

Encryption

There are two main types of data that must be protected: data at rest and data in motion. These different types of data are protected in similar ways using similar technology but the implementations can be completely different. No single protective implementation can prevent all possible methods of compromise as the same information may be at rest and in motion at different points in time.

4.1. Data at Rest

Data at rest is data that is stored on a hard drive, tape, CD, DVD, disk, or other media. This information's biggest threat comes from being physically stolen. Laptops in airports, CDs going through the mail, and backup tapes that get left in the wrong places are all examples of events where data can be compromised through theft. If the data was encrypted on the media then you wouldn't have to worry as much about the data being compromised.

4.1.1. Full Disk Encryption

Full disk or partition encryption is one of the best ways of protecting your data. Not only is each file protected but also the temporary storage that may contain parts of these files is also protected. Full disk encryption will protect all of your files so you don't have to worry about selecting what you want to protect and possibly missing a file.

Fedora natively supports LUKS Encryption. LUKS will bulk encrypt your hard drive partitions so that while your computer is off your data is protected. This will also protect your computer from attackers attempting to use single-user-mode to login to your computer or otherwise gain access.

Full disk encryption solutions like LUKS only protect the data when your computer is off. Once the computer is on and LUKS has decrypted the disk, the files on that disk are available to anyone who would normally have access to them. To protect your files when the computer is on, use full disk encryption in combination with another solution such as file based encryption. Also remember to lock your computer whenever you are away from it. A passphrase protected screen saver set to activate after a few minutes of inactivity is a good way to keep intruders out.

4.1.2. File Based Encryption

GnuPG (GPG) is an open source version of PGP that allows you to sign and/or encrypt a file or an email message. This is useful to maintain integrity of the message or file and also protects the confidentiality of the information contained within the file or email. In the case of email, GPG provides dual protection. Not only can it provide Data at Rest protection but also Data In Motion protection once the message has been sent across the network.

File based encryption is intended to protect a file after it has left your computer, such as when you send a CD through the mail. Some file based encryption solutions will leave remnants of the encrypted files that an attacker who has physical access to your computer can recover under some circumstances. To protect the contents of those files from attackers who may have access to your computer, use file based encryption combined with another solution such as full disk encryption.

4.2. Data in Motion

Data in motion is data that is being transmitted over a network. The biggest threats to data in motion are interception and alteration. Your user name and password should never be transmitted over a network without protection as it could be intercepted and used by someone else to impersonate you or gain access to sensitive information. Other private information such as bank account information should also be protected when transmitted across a network. If the network session was encrypted

then you would not have to worry as much about the data being compromised while it is being transmitted.

Data in motion is particularly vulnerable to attackers because the attacker does not have to be near the computer in which the data is being stored rather they only have to be somewhere along the path. Encryption tunnels can protect data along the path of communications.

4.2.1. Virtual Private Networks (VPNs)

Organizations with several satellite offices often connect to each other with dedicated lines for efficiency and protection of sensitive data in transit. For example, many businesses use frame relay or *Asynchronous Transfer Mode* (ATM) lines as an end-to-end networking solution to link one office with others. This can be an expensive proposition, especially for small to medium sized businesses (SMBs) that want to expand without paying the high costs associated with enterprise-level, dedicated digital circuits.

To address this need, *Virtual Private Networks* (VPNs) were developed. Following the same functional principles as dedicated circuits, VPNs allow for secured digital communication between two parties (or networks), creating a *Wide Area Network* (WAN) from existing *Local Area Networks* (LANs). Where it differs from frame relay or ATM is in its transport medium. VPNs transmit over IP using datagrams as the transport layer, making it a secure conduit through the Internet to an intended destination. Most free software VPN implementations incorporate open standard encryption methods to further mask data in transit.

Some organizations employ hardware VPN solutions to augment security, while others use software or protocol-based implementations. Several vendors provide hardware VPN solutions, such as Cisco, Nortel, IBM, and Checkpoint. There is a free software-based VPN solution for Linux called FreeS/Wan that utilizes a standardized *Internet Protocol Security* (IPsec) implementation. These VPN solutions, irrespective of whether they are hardware or software based, act as specialized routers that exist between the IP connection from one office to another.

4.2.1.1. How Does a VPN Work?

When a packet is transmitted from a client, it sends it through the VPN router or gateway, which adds an *Authentication Header* (AH) for routing and authentication. The data is then encrypted and, finally, enclosed with an *Encapsulating Security Payload* (ESP). This latter constitutes the decryption and handling instructions.

The receiving VPN router strips the header information, decrypts the data, and routes it to its intended destination (either a workstation or other node on a network). Using a network-to-network connection, the receiving node on the local network receives the packets already decrypted and ready for processing. The encryption/decryption process in a network-to-network VPN connection is transparent to a local node.

With such a heightened level of security, an attacker must not only intercept a packet, but decrypt the packet as well. Intruders who employ a man-in-the-middle attack between a server and client must also have access to at least one of the private keys for authenticating sessions. Because they employ several layers of authentication and encryption, VPNs are a secure and effective means of connecting multiple remote nodes to act as a unified intranet.

4.2.1.2. VPNs and Fedora

Fedora provides various options in terms of implementing a software solution to securely connect to a WAN. *Internet Protocol Security* (IPsec) is the supported VPN implementation for Fedora, and sufficiently addresses the usability needs of organizations with branch offices or remote users.

4.2.1.3. IPsec

Fedora supports IPsec for connecting remote hosts and networks to each other using a secure tunnel on a common carrier network such as the Internet. IPsec can be implemented using a host-to-host (one computer workstation to another) or network-to-network (one LAN/WAN to another) configuration.

The IPsec implementation in Fedora uses *Internet Key Exchange (IKE)*, a protocol implemented by the Internet Engineering Task Force (IETF), used for mutual authentication and secure associations between connecting systems.

4.2.1.4. Creating an IPsec Connection

An IPsec connection is split into two logical phases. In phase 1, an IPsec node initializes the connection with the remote node or network. The remote node or network checks the requesting node's credentials and both parties negotiate the authentication method for the connection.

On Fedora systems, an IPsec connection uses the *pre-shared key* method of IPsec node authentication. In a pre-shared key IPsec connection, both hosts must use the same key in order to move to Phase 2 of the IPsec connection.

Phase 2 of the IPsec connection is where the *Security Association (SA)* is created between IPsec nodes. This phase establishes an SA database with configuration information, such as the encryption method, secret session key exchange parameters, and more. This phase manages the actual IPsec connection between remote nodes and networks.

The Fedora implementation of IPsec uses IKE for sharing keys between hosts across the Internet. The **racoon** keying daemon handles the IKE key distribution and exchange. Refer to the **racoon** man page for more information about this daemon.

4.2.1.5. IPsec Installation

Implementing IPsec requires that the **ipsec-tools** RPM package be installed on all IPsec hosts (if using a host-to-host configuration) or routers (if using a network-to-network configuration). The RPM package contains essential libraries, daemons, and configuration files for setting up the IPsec connection, including:

- **/sbin/setkey** — manipulates the key management and security attributes of IPsec in the kernel. This executable is controlled by the **racoon** key management daemon. Refer to the **setkey(8)** man page for more information.
- **/usr/sbin/racoon** — the IKE key management daemon, used to manage and control security associations and key sharing between IPsec-connected systems.
- **/etc/racoon/racoon.conf** — the **racoon** daemon configuration file used to configure various aspects of the IPsec connection, including authentication methods and encryption algorithms used in the connection. Refer to the **racoon.conf(5)** man page for a complete listing of available directives.

