SA negotiation, all three will be offered to the peer, which will choose one. Also, for all three transform-sets, we use the default **tunnel mode**. Transport mode can be used only when the crypto endpoints are also the endpoints of the communication. Transpot mode can be specified by the **mode transport** command under the **transform-set** configuration. Tunnel mode is used primarily for the VPN scenario. Also note that **esp-rfc1829** and **ah-rfc1828** are based on the original RFCs for this technology and are obsolete transforms included for backwards compatibility. Not all vendors will support these transforms, but other vendors will support *only* these transforms. Finally, notice that the transform sets in the commands below are not necessarily the most practical. For example, both "PapaBear" and "BabyBear" have sub-standard transform-sets. You should use **esp-rfc1829** and **ah-rfc1828** together in the same transform-set.

```
dt3-45a(config)#crypto ipsec transform-set PapaBear esp-rfc1829
dt3-45a(cfg-crypto-trans)#ex
dt3-45a(config)#crypto ipsec transform-set MamaBear ah-md5-hmac esp-des
dt3-45a(cfg-crypto-trans)#ex
dt3-45a(config)#crypto ipsec transform-set BabyBear ah-rfc1828
dt3-45a(cfg-crypto-trans)#ex
dt3-45a(config)#
```

Create Crypto Map

Using the **ipsec-isakmp** tag tells the router that this crypto map will be an IPSec crypto map. Although there is only one peer declared in this crypto map, you can have multiple peers within a given crypto map. The **session key lifetime** can be expressed in either kilobytes (after x-amount of traffic, change the key) or seconds, as is shown in the commands below. The goal of this is to make a potential attacker's efforts more difficult. The **set transform-set** command is where we associate the transforms with the crypto map. In addition, the order in which we declare the transforms is significant. We most prefer "MamaBear" in this configuration, and then the rest in descending order of preference to "BabyBear." The **match address 101** command simply means to use access list 101 to determine which traffic is relevant. You can have multiple crypto maps with the same name ("armadillo", in the example below) and different sequence numbers ("10", in the example below). The combination of multiple crypto maps and different sequence numbers allows you to mix and match classic crypto and IPSec. You can also modify your PFS configuration here. PFS group1 is the default in the example below. You could change the PFS to group2, or turn it off all together, which you should not do.

```
dt3-45a(config)#crypto map armadillo 10 ipsec-isakmp
dt3-45a(config-crypto-map)#set peer 192.168.10.38
dt3-45a(config-crypto-map)#set session-key lifetime seconds 4000
dt3-45a(config-crypto-map)#set transform-set MamaBear PapaBear BabyBear
dt3-45a(config-crypto-map)#match address 101
```

Apply Crypto Map to Interface

The commands below apply the crypto map to the interface. Remember to apply the crypto map to the egress interface, not the ingress one. If you have multiple crypto maps that you wanted to apply to this interface, you would need to tack the name onto the list in the **crypto map** command.

```
dt3-45a(config)#int e0
dt3-45a(config-if)#crypto map armadillo
```

Return to previous section   *Return to table of contents*   *Go to next section*

Home | What's New | How to Buy | Login | Register | Feedback | Search | Map/Help

An Introduction to IP Security (IPSec) Encryption

Memory and CPU Considerations

Packets that are processed by IPSec will be slower than packets that are processed through classic crypto. There are several reasons for this and they may cause significant performance problems:

1. IPSec introduces packet expansion, which is more likely to require fragmentation and the corresponding reassembly of IPSec datagrams.
2. Encrypted packets will probably be authenticated, which means that there are two cryptographic operations being performed for every packet.
3. The authentication algorithms are slow (although work has been done to speed up things as the Diffie-Hellman computations).

In addition, the Diffie-Hellman key exchange used in IKE is an exponentiation of very large numbers (between 768 and 1024 bytes) and can take up to four seconds on a Cisco 2500. Performance of RSA is dependent on the size of the prime number chosen for the RSA key pair.

For each router, the SA database will take up approximately 300 bytes, plus 120 bytes for every SA therein. In situations where there are two IPSec SAs, one inbound and one outbound, 540 bytes are required; this would be in most cases. Each IKE SA entry is approximately 64 bytes each. The only time you would have one IPSec SA for a dataflow would be when the communication is one-way.

IPSec and IKE will impact performance when active. Diffie-Hellman key exchanges, public key authentication, and encryption/decryption will consume a significant amount of resources, although great lengths have been gone through to minimize this impact.

There should be a small desrease in performance for non-encrypted packets going through an interface that is doing crypto because all packets have to be checked against the crypto map. There should be no performance impact on packets traversing the router that avoid an interface doing crypto. The biggest impact will be on the encrypted data flows themselves.

To minimize the impact of the crypto subsystem on the rest of the router, use Group 1 for Diffie-Hellman key exchanges within IKE, use MD5 as your hashing algorithm, and use longer lifetimes. In tradeoff for this performance tuning, you might get weak cryptography. Ultimately, it is up to the customer's security policy to determine which features to use and which to leave alone.

Return to previous section   *Return to table of contents*   *Go to next section*

An Introduction to IP Security (IPSec) Encryption

Output from show Commands

Note: The following captures were taken from a different series of tests than those used in the previous sections of this document. Consequently, these captures may have different IP addresses and reflect slightly different configurations. Another series of **show** commands are listed under the Debug Information section of this document.

IKE Related Output

To check VeriSign CA enrollment, study the commands below. These commands show you the public keys you are using for RSA encryption and signatures.

```
dt1-45a#show cry key mypubkey rsa
% Key pair was generated at: 11:31:59 PDT Apr 9 1998
Key name: dt1-45a.cisco.com
 Usage: Signature Key
 Key Data:
  305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00C11854
39A9C75C
  4E34C987 B4D7F36C A058D697 13172767 192166E1 661483DD 0FDB907B
F9C10B7A
  CB5A034F A41DF385 23BEB6A7 C14344BE E6915A12 1C86374F 83020301 0001
% Key pair was generated at: 11:32:02 PDT Apr 9 1998
Key name: dt1-45a.cisco.com
 Usage: Encryption Key
 Key Data:
  305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00DCF5AC
360DD5A6
C69704CF 47B2362D 65123BD4 424B6FF6 AD10C33E 89983D08 16F1EA58
3700BCF9
  1EF17E71 5931A9FC 18D60D9A E0852DDD 3F25369C F09DFB75 05020301 0001
```

The command below shows us the certificates that the router recognizes. A certificate that has **pending** status has been submitted to the CA for approval.

```
dt1-45a#show cry ca cert
Certificate
  Subject Name
    Name: dt1-45a.cisco.com
```

```
        Serial Number: 01193485
      Status: Available
      Certificate Serial Number: 650534996414E2BE701F4EF3170EDFAD
      Key Usage: Signature

    CA Certificate
      Status: Available
      Certificate Serial Number: 3051DF7169BEE31B821DFE4B3A338E5F
      Key Usage: Not Set

    Certificate
      Subject Name
        Name: dt1-45a.cisco.com
        Serial Number: 01193485
      Status: Available
      Certificate Serial Number: 1e621faf3b9902bc5b49d0f99dc66d14
      Key Usage: Encryption
```

Below, we see the router's public keys and where the router learned about them:

```
    dt1-45a#show cry key pubkey-chain rsa
    Codes: M - Manually configured, C - Extracted from certificate

    Code Usage   IP-Address       Name
    C    Signing                  Cisco SystemsDevtestCISCOCA-ULTRA
    C    General 172.21.30.71     dt1-7ka.cisco.com
```

This is the ISAKMP (IKE) SA table. Here we see that a SA currently exists between 172.21.30.71 and 172.21.30.70. The peer should have an SA entry in the same state as this router's output.

```
    dt1-7ka#show cry isa sa
        dst           src           state        conn-id   slot
    172.21.30.70   172.21.30.71   QM_IDLE            47      5
```

The lines below show the policy objects configured. In this case, policies **1**, **2,** and **4** are used, in addition to the default. The policies are proposed to the peer in order, with **1** being most preffered.

```
    dt1-45a#show cry isa policy
    Protection suite of priority 1
    encryption algorithm:   DES - Data Encryption Standard (56 bit
    keys).
    hash algorithm:         Message Digest 5
    authentication method:  Rivest-Shamir-Adleman Signature
    Diffie-Hellman group:   #1 (768 bit)
    lifetime:               180 seconds, no volume limit

    Protection suite of priority 2
    encryption algorithm:   DES - Data Encryption Standard (56 bit keys).
    hash algorithm:         Secure Hash Standard
    authentication method:  Pre-Shared Key
    Diffie-Hellman group:   #2 (1024 bit)
    lifetime:               180 seconds, no volume limit
    Protection suite of priority 4
    encryption algorithm:   DES - Data Encryption Standard (56 bit keys).
    hash algorithm:         Message Digest 5
    authentication method:  Pre-Shared Key
    Diffie-Hellman group:   #2 (1024 bit)
    lifetime:               180 seconds, no volume limit

    Default protection suite
    encryption algorithm:   DES - Data Encryption Standard (56 bit keys).
    hash algorithm:         Secure Hash Standard
    authentication method:  Rivest-Shamir-Adleman Signature
    Diffie-Hellman group:   #1 (768 bit)
    lifetime:               86400 seconds, no volume limit
```

IPSec Related show Commands

The command below shows the crypto map **ToOtherRouter**, the ACLs and the transform proposals applied to this crypto map, the peers, and the key lifetime.

```
    S3-2513-2#show cry map
    Crypto Map "ToOtherRouter" 10 ipsec-isakmp
            Peer = 192.168.1.1
            Extended IP access list 101
                access-list 101 permit ip
                    source: addr = 192.168.45.0/0.0.0.255
                    dest:   addr = 192.168.3.0/0.0.0.255
            Connection Id = UNSET    (0 established,    0 failed)
            Current peer: 192.168.1.1
            Session key lifetime: 4608000 kilobytes/3600 seconds
            PFS (Y/N): N
            Transform proposals={ Elvis, Bubba, BarneyDino, }
```

The config below uses the same router as above, but different commands. We see all transform propsals, which settings they will negotiate, and the defaults.

```
    S3-2513-2#show cry ip transform
    Transform proposal Elvis: { ah-sha-hmac  }
       supported settings = { Tunnel,  },
       default settings = { Tunnel,  },
       will negotiate = { Tunnel,  },

       { esp-des  }
       supported settings = { Tunnel,  },
       default settings = { Tunnel,  },
       will negotiate = { Tunnel,  },

    Transform proposal Bubba: { ah-rfc1828  }
```

```
       supported settings = { Tunnel,  },
       default settings = { Tunnel,  },
       will negotiate = { Tunnel,  },

       { esp-des esp-md5-hmac  }
    supported settings = { Tunnel,  },
       default settings = { Tunnel,  },
       will negotiate = { Tunnel,  },

    Transform proposal BarneyDino: { ah-md5-hmac  }
       supported settings = { Tunnel,  },
       default settings = { Tunnel,  },
       will negotiate = { Tunnel,  },
```