To configure IPsec on Fedora, you can use the **Network Administration Tool**, or manually edit the networking and IPsec configuration files.

- To connect two network-connected hosts via IPsec, refer to [Section 4.2.1.6, “IPsec Host-to-Host Configuration”](#).
- To connect one LAN/WAN to another via IPsec, refer to [Section 4.2.1.7, “IPsec Network-to-Network Configuration”](#).

4.2.1.6. IPsec Host-to-Host Configuration

IPsec can be configured to connect one desktop or workstation (host) to another using a host-to-host connection. This type of connection uses the network to which each host is connected to create a secure tunnel between each host. The requirements of a host-to-host connection are minimal, as is the configuration of IPsec on each host. The hosts need only a dedicated connection to a carrier network (such as the Internet) and Fedora to create the IPsec connection.

4.2.1.6.1. Host-to-Host Connection

A host-to-host IPsec connection is an encrypted connection between two systems, both running IPsec with the same authentication key. With the IPsec connection active, any network traffic between the two hosts is encrypted.

To configure a host-to-host IPsec connection, use the following steps for each host:

Note

You should perform the following procedures on the actual machine that you are configuring. Avoid attempting to configure and establish IPsec connections remotely.

1. In a command shell, type **system-config-network** to start the **Network Administration Tool**.
2. On the **IPsec** tab, click **New** to start the IPsec configuration wizard.
3. Click **Forward** to start configuring a host-to-host IPsec connection.
4. Enter a unique name for the connection, for example, **ipsec0**. If required, select the check box to automatically activate the connection when the computer starts. Click **Forward** to continue.
5. Select **Host to Host encryption** as the connection type, and then click **Forward**.
6. Select the type of encryption to use: manual or automatic.

If you select manual encryption, an encryption key must be provided later in the process. If you select automatic encryption, the **racoon** daemon manages the encryption key. The **ipsec-tools** package must be installed if you want to use automatic encryption.

Click **Forward** to continue.

7. Enter the IP address of the remote host.

To determine the IP address of the remote host, use the following command *on the remote host*:

```
[root@myServer ~] # /sbin/ifconfig <device>
```

where *<device>* is the Ethernet device that you want to use for the VPN connection.

If only one Ethernet card exists in the system, the device name is typically `eth0`. The following example shows the relevant information from this command (note that this is an example output only):

```
eth0      Link encap:Ethernet  HWaddr 00:0C:6E:E8:98:1D
```

```
inet addr:172.16.44.192 Bcast:172.16.45.255 Mask:255.255.254.0
```

The IP address is the number following the **inet addr:** label.



Note

For host-to-host connections, both hosts should have a public, routable address. Alternatively, both hosts can have a private, non-routable address (for example, from the 10.x.x.x or 192.168.x.x ranges) as long as they are on the same LAN.

If the hosts are on different LANs, or one has a public address while the other has a private address, refer to [Section 4.2.1.7, “IPsec Network-to-Network Configuration”](#).

Click **Forward** to continue.

8. If manual encryption was selected in step 6, specify the encryption key to use, or click **Generate** to create one.
 - a. Specify an authentication key or click **Generate** to generate one. It can be any combination of numbers and letters.
 - b. Click **Forward** to continue.
9. Verify the information on the **IPsec — Summary** page, and then click **Apply**.
10. Click **File => Save** to save the configuration.

You may need to restart the network for the changes to take effect. To restart the network, use the following command:

```
[root@myServer ~]# service network restart
```

11. Select the IPsec connection from the list and click the **Activate** button.
12. Repeat the entire procedure for the other host. It is essential that the same keys from step 8 be used on the other hosts. Otherwise, IPsec will not work.

After configuring the IPsec connection, it appears in the IPsec list as shown in [Figure 4.1, “IPsec Connection”](#).

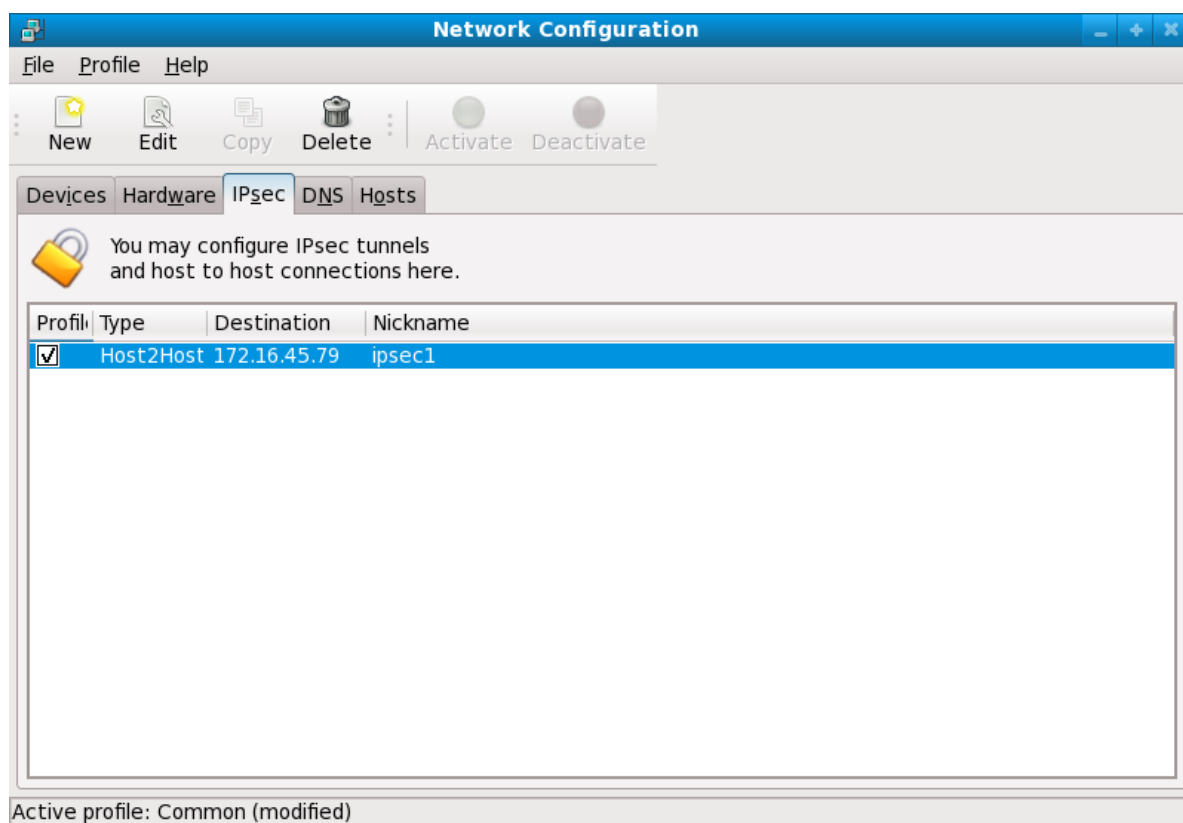


Figure 4.1. IPsec Connection

The following files are created when the IPsec connection is configured:

- `/etc/sysconfig/network-scripts/ifcfg-<nickname>`
- `/etc/sysconfig/network-scripts/keys-<nickname>`
- `/etc/racoon/<remote-ip>.conf`
- `/etc/racoon/psk.txt`

If automatic encryption is selected, `/etc/racoon/racoon.conf` is also created.