This command shows us the current IPSec security associations of this router. The router has one AH SA for both incoming and outgoing.

```
    S3-2513-2#show cry ip session
    Session key lifetime: 4608000 kilobytes/3600 seconds

    S3-2513-2#show cry ip sa

    interface: Ethernet0
        Crypto map tag: ToOtherRouter, local addr. 192.168.1.2

       local  ident (addr/mask/prot/port): (192.168.45.0/255.255.255.0/0/0)
       remote ident (addr/mask/prot/port): (192.168.3.0/255.255.255.0/0/0)
       current_peer: 192.168.1.1
         PERMIT, flags={origin_is_acl,}
        #pkts encaps: 0, #pkts encrypt: 0, #pkts digest 0
        #pkts decaps: 0, #pkts decrypt: 0, #pkts verify 0
        #send errors 5, #recv errors 0

         local crypto endpt.: 192.168.1.2, remote crypto endpt.: 192.168.1.1
         path mtu 1500, media mtu 1500
         current outbound spi: 25081A81

         inbound esp sas:


         inbound ah sas:
          spi: 0x1EE91DDC(518594012)
            transform: ah-md5-hmac ,
            in use settings ={Tunnel, }
            slot: 0, conn id: 16, crypto map: ToOtherRouter
            sa timing: remaining key lifetime (k/sec): (4608000/3423)
            replay detection support: Y
         outbound esp sas:


         outbound ah sas:
          spi: 0x25081A81(621288065)
            transform: ah-md5-hmac ,
    in use settings ={Tunnel, }
            slot: 0, conn id: 17, crypto map: ToOtherRouter
            sa timing: remaining key lifetime (k/sec): (4608000/3424)
            replay detection support: Y
```

Return to previous section   *Return to table of contents*   *Go to next section*

Home | What's New | How to Buy | Login | Register | Feedback | Search | Map/Help

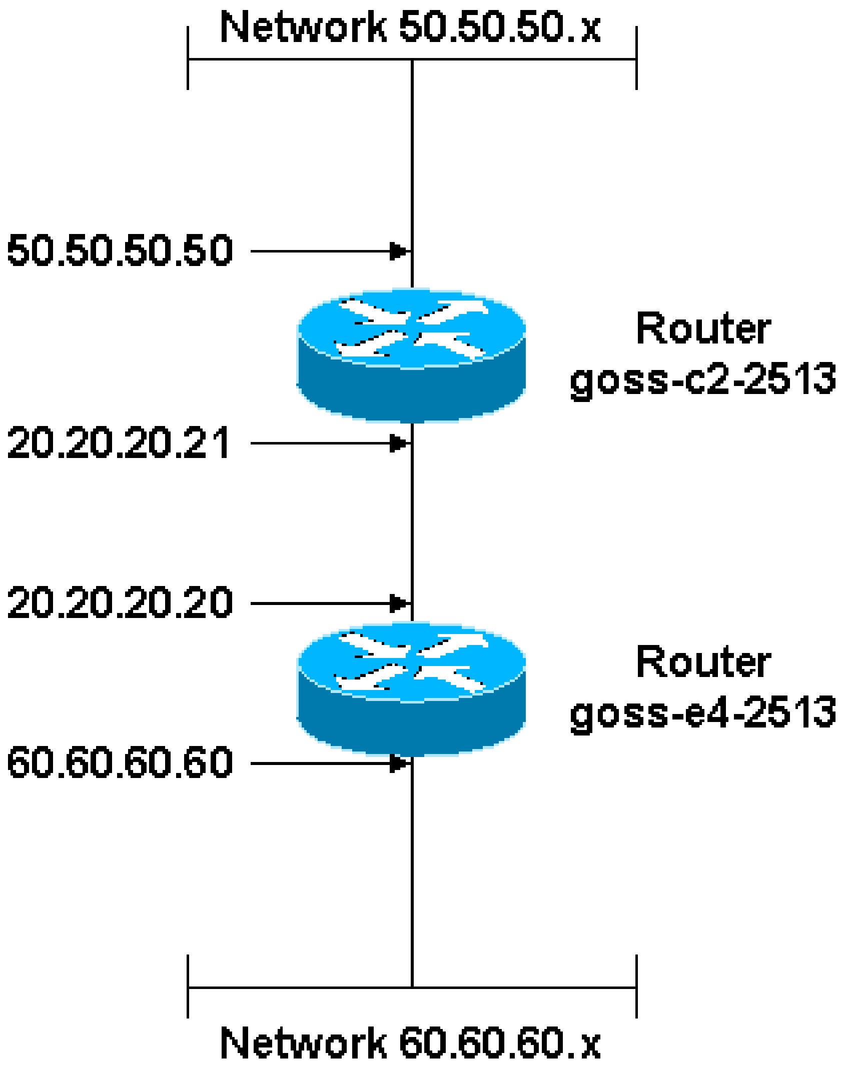
An Introduction to IP Security (IPSec) Encryption

Sample Configurations

Please provide your feedback on this document.

The following configuration uses **pre-shared** keys. This router configuration was used to create the debug output listed in the Debug Information section. This configuration allows a network called "X" located behind "Source Router" to talk to a network called "Y" located behind "Peer Router."

Network Diagram



Configurations

Source Router

```
Current configuration:
!
version 11.3
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname goss-e4-2513
!
enable secret 5 $1$ZuRD$YBaAh3oIv4iltIn0TMCUX1
enable password ww
!
crypto isakmp policy 1
 hash md5
 authentication pre-share
crypto isakmp key Slurpee-Machine address 20.20.20.21
!
crypto ipsec transform-set BearPapa esp-rfc1829
crypto ipsec transform-set BearMama ah-md5-hmac esp-des
crypto ipsec transform-set BearBaby ah-rfc1828
 !
 crypto map armadillo 1 ipsec-isakmp
 set peer 20.20.20.21
 set security-association lifetime seconds 190
 set transform-set BearPapa BearMama BearBaby
 match address 101
!
interface Ethernet0
 ip address 60.60.60.60 255.255.255.0
 no mop enabled
!
interface Serial0
 ip address 20.20.20.20 255.255.255.0
 no ip mroute-cache
 no fair-queue
 crypto map armadillo
!
interface Serial1
 no ip address
 shutdown
!
interface TokenRing0
 no ip address
 shutdown
!
ip classless
ip route 0.0.0.0 0.0.0.0 20.20.20.21
!
access-list 101 permit ip 60.60.60.0 0.0.0.255 50.50.50.0 0.0.0.255
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
!
line con 0
 exec-timeout 0 0
line aux 0
line vty 0 4
 password ww
 login
!
end
```

Peer Router

```
Current configuration:
!
version 11.3
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname goss-c2-2513
!
enable secret 5 $1$DBTl$Wtg2eS7Eb/Cw5l.nDhkEi/
enable password ww
!
ip subnet-zero
!
!
crypto isakmp policy 1
 hash md5
 authentication pre-share
crypto isakmp key Slurpee-Machine address 20.20.20.20
!
!
crypto ipsec transform-set PapaBear esp-rfc1829
crypto ipsec transform-set MamaBear ah-md5-hmac esp-des
crypto ipsec transform-set BabyBear ah-rfc1828
!
 !
 crypto map armadillo 1 ipsec-isakmp
 set peer 20.20.20.20
 set security-association lifetime seconds 190
 set transform-set MamaBear PapaBear BabyBear
 match address 101
!
!
!
interface Ethernet0
 ip address 50.50.50.50 255.255.255.0
 no ip directed-broadcast
!
interface Serial0
 ip address 20.20.20.21 255.255.255.0
 no ip directed-broadcast
 no ip mroute-cache
 no fair-queue
 clockrate 9600
 crypto map armadillo
!
interface Serial1
 no ip address
 no ip directed-broadcast
 shutdown
!
interface TokenRing0
 no ip address
 no ip directed-broadcast
 shutdown
!
ip classless
ip route 0.0.0.0 0.0.0.0 20.20.20.20
!
access-list 101 permit ip 50.50.50.0 0.0.0.255 60.60.60.0 0.0.0.255
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
!
!
line con 0
 exec-timeout 0 0
 transport input none
line aux 0
line aux 0
line vty 0 4
 password ww
 login
!
end
```

Return to previous section *Return to table of contents* *Go to next section*

Home | What's New | How to Buy | Login | Register | Feedback | Search | Map/Help

An Introduction to IP Security (IPSec) Encryption

Debug Information

Please provide your feedback on this document.

This section has the debug output from a normal IKE/IPSec session between two routers. The configurations come from the Sample Configurations section of this document. The routers use a **pre-shared** key. Both routers have the **debug crypto isakmp**, **debug crypto ipsec**, and **debug crypto engine** commands enabled. This was tested with an extended ping from the Source Router ethernet interface to the Peer Router ethernet interface (60.60.60.60 to 50.50.50.50).

Note: The blue, italic statements in the following sample debug output are notes to help you follow what is happening, they are not part of the debug output.