When the interface is up, `/etc/racoon/racoon.conf` is modified to include `<remote-ip>.conf`.

4.2.1.6.2. Manual IPsec Host-to-Host Configuration

The first step in creating a connection is to gather system and network information from each workstation. For a host-to-host connection, you need the following:

- The IP address of each host
- A unique name, for example, **ipsec1**. This is used to identify the IPsec connection and to distinguish it from other devices or connections.
- A fixed encryption key or one automatically generated by **racoon**.
- A pre-shared authentication key that is used during the initial stage of the connection and to exchange encryption keys during the session.

For example, suppose Workstation A and Workstation B want to connect to each other through an IPsec tunnel. They want to connect using a pre-shared key with the value of **Key_Value01**, and the

users agree to let **racoon** automatically generate and share an authentication key between each host. Both host users decide to name their connections **ipsec1**.

Note

You should choose a PSK that uses a mixture of upper- and lower-case characters, numbers and punctuation. An easily-guessable PSK constitutes a security risk.

It is not necessary to use the same connection name for each host. You should choose a name that is convenient and meaningful for your installation.

The following is the IPsec configuration file for Workstation A for a host-to-host IPsec connection with Workstation B. The unique name to identify the connection in this example is *ipsec1*, so the resulting file is called **/etc/sysconfig/network-scripts/ifcfg-ipsec1**.

```
DST=X.X.X.XTYPE=IPSEC
ONBOOT=no
IKE_METHOD=PSK
```

For Workstation A, *X.X.X.X* is the IP address of Workstation B. For Workstation B, *X.X.X.X* is the IP address of Workstation A. This connection is not set to initiate on boot-up (**ONBOOT=no**) and it uses the pre-shared key method of authentication (**IKE_METHOD=PSK**).

The following is the content of the pre-shared key file (called **/etc/sysconfig/network-scripts/keys-ipsec1**) that both workstations need to authenticate each other. The contents of this file should be identical on both workstations, and only the root user should be able to read or write this file.

```
IKE_PSK=Key_Value01
```

Important

To change the **keys-ipsec1** file so that only the root user can read or edit the file, use the following command after creating the file:

```
[root@myServer ~] # chmod 600 /etc/sysconfig/network-scripts/keys-ipsec1
```

To change the authentication key at any time, edit the **keys-ipsec1** file on both workstations. *Both authentication keys must be identical for proper connectivity.*

The next example shows the specific configuration for the phase 1 connection to the remote host. The file is called **X.X.X.X.conf**, where *X.X.X.X* is the IP address of the remote IPsec host. Note that this file is automatically generated when the IPsec tunnel is activated and should not be edited directly.

```
remote X.X.X.X{
    exchange_mode aggressive, main;
    my_identifier address;
    proposal {
```

```
    encryption_algorithm 3des;
    hash_algorithm sha1;
    authentication_method pre_shared_key;
    dh_group 2 ;
  }
}
```

The default phase 1 configuration file that is created when an IPsec connection is initialized contains the following statements used by the Fedora implementation of IPsec:

`remote X.X.X.X`

Specifies that the subsequent stanzas of this configuration file apply only to the remote node identified by the `X.X.X.X` IP address.

`exchange_mode aggressive`

The default configuration for IPsec on Fedora uses an aggressive authentication mode, which lowers the connection overhead while allowing configuration of several IPsec connections with multiple hosts.

`my_identifier address`

Specifies the identification method to use when authenticating nodes. Fedora uses IP addresses to identify nodes.

`encryption_algorithm 3des`

Specifies the encryption cipher used during authentication. By default, *Triple Data Encryption Standard* (3DES) is used.

`hash_algorithm sha1;`

Specifies the hash algorithm used during phase 1 negotiation between nodes. By default, Secure Hash Algorithm version 1 is used.

`authentication_method pre_shared_key`

Specifies the authentication method used during node negotiation. By default, Fedora uses pre-shared keys for authentication.

`dh_group 2`

Specifies the Diffie-Hellman group number for establishing dynamically-generated session keys. By default, modp1024 (group 2) is used.

4.2.1.6.2.1. The Racoon Configuration File

The `/etc/racoon/racoon.conf` files should be identical on all IPsec nodes *except* for the **`include "/etc/racoon/X.X.X.X.conf"`** statement. This statement (and the file it references) is generated when the IPsec tunnel is activated. For Workstation A, the `X.X.X.X` in the **`include`** statement is Workstation B's IP address. The opposite is true of Workstation B. The following shows a typical **`racoon.conf`** file when the IPsec connection is activated.

```
# Racoon IKE daemon configuration file.
# See 'man racoon.conf' for a description of the format and entries.

path include "/etc/racoon";
path pre_shared_key "/etc/racoon/psk.txt";
path certificate "/etc/racoon/certs";

sainfo anonymous
{
    pfs_group 2;
    lifetime time 1 hour ;
    encryption_algorithm 3des, blowfish 448, rijndael ;
}
```



```

    authentication_algorithm hmac_sha1, hmac_md5 ;
    compression_algorithm deflate ;
}
include "/etc/racoon/X.X.X.X.conf";

```

This default **racoon.conf** file includes defined paths for IPsec configuration, pre-shared key files, and certificates. The fields in **sainfo anonymous** describe the phase 2 SA between the IPsec nodes — the nature of the IPsec connection (including the supported encryption algorithms used) and the method of exchanging keys. The following list defines the fields of phase 2:

sainfo anonymous

Denotes that SA can anonymously initialize with any peer provided that the IPsec credentials match.

pfs_group 2

Defines the Diffie-Hellman key exchange protocol, which determines the method by which the IPsec nodes establish a mutual temporary session key for the second phase of IPsec connectivity. By default, the Fedora implementation of IPsec uses group 2 (or **modp1024**) of the Diffie-Hellman cryptographic key exchange groups. Group 2 uses a 1024-bit modular exponentiation that prevents attackers from decrypting previous IPsec transmissions even if a private key is compromised.

lifetime time 1 hour

This parameter specifies the lifetime of an SA and can be quantified either by time or by bytes of data. The default Fedora implementation of IPsec specifies a one hour lifetime.

encryption_algorithm 3des, blowfish 448, rijndael

Specifies the supported encryption ciphers for phase 2. Fedora supports 3DES, 448-bit Blowfish, and Rijndael (the cipher used in the *Advanced Encryption Standard*, or AES).

authentication_algorithm hmac_sha1, hmac_md5

Lists the supported hash algorithms for authentication. Supported modes are sha1 and md5 hashed message authentication codes (HMAC).

compression_algorithm deflate

Defines the Deflate compression algorithm for IP Payload Compression (IPCOMP) support, which allows for potentially faster transmission of IP datagrams over slow connections.

To start the connection, use the following command on each host:

```
[root@myServer ~]# /sbin/ifup <nickname>
```

where <nickname> is the name you specified for the IPsec connection.

To test the IPsec connection, run the **tcpdump** utility to view the network packets being transferred between the hosts and verify that they are encrypted via IPsec. The packet should include an AH header and should be shown as ESP packets. ESP means it is encrypted. For example:

```

[root@myServer ~]# tcpdump -n -i eth0 host <targetSystem>

IP 172.16.45.107 > 172.16.44.192: AH(spi=0x0954ccb6, seq=0xbb): ESP(spi=0x0c9f2164, seq=0xbb)

```

4.2.1.7. IPsec Network-to-Network Configuration

IPsec can also be configured to connect an entire network (such as a LAN or WAN) to a remote network using a network-to-network connection. A network-to-network connection requires the

setup of IPsec routers on each side of the connecting networks to transparently process and route information from one node on a LAN to a node on a remote LAN. [Figure 4.2, “A network-to-network IPsec tunneled connection”](#) shows a network-to-network IPsec tunneled connection.

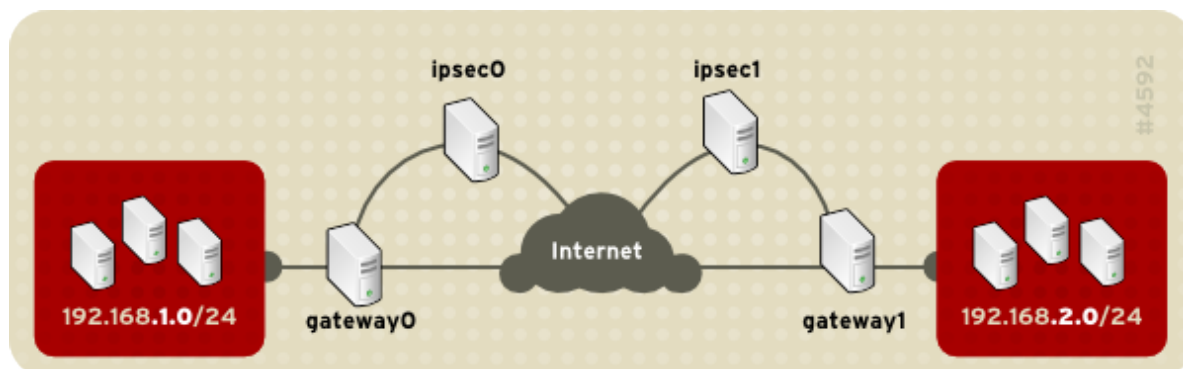


Figure 4.2. A network-to-network IPsec tunneled connection

This diagram shows two separate LANs separated by the Internet. These LANs use IPsec routers to authenticate and initiate a connection using a secure tunnel through the Internet. Packets that are intercepted in transit would require brute-force decryption in order to crack the cipher protecting the packets between these LANs. The process of communicating from one node in the 192.168.1.0/24 IP range to another in the 192.168.2.0/24 range is completely transparent to the nodes as the processing, encryption/decryption, and routing of the IPsec packets are completely handled by the IPsec router.

The information needed for a network-to-network connection include:

- The externally-accessible IP addresses of the dedicated IPsec routers
- The network address ranges of the LAN/WAN served by the IPsec routers (such as 192.168.1.0/24 or 10.0.1.0/24)
- The IP addresses of the gateway devices that route the data from the network nodes to the Internet
- A unique name, for example, **ipsec1**. This is used to identify the IPsec connection and to distinguish it from other devices or connections.
- A fixed encryption key or one automatically generated by **racoon**
- A pre-shared authentication key that is used during the initial stage of the connection and to exchange encryption keys during the session.

4.2.1.7.1. Network-to-Network (VPN) Connection

A network-to-network IPsec connection uses two IPsec routers, one for each network, through which the network traffic for the private subnets is routed.

For example, as shown in [Figure 4.3, “Network-to-Network IPsec”](#), if the 192.168.1.0/24 private network sends network traffic to the 192.168.2.0/24 private network, the packets go through gateway0, to ipsec0, through the Internet, to ipsec1, to gateway1, and to the 192.168.2.0/24 subnet.

IPsec routers require publicly addressable IP addresses and a second Ethernet device connected to their respective private networks. Traffic only travels through an IPsec router if it is intended for another IPsec router with which it has an encrypted connection.

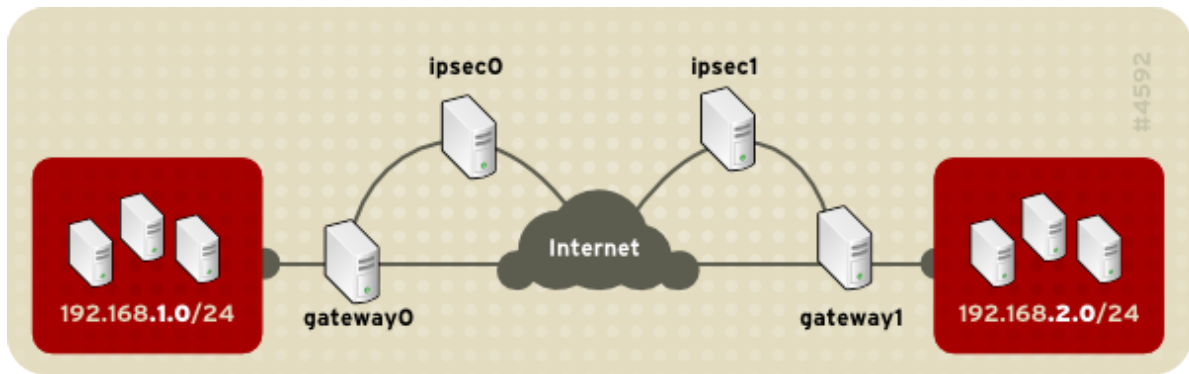


Figure 4.3. Network-to-Network IPsec

Alternate network configuration options include a firewall between each IP router and the Internet, and an intranet firewall between each IPsec router and subnet gateway. The IPsec router and the gateway for the subnet can be one system with two Ethernet devices: one with a public IP address that acts as the IPsec router; and one with a private IP address that acts as the gateway for the private subnet. Each IPsec router can use the gateway for its private network or a public gateway to send the packets to the other IPsec router.

Use the following procedure to configure a network-to-network IPsec connection:

1. In a command shell, type **system-config-network** to start the **Network Administration Tool**.
2. On the **IPsec** tab, click **New** to start the IPsec configuration wizard.
3. Click **Forward** to start configuring a network-to-network IPsec connection.
4. Enter a unique nickname for the connection, for example, **ipsec0**. If required, select the check box to automatically activate the connection when the computer starts. Click **Forward** to continue.
5. Select **Network to Network encryption (VPN)** as the connection type, and then click **Forward**.
6. Select the type of encryption to use: manual or automatic.

If you select manual encryption, an encryption key must be provided later in the process. If you select automatic encryption, the **racoona** daemon manages the encryption key. The **ipsec-tools** package must be installed if you want to use automatic encryption.

Click **Forward** to continue.

7. On the **Local Network** page, enter the following information:
 - **Local Network Address** — The IP address of the device on the IPsec router connected to the private network.
 - **Local Subnet Mask** — The subnet mask of the local network IP address.
 - **Local Network Gateway** — The gateway for the private subnet.

Click **Forward** to continue.