Source Router

```
goss-e4-2513#show clock
goss-e4-2513#ping
Protocol [ip]:
Target IP address: 50.50.50.50
Repeat count [5]: 10
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 60.60.60.60
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 10, 100-byte ICMP Echos to 50.50.50.50, timeout is 2 seconds:

Apr  2 12:03:55.347: IPSEC(sa_request): ,
  (key eng. msg.) src= 20.20.20.20, dest= 20.20.20.21,
    src_proxy= 60.60.60.0/255.255.255.0/0/0 (type=4),
    dest_proxy= 50.50.50.0/255.255.255.0/0/0 (type=4),
    protocol= ESP, transform= esp-rfc1829 ,
    lifedur= 190s and 4608000kb,
    spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4004
Apr  2 12:03:55.355: IPSEC(sa_request): ,
  (key eng. msg.) src= 20.20.20.20, dest= 20.20.20.21,
    src_proxy= 60.60.60.0/255.255.255.0/0/0 (type=4),
    dest_proxy= 50.50.50.0/255.255.255.0/0/0 (type=4),
    protocol= AH, transform= ah-md5-hmac ,
    lifedur= 190s and 4608000kb,
    spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4004
Apr  2 12:03:55.363: IPSEC(sa_request): ,
  (key eng. msg.) src= 20.20.20.20, dest= 20.20.20.21,
    src_proxy= 60.60.60.0/255.255.255.0/0/0 (type=4),
    dest_proxy= 50.50.50.0/255.255.255.0/0/0 (type=4),
    protocol= ESP, transform= esp-des ,
    lifedur= 190s and 4608000kb,
    spi= 0x0(0), conn_id= 0, keysi.ze= 0, flags= 0x4004
Apr  2 12:03:55.375: IPSEC(sa_request): ,
  (key eng. msg.) src= 20.20.20.20, dest= 20.20.20.21,
    src_proxy= 60.60.60.0/255.255.255.0/0/0 (type=4),
    dest_proxy= 50.50.50.0/255.255.255.0/0/0 (type=4),
    protocol= AH, transform= ah-rfc1828 ,
    lifedur= 190s and 4608000kb,
    spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4004

!--- Note that the router is offering to the peer all of the
!--- available transforms (above).

Apr  2 12:03:55.391: ISAKMP (14): beginning Main Mode exchange
Apr  2 12:03:57.199: ISAKMP (14): processing SA payload. message ID = 0
Apr  2 12:03:57.203: ISAKMP (14): Checking ISAKMP transform 1 against
    priority 1 policy
Apr  2 12:03:57.203: ISAKMP:      encryption DES-CBC
Apr  2 12:03:57.207: ISAKMP:      hash MD5
Apr  2 12:03:57.207: ISAKMP:      default group 1
Apr  2 12:03:57.207: ISAKMP:      auth pre-share
Apr  2 12:03:57.211: ISAKMP (14): atts are acceptable. Next payload is 0
Apr  2 12:03:57.215: Crypto engine 0: generate alg param

Apr  2 12:03:5.8.867: CRYPTO_ENGINE: Dh phase 1 status: 0
Apr  2 12:03:58.871: ISAKMP (14): SA is doing pre-shared key authentication..
Apr  2 12:04:01.291: ISAKMP (14): processing KE payload. message ID = 0
Apr  2 12:04:01.295: Crypto engine 0: generate alg param

Apr  2 12:04:03.343: ISAKMP (14): processing NONCE payload. message ID = 0
Apr  2 12:04:03.347: Crypto engine 0: create ISAKMP SKEYID for conn id 14
Apr  2 12:04:03.363: ISAKMP (14): SKEYID state generated
Apr  2 12:04:03.367: ISAKMP (14): processing vendor id payload
Apr  2 12:04:03.371: ISAKMP (14): speaking to another IOS box!
Apr  2 12:04:03.371: generate hmac context for conn id 14
Apr  2 12:04:03.615: ISAKMP (14): processing ID payload. message ID = 0
Apr  2 12:04:03.615: ISAKMP (14): processing HASH payload. message ID = 0
Apr  2 12:04:03.619: generate hmac context for conn id 14
Apr  2 12:04:03.627: ISAKMP (14): SA has been authenticated
Apr  2 12:04:03.627: ISAKMP (14): beginning Quick Mode exchange, M-ID of 1628162439

!--- The lines above represent verification that the policy
!--- attributes are fine, and the final authentication of the IKE SA.
!--- Once the IKE SA has been authenticated, a valid IKE SA exists.
!--- New IKE kicks off IPSec negotiation:

Apr  2 12:04:03.635: IPSEC(key_engine): got a queue event...
Apr  2 12:04:03.635: IPSEC(spi_response): getting spi 303564824ld for SA
```

```
        .!!!from 20.20.20.21     to 20.20.20.20     for prot 3
Apr  2 12:04:03.639: IPSEC(spi_response): getting spi 423956280ld for SA
        from 20.20.20.21     to 20.20.20.20     for prot 2
Apr  2 12:04:03.643: IPSEC(spi_response): getting spi 415305621ld for SA
        from 20.20.20.21     to 20.20.20.20     for prot 3
Apr  2 12:04:03.647: IPSEC(spi_response): getting spi 218308976ld for SA
        from 20.20.20.21     to 20.20.20.20     for prot 2
Apr  2 12:04:03.891: generate hmac context for conn id 14
Apr  2 12:04:04.!!
Success rate is 50 percent (5/10), round-trip min/avg/max = 264/265/268 ms
goss-e4-2513#723: generate hmac context for conn id 14
Apr  2 12:04:04.731: ISAKMP (14): processing SA payload. message ID = 1628162439
Apr  2 12:04:04.731: ISAKMP (14): Checking IPSec proposal 1
Apr  2 12:04:04.735: ISAKMP: transform 1, ESP_DES_IV64
Apr  2 12:04:04.735: ISAKMP:   attributes in transform:
Apr  2 12:04:04.735: ISAKMP:      encaps is 1
Apr  2 12:04:04.739: ISAKMP:      SA life type in seconds
Apr  2 12:04:04.739: ISAKMP:      SA life duration (basic) of 190
Apr  2 12:04:04.739: ISAKMP:      SA life type in kilobytes
Apr  2 12:04:04.743: ISAKMP:      SA life duration (VPI) of  0x0 0x46 0x50 0x0
Apr  2 12:04:04.747: ISAKMP (14): atts are acceptable.

!--- You may be wondering why ISAKMP debug is listed above,
!--- even after IKE has done its job. Remember that IKE is the
!--- entity that negotiates IPSec SAs on behalf of IPSec.

Apr  2 12:04:04.747: IPSEC(validate_proposal_request): proposal part #1,
  (key eng. msg.) dest= 20.20.20.21, src= 20.20.20.20,
    dest_proxy= 50.50.50.0/255.255.255.0/0/0 (type=4),
    src_proxy= 60.60.60.0/255.255.255.0/0/0 (type=4),
    protocol= ESP, transform= esp-rfc1829 ,
    lifedur= 0s and 0kb,
    spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4
Apr  2 12:04:04.759: ISAKMP (14): processing NONCE payload. message ID = 1628162439
Apr  2 12:04:04.759: ISAKMP (14): processing ID payload. message ID = 1628162439
Apr  2 12:04:04.763: ISAKMP (14): processing ID payload. message ID = 1628162439
Apr  2 12:04:04.767: generate hmac context for conn id 14
Apr  2 12:04:04.799: ISAKMP (14): Creating IPSec SAs
Apr  2 12:04:04.803:         inbound SA from 20.20.20.21 to 20.20.20.20
    (proxy 50.50.50.0 to 60.60.60.0)
Apr  2 12:04:04.803:         has spi 303564824 and conn_id 15 and flags 4
Apr  2 12:04:04.807:         lifetime of 190 seconds
Apr  2 12:04:04.807:         lifetime of 4608000 kilobytes
Apr  2 12:04:04.811:         outbound SA from 20.20.20.20 to 20.20.20.21
    (proxy 60.60.60.0 to 50.50.50.0)
Apr  2 12:04:04.811:         has spi 183903875 and conn_id 16 and flags 4
Apr  2 12:04:04.815:         lifetime of 190 seconds
Apr  2 12:04:04.815:         lifetime of 4608000 kilobytes
Apr  2 12:04:04.823: IPSEC(key_engine): got a queue event...
Apr  2 12:04:04.823: IPSEC(initialize_sas): ,
  (key eng. msg.) dest= 20.20.20.20, src= 20.20.20.21,
    dest_proxy= 60.60.60.0/255.255.255.0/0/0 (type=4),
    src_proxy= 50.50.50.0/255.255.255.0/0/0 (type=4),
    protocol= ESP, transform= esp-rfc1829 ,
    lifedur= 190s and 4608000kb,
    spi= 0x12180818(303564824), conn_id= 15, keysize= 0, flags= 0x4
Apr  2 12:04:04.831: IPSEC(initialize_sas): ,
  (key eng. msg.) src= 20.20.20.20, dest= 20.20.20.21,
    src_proxy= 60.60.60.0/255.255.255.0/0/0 (type=4),
    dest_proxy= 50.50.50.0/255.255.255.0/0/0 (type=4),
    protocol= ESP, transform= esp-rfc1829 ,
    lifedur= 190s and 4608000kb,
    spi= 0xAF62683(183903875), conn_id= 16, keysize= 0, flags= 0x4
Apr  2 12:04:04.839: IPSEC(create_sa): sa created,
  (sa) sa_dest= 20.20.20.20, sa_prot= 50,
    sa_spi= 0x12180818(303564824),
    sa_trans= esp-rfc1829 , sa_conn_id= 15
Apr  2 12:04:04.843: IPSEC(create_sa): sa created,
  (sa) sa_dest= 20.20.20.21, sa_prot= 50,
    sa_spi= 0xAF62683(183903875),
    sa_trans= esp-rfc1829 , sa_conn_id= 16

!--- The lines above show that IPSec SAs are created and
!--- encrypted traffic can now pass.
```