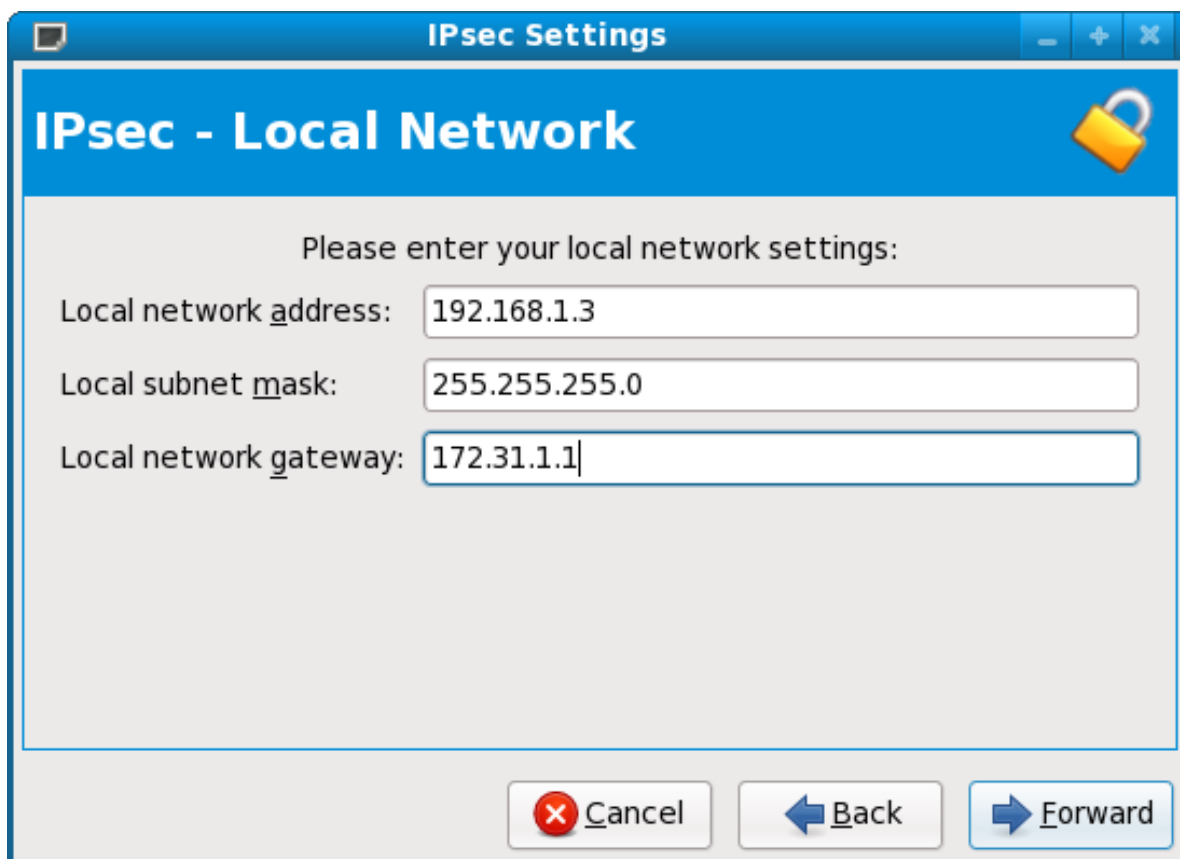
The image shows a Windows-style dialog box titled "IPsec Settings" with a blue header bar. Below the header, the title "IPsec - Local Network" is displayed in large white text on a blue background, accompanied by a yellow padlock icon. The main area of the dialog is light gray and contains the instruction "Please enter your local network settings:". Below this, there are three text input fields: "Local network address:" with the value "192.168.1.3", "Local subnet mask:" with the value "255.255.255.0", and "Local network gateway:" with the value "172.31.1.1". At the bottom of the dialog, there are three buttons: "Cancel" (with a red X icon), "Back" (with a blue left arrow icon), and "Forward" (with a blue right arrow icon).

Figure 4.4. Local Network Information

8. On the **Remote Network** page, enter the following information:
- **Remote IP Address** — The publicly addressable IP address of the IPsec router for the *other* private network. In our example, for ipsec0, enter the publicly addressable IP address of ipsec1, and vice versa.
 - **Remote Network Address** — The network address of the private subnet behind the *other* IPsec router. In our example, enter **192.168.1.0** if configuring ipsec1, and enter **192.168.2.0** if configuring ipsec0.
 - **Remote Subnet Mask** — The subnet mask of the remote IP address.
 - **Remote Network Gateway** — The IP address of the gateway for the remote network address.
 - If manual encryption was selected in step 6, specify the encryption key to use or click **Generate** to create one.

Specify an authentication key or click **Generate** to generate one. This key can be any combination of numbers and letters.

Click **Forward** to continue.

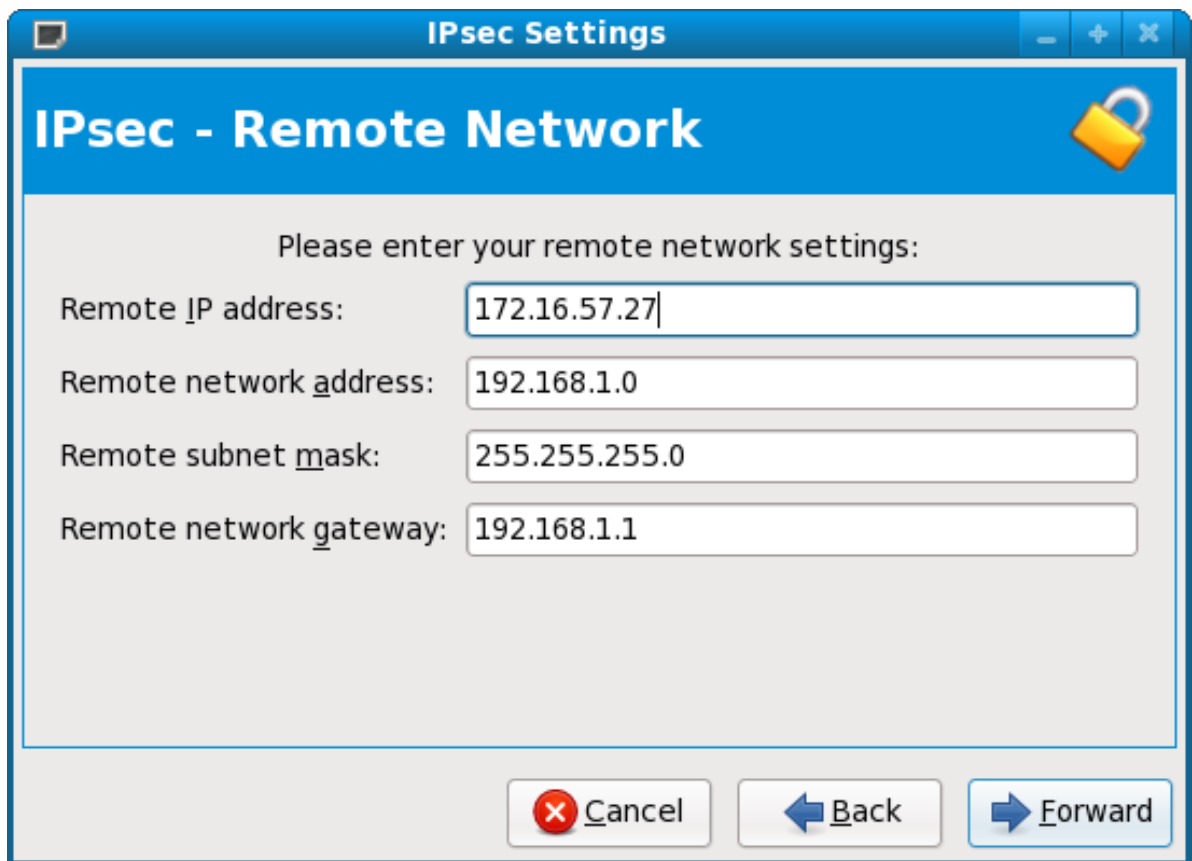


Figure 4.5. Remote Network Information

9. Verify the information on the **IPsec — Summary** page, and then click **Apply**.
10. Select **File => Save** to save the configuration.
11. Select the IPsec connection from the list, and then click **Activate** to activate the connection.
12. Enable IP forwarding:
 - a. Edit `/etc/sysctl.conf` and set `net.ipv4.ip_forward` to **1**.
 - b. Use the following command to enable the change:

```
[root@myServer ~]# /sbin/sysctl -p /etc/sysctl.conf
```

The network script to activate the IPsec connection automatically creates network routes to send packets through the IPsec router if necessary.