Source Router show Command Output After IKE/IPSec Negotiation

```
goss-e4-2513#
goss-e4-2513#sho cry is sa
    dst              src            state          conn-id   slot
20.20.20.21    20.20.20.20    QM_IDLE             14       0

goss-e4-2513#show cry ips sa

interface: Serial0
    Crypto map tag: armadillo, local addr. 20.20.20.20

   local  ident (addr/mask/prot/port): (60.60.60.0/255.255.255.0/0/0)
   remote ident (addr/mask/prot/port): (50.50.50.0/255.255.255.0/0/0)
   current_peer: 20.20.20.21
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 5, #pkts encrypt: 5, #pkts digest 0
    #pkts decaps: 5, #pkts decrypt: 5, #pkts verify 0
    #send errors 5, #recv errors 0

     local crypto endpt.: 20.20.20.20, remote crypto endpt.: 20.20.20.21
     path mtu 1500, media mtu 1500
     current outbound spi: AF62683

     inbound esp sas:
      spi: 0x12180818(303564824)
        transform: esp-rfc1829 ,
        in use settings ={Var len IV, Tunnel, }
        slot: 0, conn id: 15, crypto map: armadillo
        sa timing: remaining key lifetime (k/sec): (4607999/135)
        IV size: 8 bytes
        replay detection support: N
     inbound ah sas:


     outbound esp sas:
      spi: 0xAF62683(183903875)
        transform: esp-rfc1829 ,
        in use settings ={Var len IV, Tunnel, }
        slot: 0, conn id: 16, crypto map: armadillo
        sa timing: remaining key lifetime (k/sec): (4607999/117)
        IV size: 8 bytes
        replay detection support: N


     outbound ah sas:



goss-e4-2513#sho cry is pol
Protection suite of priority 1
        encryption algorithm:   DES - Data Encryption Standard (56 bit keys).
        hash algorithm:         Message Digest 5
        authentication method:  Pre-Shared Key
        Diffie-Hellman group:   #1 (768 bit)
        lifetime:               86400 seconds, no volume limit
Default protection suite
        encryption algorithm:   DES - Data Encryption Standard (56 bit keys).
        hash algorithm:         Secure Hash Standard
        authentication method:  Rivest-Shamir-Adleman Signature
        Diffie-Hellman group:   #1 (768 bit)
        lifetime:               86400 seconds, no volume limit
goss-e4-2513#sho cry map
Crypto Map "armadillo" 1 ipsec-isakmp
        Peer = 20.20.20.21
        Extended IP access list 101
                access-list 101 permit ip 60.60.60.0 0.0.0.255 50.50.50.0 0.0.0.255
        Current peer: 20.20.20.21
        Security association lifetime: 4608000 kilobytes/190 seconds
        PFS (Y/N): N
        Transform sets={ BearPapa, BearMama, BearBaby, }
```

Peer Router with Same Ping Sequence, as Seen from the Other Side

```
goss-c2-2513#sho deb
Cryptographic Subsystem:
  Crypto ISAKMP debugging is on
  Crypto Engine debugging is on
  Crypto IPSEC debugging is on
goss-c2-2513#
Apr  2 12:03:55.107: ISAKMP (14): processing SA payload. message ID = 0
Apr  2 12:03:55.111: ISAKMP (14): Checking ISAKMP transform 1 against
     priority 1 policy
Apr  2 12:03:55.111: ISAKMP:      encryption DES-CBC
Apr  2 12:03:55.111: ISAKMP:      hash MD5
Apr  2 12:03:55.115: ISAKMP:      default group 1
Apr  2 12:03:55.115: ISAKMP:      auth pre-share
Apr  2 12:03:55.115: ISAKMP (14): atts are acceptable. Next payload is 0

!--- IKE performs its operation, and then kicks off IPSec.

Apr  2 12:03:55.119: Crypto engine 0: generate alg param

Apr  2 12:03:56.707: CRYPTO_ENGINE: Dh phase 1 status: 0
Apr  2 12:03:56.711: ISAKMP (14): SA is doing pre-shared key authentication
Apr  2 12:03:58.667: ISAKMP (14): processing KE payload. message ID = 0
Apr  2 12:03:58.671: Crypto engine 0: generate alg param

Apr  2 12:04:00.687: ISAKMP (14): processing NONCE payload. message ID = 0
Apr  2 12:04:00.695: Crypto engine 0: create ISAKMP SKEYID for conn id 14
Apr  2 12:04:00.707: ISAKMP (14): SKEYID state generated
Apr  2 12:04:00.711: ISAKMP (14): processing vendor id payload
Apr  2 12:04:00.715: ISAKMP (14): speaking to another IOS box!
Apr  2 12:04:03.095: ISAKMP (14): processing ID payload. message ID = 0
Apr  2 12:04:03.095: ISAKMP (14): processing HASH payload. message ID = 0
Apr  2 12:04:03.099: generate hmac context for conn id 14
Apr  2 12:04:03.107: ISAKMP (14): SA has been authenticated
Apr  2 12:04:03.111: generate hmac context for conn id 14
Apr  2 12:04:03.835: generate hmac context for conn id 14
Apr  2 12:04:03.839: ISAKMP (14): processing SA payload. message ID = 1628162439
Apr  2 12:04:03.843: ISAKMP (14): Checking IPSec proposal 1
Apr  2 12:04:03.843: ISAKMP: transform 1, ESP_DES_IV64
Apr  2 12:04:03.847: ISAKMP:   attributes in transform:
Apr  2 12:04:03.847: ISAKMP:      encaps is 1
Apr  2 12:04:03.847: ISAKMP:      SA life type in seconds
Apr  2 12:04:03.851: ISAKMP:      SA life duration (basic) of 190
Apr  2 12:04:03.851: ISAKMP:      SA life type in kilobytes
Apr  2 12:04:03.855: ISAKMP:      SA life duration (VPI) of  0x0 0x46 0x50 0x0
Apr  2 12:04:03.855: ISAKMP (14): atts are acceptable.
Apr  2 12:04:03.859: IPSEC(validate_proposal_request): proposal part #1,
  (key eng. msg.) dest= 20.20.20.21, src= 20.20.20.20,
    dest_proxy= 50.50.50.0/255.255.255.0/0/0 (type=4),
    src_proxy= 60.60.60.0/255.255.255.0/0/0 (type=4),
    protocol= ESP, transform= esp-rfc1829 ,
    lifedur= 0s and 0kb,
    spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4
Apr  2 12:04:03.867: ISAKMP (14): processing NONCE payload. message ID = 1628162439
Apr  2 12:04:03.871: ISAKMP (14): processing ID payload. message ID = 1628162439
Apr  2 12:04:03.871: ISAKMP (14): processing ID payload. message ID = 1628162439
Apr  2 12:04:03.879: IPSEC(key_engine): got a queue event...
Apr  2 12:04:03.879: IPSEC(spi_response): getting spi 183903875ld for SA
        from 20.20.20.20     to 20.20.20.21     for prot 3
Apr  2 12:04:04.131: generate hmac context for conn id 14
Apr  2 12:04:04.547: generate hmac context for conn id 14
Apr  2 12:04:04.579: ISAKMP (14): Creating IPSec SAs
Apr  2 12:04:04.579:         inbound SA from 20.20.20.20 to 20.20.20.21
    (proxy 60.60.60.0 to 50.50.50.0)
Apr  2 12:04:04.583:         has spi 183903875 and conn_id 15 and flags 4
Apr  2 12:04:04.583:         lifetime of 190 seconds
Apr  2 12:04:04.587:         lifetime of 4608000 kilobytes
Apr  2 12:04:04.587:         outbound SA from 20.20.20.21 to 20.20.20.20
    (proxy 50.50.50.0 to 60.60.60.0)
```

```
Apr  2 12:04:04.591:         has spi 303564824 and conn_id 16 and flags 4
Apr  2 12:04:04.591:         lifetime of 190 seconds
Apr  2 12:04:04.595:         lifetime of 4608000 kilobytes
Apr  2 12:04:04.599: IPSEC(key_engine): got a queue event...
Apr  2 12:04:04.599: IPSEC(initialize_sas): ,
  (key eng. msg.) dest= 20.20.20.21, src= 20.20.20.20,
    dest_proxy= 50.50.50.0/255.255.255.0/0/0 (type=4),
    src_proxy= 60.60.60.0/255.255.255.0/0/0 (type=4),
    protocol= ESP, transform= esp-rfc1829 ,
    lifedur= 190s and 4608000kb,
    spi= 0xAF62683(183903875), conn_id= 15, keysize= 0, flags= 0x4
Apr  2 12:04:04.607: IPSEC(initialize_sas): ,
  (key eng. msg.) src= 20.20.20.21, dest= 20.20.20.20,
    src_proxy= 50.50.50.0/255.255.255.0/0/0 (type=4),
    dest_proxy= 60.60.60.0/255.255.255.0/0/0 (type=4),
    protocol= ESP, transform= esp-rfc1829 ,
    lifedur= 190s and 4608000kb,
    spi= 0x12180818(303564824), conn_id= 16, keysize= 0, flags= 0x4
Apr  2 12:04:04.615: IPSEC(create_sa): sa created,
  (sa) sa_dest= 20.20.20.21, sa_prot= 50,
    sa_spi= 0xAF62683(183903875),
    sa_trans= esp-rfc1829 , sa_conn_id= 15
Apr  2 12:04:04.619: IPSEC(create_sa): sa created,
  (sa) sa_dest= 20.20.20.20, sa_prot= 50,
    sa_spi= 0x12180818(303564824),
    sa_trans= esp-rfc1829 , sa_conn_id= 16

!--- The IPSec SAs are created, and ICMP traffic can flow.
```