4.2.1.7.2. Manual IPsec Network-to-Network Configuration

Suppose LAN A (lana.example.com) and LAN B (lanb.example.com) want to connect to each other through an IPsec tunnel. The network address for LAN A is in the 192.168.1.0/24 range, while LAN B uses the 192.168.2.0/24 range. The gateway IP address is 192.168.1.254 for LAN A and 192.168.2.254 for LAN B. The IPsec routers are separate from each LAN gateway and use two network devices: eth0 is assigned to an externally-accessible static IP address which accesses the Internet, while eth1 acts as a routing point to process and transmit LAN packets from one network node to the remote network nodes.

The IPsec connection between each network uses a pre-shared key with the value of **r3dh4t11nux**, and the administrators of A and B agree to let **racoon** automatically generate and share an authentication key between each IPsec router. The administrator of LAN A decides to name the IPsec connection **ipsec0**, while the administrator of LAN B names the IPsec connection **ipsec1**.

The following example shows the contents of the **ifcfg** file for a network-to-network IPsec connection for LAN A. The unique name to identify the connection in this example is *ipsec0*, so the resulting file is called **/etc/sysconfig/network-scripts/ifcfg-ipsec0**.

```
TYPE=IPSEC
ONBOOT=yes
IKE_METHOD=PSK
SRCGW=192.168.1.254
DSTGW=192.168.2.254
SRCNET=192.168.1.0/24
DSTNET=192.168.2.0/24
DST=X.X.X.X
```

The following list describes the contents of this file:

TYPE=IPSEC

Specifies the type of connection.

ONBOOT=yes

Specifies that the connection should initiate on boot-up.

IKE_METHOD=PSK

Specifies that the connection uses the pre-shared key method of authentication.

SRCGW=192.168.1.254

The IP address of the source gateway. For LAN A, this is the LAN A gateway, and for LAN B, the LAN B gateway.

DSTGW=192.168.2.254

The IP address of the destination gateway. For LAN A, this is the LAN B gateway, and for LAN B, the LAN A gateway.

SRCNET=192.168.1.0/24

Specifies the source network for the IPsec connection, which in this example is the network range for LAN A.

DSTNET=192.168.2.0/24

Specifies the destination network for the IPsec connection, which in this example is the network range for LAN B.

DST=X.X.X.X

The externally-accessible IP address of LAN B.

The following example is the content of the pre-shared key file called **/etc/sysconfig/network-scripts/keys-ipsecX** (where X is 0 for LAN A and 1 for LAN B) that both networks use to authenticate each other. The contents of this file should be identical and only the root user should be able to read or write this file.

```
IKE_PSK=r3dh4t11nux
```



Important

To change the **keys-ipsecX** file so that only the root user can read or edit the file, use the following command after creating the file:

```
chmod 600 /etc/sysconfig/network-scripts/keys-ipsec1
```

To change the authentication key at any time, edit the **keys-ipsecX** file on both IPsec routers. *Both keys must be identical for proper connectivity.*

The following example is the contents of the **/etc/racoon/racoon.conf** configuration file for the IPsec connection. Note that the **include** line at the bottom of the file is automatically generated and only appears if the IPsec tunnel is running.

```
# Racoon IKE daemon configuration file.
# See 'man racoon.conf' for a description of the format and entries.
path include "/etc/racoon";
path pre_shared_key "/etc/racoon/psk.txt";
path certificate "/etc/racoon/certs";

sainfo anonymous
{
  pfs_group 2;
  lifetime time 1 hour ;
  encryption_algorithm 3des, blowfish 448, rijndael ;
  authentication_algorithm hmac_sha1, hmac_md5 ;
  compression_algorithm deflate ;
}
include "/etc/racoon/X.X.X.X.conf"
```

The following is the specific configuration for the connection to the remote network. The file is called **X.X.X.X.conf** (where X.X.X.X is the IP address of the remote IPsec router). Note that this file is automatically generated when the IPsec tunnel is activated and should not be edited directly.

```
remote X.X.X.X{
  exchange_mode aggressive, main;
  my_identifier address;
  proposal {
    encryption_algorithm 3des;
    hash_algorithm sha1;
    authentication_method pre_shared_key;
    dh_group 2 ;
  }
}
```

Prior to starting the IPsec connection, IP forwarding should be enabled in the kernel. To enable IP forwarding:

1. Edit **/etc/sysctl.conf** and set **net.ipv4.ip_forward** to **1**.
2. Use the following command to enable the change:

```
[root@myServer ~] # sysctl -p /etc/sysctl.conf
```

Chapter 4. Encryption

To start the IPsec connection, use the following command on each router:

```
[root@myServer ~] # /sbin/ifup ipsec0
```

The connections are activated, and both LAN A and LAN B are able to communicate with each other. The routes are created automatically via the initialization script called by running **ifup** on the IPsec connection. To show a list of routes for the network, use the following command:

```
[root@myServer ~] # /sbin/ip route list
```

To test the IPsec connection, run the **tcpdump** utility on the externally-routable device (eth0 in this example) to view the network packets being transferred between the hosts (or networks), and verify that they are encrypted via IPsec. For example, to check the IPsec connectivity of LAN A, use the following command:

```
[root@myServer ~] # tcpdump -n -i eth0 host lana.example.com
```

The packet should include an AH header and should be shown as ESP packets. ESP means it is encrypted. For example (back slashes denote a continuation of one line):

```
12:24:26.155529 lanb.example.com > lana.example.com: AH(spi=0x021c9834, seq=0x358): \
lanb.example.com > lana.example.com: ESP(spi=0x00c887ad, seq=0x358) (DF) \
(ipip-proto-4)
```

4.2.1.8. Starting and Stopping an IPsec Connection

If the IPsec connection was not configured to activate on boot, you can control it from the command line.

To start the connection, use the following command on each host for host-to-host IPsec, or each IPsec router for network-to-network IPsec:

```
[root@myServer ~] # /sbin/ifup <nickname>
```

where *<nickname>* is the nickname configured earlier, such as **ipsec0**.

To stop the connection, use the following command:

```
[root@myServer ~] # /sbin/ifdown <nickname>
```

4.2.2. Secure Shell

Secure Shell (SSH) is a powerful network protocol used to communicate with another system over a secure channel. The transmissions over SSH are encrypted and protected from interception. Cryptographic log-on can also be utilized to provide a better authentication method over traditional usernames and passwords.

SSH is very easy to activate. By simply starting the **sshd** service, the system will begin to accept connections and will allow access to the system when a correct username and password is provided during the connection process. The standard TCP port for the SSH service is 22, however this can be changed by modifying the configuration file */etc/ssh/sshd_config* and restarting the service. This file also contains other configuration options for SSH.

Secure Shell (SSH) also provides encrypted tunnels between computers but only using a single port. *Port forwarding can be done over an SSH tunnel*¹ and traffic will be encrypted as it passes over that tunnel but using port forwarding is not as fluid as a VPN.