Peer Router show Commands

```
!--- This illustrates a series of show command output after
IKE/IPSec negotiation has taken place.

goss-c2-2513#sho cry is sa
    dst              src            state          conn-id    slot
20.20.20.21    20.20.20.20    QM_IDLE             14       0

goss-c2-2513#sho cry ips sa

interface: Serial0
    Crypto map tag: armadillo, local addr. 20.20.20.21

   local  ident (addr/mask/prot/port): (50.50.50.0/255.255.255.0/0/0)
   remote ident (addr/mask/prot/port): (60.60.60.0/255.255.255.0/0/0)
   current_peer: 20.20.20.20
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 5, #pkts encrypt: 5, #pkts digest 0
    #pkts decaps: 5, #pkts decrypt: 5, #pkts verify 0
    #send errors 0, #recv errors 0

     local crypto endpt.: 20.20.20.21, remote crypto endpt.: 20.20.20.20
     path mtu 1500, media mtu 1500
     current outbound spi: 12180818
     inbound esp sas:
      spi: 0xAF62683(183903875)
        transform: esp-rfc1829 ,
        in use settings ={Var len IV, Tunnel, }
        slot: 0, conn id: 15, crypto map: armadillo
        sa timing: remaining key lifetime (k/sec): (4607999/118)
        IV size: 8 bytes
        replay detection support: N


     inbound ah sas:


     outbound esp sas:
      spi: 0x12180818(303564824)
        transform: esp-rfc1829 ,
        in use settings ={Var len IV, Tunnel, }
        slot: 0, conn id: 16, crypto map: armadillo
        sa timing: remaining key lifetime (k/sec): (4607999/109)
        IV size: 8 bytes
        replay detection support: N


     outbound ah sas:



goss-c2-2513#sho cry is pol
Protection suite of priority 1
        encryption algorithm:   DES - Data Encryption Standard (56 bit keys).
        hash algorithm:         Message Digest 5
        authentication method:  Pre-Shared Key
        Diffie-Hellman group:   #1 (768 bit)
        lifetime:               86400 seconds, no volume limit
Default protection suite
        encryption algorithm:   DES - Data Encryption Standard (56 bit keys).
        hash algorithm:         Secure Hash Standard
        authentication method:  Rivest-Shamir-Adleman Signature
        Diffie-Hellman group:   #1 (768 bit)
        lifetime:               86400 seconds, no volume limit
goss-c2-2513#sho cry map
Crypto Map "armadillo" 1 ipsec-isakmp
        Peer = 20.20.20.20
        Extended IP access list 101
                access-list 101 permit ip 50.50.50.0 0.0.0.255 60.60.60.0 0.0.0.255
        Current peer: 20.20.20.20
        Security association lifetime: 4608000 kilobytes/190 seconds
        PFS (Y/N): N
        Transform sets={ MamaBear, PapaBear, BabyBear, }
```

An Introduction to IP Security (IPSec) Encryption

Implementation Tips for IPSec

Here are some implementation tips for IPSec:

- Before configuring crypto, make certain that you have connectivity between the endpoints of the communication.
- Make sure you have either DNS working on the router, or you have entered the CA hostname (if you are using a CA).
- IPSec uses IP protocols 50 and 51, and IKE traffic passes on protocol 17, port 500 (UDP 500). Make sure these are permitted appropriately.
- Be careful not to use the word "any" in your ACL. This will cause problems.
- Recommended transform combinations are:

```
        esp-des and esp-sha-hmac
        ah-sha-hmac and esp-des
```

- Remember that AH is just an authenticated header. The actual user datastream is not being encrypted. For datastream encryption, you need ESP. If you use only AH and see cleartext going across the network, don't be surprised. You if you use AH, you should use ESP as well. Note that ESP can perform authentication also. Therefore, you can use a transform combination such as **esp-des** and **esp-sha-hmac**.
- **ah-rfc1828** and **esp-rfc1829** are obsolete transforms included for backwards compatibility with older IPSec implementations. If the peer doesn't support newer transforms, try these instead.
- Remember that SHA is slower and more secure than MD5, whereas MD5 is faster and less secure that SHA. In fact, in some communities, the comfort level with MD5 is very low.
- When in doubt, use tunnel mode. Tunnel mode is the default and it can be used in transport mode, as well as for its VPN capabilities.
- For classic crypto users upgrading to Cisco IOS® Software release 11.3, crypto commands storage methods in the configuration has changed to allow for IPSec. Consequently, should classic crypto users ever revert to IOS release 11.2, these users will have to re-enter their crypto configurations.
- If you do a ping test across the encrypted link when you finish your configuration, the negotiation process may take some time (about six seconds on a Cisco 4500, and about 20 seconds on a Cisco 2500) because SAs have not yet been negotiated. Therefore, even though everything may be configured correctly, your ping may initially fail. The **debug cry ipsec** and **debug cry isakmp**

commands will show you what is going on. Once your encrypted datastreams have finished setting up, the ping should work fine.

- If you run into trouble with your negotiation(s) and make config changes, use the **clear cry is** and **clear cry sa** commands to flush the databases before retrying. This forces negotiation to start anew, without any legacy negotiation getting in the way. The **clear cry is** and **clear cry sa** commands are very useful in this manner.

Return to previous section *Return to table of contents* *Go to next section*

Home | What's New | How to Buy | Login | Register | Feedback | Search | Map/Help

An Introduction to IP Security (IPSec) Encryption

Help and Relevant Links

IPSec Support Page

ECRA Encryption Policies & Procedures
Send mail to: export@cisco.com

More Sample Configs for IPSec:

IPSec Technical Tips and Sample Configurations

Configuring and Troubleshooting Cisco's Network-Layer Encryption: IPSec and ISAKMP

IPSec Overview

PIX Firewall IPSec configuration documentation:

- PIX 5.1
- PIX 5.2
- PIX 5.3
- PIX 6.0

If you require further assistance with IPSec, please contact the Cisco Systems Technical Assistance Center (TAC) at (800)553-24HR, (408)526-7209, or send e-mail to tac@cisco.com.

Return to previous section   *Return to table of contents*   *Go to next section*

An Introduction to IP Security (IPSec) Encryption

References

Harkins, D. *ISAKMP/Oakley Protocol Feature Software Unit Functional Specification.* ENG-0000 Rev A. Cisco Systems.

Madson, C. *IPSec Software Unit Functional Specification ENG-17610 Rev F.* Cisco Systems.

Kaufman, C. Perlman R. and Spencer, M. *Network Security: Private Communication in a Public World.* Prentice Hall, 1995.

Schneier, B. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* Second Ed. John Wiley & Sons, Inc.

Various IETF IP Security working-drafts

Return to previous section   *Return to table of contents*

Home | What's New | How to Buy | Login | Register | Feedback | Search | Map/Help