4.2.2.1. Cryptographic Logon

SSH supports the use of cryptographic keys to login to a computer. This is much more secure than using a password and if setup properly could be considered multifactor authentication.

A configuration change must occur before cryptographic logon can occur. In the file `/etc/ssh/sshd_config` uncomment and modify the following lines so that appear as such:

```
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
```

The first line tells the SSH program to allow public key authentication. The second line points to a file in the home directory where the public key of authorized key pairs exists on the system.

The next thing to do is to generate the ssh key pairs on the client you will use to connect to the system. The command **ssh-keygen** will generate an RSA 2048-bit key set for logging into the system. The keys are stored, by default, in the `~/.ssh` directory. You can utilize the switch **-b** to modify the bit-strength of the key. 2048-bits is probably okay but you can go up to, and possibly beyond, 8192-bit keys.

In your `~/.ssh` directory you should see the two keys you just created. If you accepted the defaults when running the **ssh-keygen** then your keys are named **id_rsa** and **id_rsa.pub**, the private and public keys. You should always protect the private key from exposure. The public key, however, needs to be transferred over to the system you are going to login to. Once you have it on your system the easiest way to add the key to the approved list is by:

```
$ cat id_rsa.pub >> ~/.ssh/authorized_keys
```

This will append the public key to the `authorized_key` file. The **SSH** application will check this file when you attempt to login to the computer.

Similarly to passwords and any other authentication mechanism, you should change your **SSH** keys regularly. When you do make sure you clean out any unused key from the `authorized_key` file.

4.2.3. LUKS Disk Encryption

Linux Unified Key Setup-on-disk-format (or LUKS) allows you to encrypt partitions on your Linux computer. This is particularly important when it comes to mobile computers and removable media. LUKS allows multiple user keys to decrypt a master key which is used for the bulk encryption of the partition.

4.2.3.1. LUKS Implementation in Fedora

Fedora utilizes LUKS to perform file system encryption. By default, the option to encrypt the file system is unchecked during the installation. If you select the option to encrypt you hard drive, you will be prompted for a passphrase that will be asked every time you boot the computer. This passphrase

¹ <http://www.redhatmagazine.com/2007/11/27/advanced-ssh-configuration-and-tunneling-we-dont-need-no-stinking-vpn-software>

"unlocks" the bulk encryption key that is used to decrypt your partition. If you choose to modify the default partition table you can choose which partitions you want to encrypt. This is set in the partition table settings

Fedora's default implementation of LUKS is AES 128 with a SHA256 hashing. Ciphers that are available are:

- AES - Advanced Encryption Standard - [FIPS PUB 197](#)²
- Twofish (A 128-bit Block Cipher)
- Serpent
- cast5 - [RFC 2144](#)³
- cast6 - [RFC 2612](#)⁴

4.2.3.2. Manually Encrypting Directories



Warning

Following this procedure will remove all data on the partition that you are encrypting. You WILL lose all your information! Make sure you backup your data to an external source before beginning this procedure!



Note

This procedure uses *scrub* to destroy the existing data on the partition and provide a random base for LUKS to use. This random base is important to prevent certain attacks against the cryptography. *Scrub* is not installed by default and you will have to install it before use. Alternatively you may use another random number generator to accomplish the same thing.

If you want to manually encrypt a partition the following directions are for you. The below example demonstrates encrypting your /home partition but any partition can be used.

The following procedure will wipe all your existing data, so be sure to have a tested backup before you start. This also requires you to have a separate partition for /home (in my case that is /dev/VG00/LV_home). All the following must be done as root. Any of these steps failing means you must not continue until the step succeeded.

4.2.3.3. Step-by-Step Instructions

1. enter runlevel 1: **telinit 1**

² <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>

³ <http://www.ietf.org/rfc/rfc2144.txt>

⁴ <http://www.ietf.org/rfc/rfc2612.txt>

2. Fill your partition with random data: **scrub -p random /home**
3. unmount your existing /home: **umount /home**
4. if it fails use **fuser** to find and kill processes hogging /home: **fuser -mvk /home**
5. verify /home is not mounted any longer: **cat /proc/mounts | grep home**
6. initialize your partition: **cryptsetup --verbose --verify-passphrase luksFormat /dev/VG00/LV_home**
7. open the newly encrypted device: **cryptsetup luksOpen /dev/VG00/LV_home home**
8. check it's there: **ls -l /dev/mapper | grep home**
9. create a filesystem: **mkfs.ext3 /dev/mapper/home**
10. mount it: **mount /dev/mapper/home /home**
11. check it's visible: **df -h | grep home**
12. add the following to /etc/crypttab: **home /dev/VG00/LV_home none**
13. edit your /etc/fstab, removing the old entry for /home and adding **/dev/mapper/home /home ext3 defaults 1 2**
14. verify your fstab entry: **mount /home**
15. restore default SELinux security contexts: **/sbin/restorecon -v -R /home**
16. reboot: **shutdown -r now**
17. The entry in /etc/crypttab makes your computer ask your **luks** passphrase on boot
18. Login as root and restore your backup

4.2.3.4. What you have just accomplished.

Congratulations, you now have an encrypted partition for all of your data to safely rest while the computer is off.

4.2.3.5. Links of Interest

For additional information on LUKS or encrypting hard drives please visit one of the following links:

- [LUKS - Linux Unified Key Setup](https://code.google.com/p/cryptsetup/)⁵
- [HOWTO: Creating an encrypted Physical Volume \(PV\) using a second hard drive, pvmove, and a Fedora LiveCD](https://bugzilla.redhat.com/attachment.cgi?id=161912)⁶

4.2.4. 7-Zip Encrypted Archives

7-Zip⁷ is a cross-platform, next generation, file compression tool that can also use strong encryption (AES-256) to protect the contents of the archive. This is extremely useful when you need to move data

⁵ <https://code.google.com/p/cryptsetup/>

⁶ <https://bugzilla.redhat.com/attachment.cgi?id=161912>

⁷ <http://www.7-zip.org/>

between multiple computers that use varying operating systems (i.e. Linux at home, Windows at work) and you want a portable encryption solution.

4.2.4.1. 7-Zip Installation

7-Zip is not a base package in Fedora, but it is available in the software repository. Once installed, the package will update alongside the rest of the software on the computer with no special attention necessary.

4.2.4.2. Step-by-Step Installation Instructions

- Open a Terminal: **Click Applications -> System Tools -> Terminal** or in GNOME 3: **Activities -> Applications -> Terminal**
- Install 7-Zip with sudo access: **sudo yum install p7zip**
- Close the Terminal: **exit**

4.2.4.3. Step-by-Step Usage Instructions

By following these instructions you are going to compress and encrypt your "Documents" directory. Your original "Documents" directory will remain unaltered. This technique can be applied to any directory or file you have access to on the filesystem.

- Open a Terminal: **Click Applications -> System Tools -> Terminal**
- Compress and Encrypt: (enter a password when prompted) **7za a -mhe=on -ms=on -p Documents.7z Documents/**

The "Documents" directory is now compressed and encrypted. The following instructions will move the encrypted archive somewhere new and then extract it.

- Create a new directory: **mkdir newplace**
- Move the encrypted file: **mv Documents.7z newplace**
- Go to the new directory: **cd newplace**
- Extract the file: (enter the password when prompted) **7za x Documents.7z**

The archive is now extracted into the new location. The following instructions will clean up all the prior steps and restore your computer to its previous state.

- Go up a directory: **cd . .**
- Delete the test archive and test extraction: **rm -r newplace**
- Close the Terminal: **exit**

4.2.4.4. Creating a Secure 7-Zip Archive via the GUI

7-Zip archives can be extracted just like any other archive via the GUI, but creating a secure 7-Zip archive requires a few additional steps.

By following these instructions you are going to compress and encrypt your "Documents" directory. Your original "Documents" directory will remain unaltered. This technique can be applied to any directory or file you have access to on the filesystem.

- Open the file browser: Click Activities -> Files
- Right-Click on the "Documents" folder
- Select the "Compress" option
- Select ".7z" as the file extension
- Expand "Other Options"
- Check "Encrypt the file list too"
- Enter a password into the password field
- Click the "Create" button

You will now see a "Documents.7z" file appear in your home directory. If you try to open the file, you will be asked for the archive password before being shown the contents of the archive. The file will open once the correct password is supplied, and the archive can then be manipulated as usual. Deleting the "Documents.7z" file will conclude this exercise and return your computer to its previous state.

4.2.4.5. Things of note

7-Zip is not shipped by default with Microsoft Windows or Mac OS X. If you need to use your 7-Zip files on those platforms you will need to install the appropriate version of 7-Zip on those computers. See the 7-Zip [download page](http://www.7-zip.org/download.html)⁸.

4.2.5. Using GNU Privacy Guard (GnuPG)

GnuPG (GPG) is used to identify yourself and authenticate your communications, including those with people you don't know. GPG allows anyone reading a GPG-signed email to verify its authenticity. In other words, GPG allows someone to be reasonably certain that communications signed by you actually are from you. GPG is useful because it helps prevent third parties from altering code or intercepting conversations and altering the message.

GPG can also be used to sign and/or encrypt files kept on your computer or on a network drive. This can add additional protection in preventing a file from being altered or read by unauthorized people.

To utilize GPG for authentication or encryption of email you must first generate your public and private keys. After generating the keys you will have to setup your email client to utilize them.

4.2.5.1. Generating GPG Keys in GNOME

The Seahorse utility makes GPG key management easier. You can install *Seahorse* at the command line with the command `su -c "yum install seahorse"` or in the GUI using **Add/Remove Software**.

To create a key select **Passwords and Keys**, which starts the application **Seahorse**. From the **File** menu select **New** then **PGP Key** then select **Continue**. Type your full name, email address, and an optional comment describing who are you (e.g.: John C. Smith, jsmith@example.com, The Man). Select **Create**. A dialog is displayed asking for a passphrase for the key. Choose a strong passphrase but also easy to remember. Click **OK** and the key is created.

⁸ <http://www.7-zip.org/download.html>



Warning

If you forget your passphrase, the key cannot be used and any data encrypted using that key will be lost.

To find your GPG key ID, look in the Key ID column next to the newly created key. In most cases, if you are asked for the key ID, you should prepend "0x" to the key ID, as in "0x6789ABCD". You should make a backup of your private key and store it somewhere secure.

4.2.5.2. Generating GPG Keys in KDE

Start the KGpg program from the main menu by selecting Applications > Utilities > Encryption Tool. If you have never used KGpg before, the program walks you through the process of creating your own GPG keypair. A dialog box appears prompting you to create a new key pair. Enter your name, email address, and an optional comment. You can also choose an expiration time for your key, as well as the key strength (number of bits) and algorithms. The next dialog box prompts you for your passphrase. At this point, your key appears in the main **KGpg** window.



Warning

If you forget your passphrase, the key cannot be used and any data encrypted using that key will be lost.

To find your GPG key ID, look in the Key ID column next to the newly created key. In most cases, if you are asked for the key ID, you should prepend "0x" to the key ID, as in "0x6789ABCD". You should make a backup of your private key and store it somewhere secure.

4.2.5.3. Generating GPG Keys Using the Command Line

Use the following shell command: **gpg --gen-key**

This command generates a key pair that consists of a public and a private key. Other people use your public key to authenticate and/or decrypt your communications. Distribute your public key as widely as possible, especially to people who you know will want to receive authentic communications from you, such as a mailing list.

A series of prompts directs you through the process. Press the **Enter** key to assign a default value if desired. The first prompt asks you to select what kind of key you prefer:

```
Please select what kind of key you want:
(1) RSA and RSA (default)
(2) DSA and Elgamal
(3) DSA (sign only)
(4) RSA (sign only)
Your selection?
```

In almost all cases, the default is the correct choice. A RSA key allows you not only to sign communications, but also to encrypt files.

Next, choose the key size:

```
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
```

Again, the default is sufficient for almost all users, and represents a strong level of security.

Next, choose when the key will expire. It is a good idea to choose an expiration date instead of using the default, which is none. If, for example, the email address on the key becomes invalid, an expiration date will remind others to stop using that public key.

```
Please specify how long the key should be valid.
 0 = key does not expire
 d = key expires in n days
 w = key expires in n weeks
 m = key expires in n months
 y = key expires in n years
Key is valid for? (0)
```

Entering a value of **1y**, for example, makes the key valid for one year. (You may change this expiration date after the key is generated, if you change your mind.)

Before the **gpgcode>** program asks for signature information, the following prompt appears: **Is this correct (y/n)?** Enter **ycode>** to finish the process.

Next, enter your name and email address. Remember this process is about authenticating you as a real individual. For this reason, include your real name. Do not use aliases or handles, since these disguise or obfuscate your identity.

Enter your real email address for your GPG key. If you choose a bogus email address, it will be more difficult for others to find your public key. This makes authenticating your communications difficult. If you are using this GPG key for `[[DocsProject/SelfIntroduction| self-introduction]]` on a mailing list, for example, enter the email address you use on that list.

Use the comment field to include aliases or other information. (Some people use different keys for different purposes and identify each key with a comment, such as "Office" or "Open Source Projects.")

At the confirmation prompt, enter the letter **O** to continue if all entries are correct, or use the other options to fix any problems. Finally, enter a passphrase for your secret key. The **gpg** program asks you to enter your passphrase twice to ensure you made no typing errors.

Finally, **gpg** generates random data to make your key as unique as possible. Move your mouse, type random keys, or perform other tasks on the system during this step to speed up the process. Once this step is finished, your keys are complete and ready to use:

```
pub 1024D/1B2AFA1C 2005-03-31 John Q. Doe <jqdoe@example.com>
Key fingerprint = 117C FE83 22EA B843 3E86 6486 4320 545E 1B2A FA1C
sub 1024g/CEA4B22E 2005-03-31 [expires: 2006-03-31]
```

The key fingerprint is a shorthand "signature" for your key. It allows you to confirm to others that they have received your actual public key without any tampering. You do not need to write this fingerprint down. To display the fingerprint at any time, use this command, substituting your email address: **gpg --fingerprint jqdoe@example.com**

Your "GPG key ID" consists of 8 hex digits identifying the public key. In the example above, the GPG key ID is 1B2AFA1C. In most cases, if you are asked for the key ID, you should prepend "0x" to the key ID, as in "0x1B2AFA1C".



Warning

If you forget your passphrase, the key cannot be used and any data encrypted using that key will be lost.

4.2.5.4. Using GPG with Alpine

If you are using the email client *Alpine* or *Pine* then you will also need to download and install *ez-pine-gpg*. This software is currently available from <http://business-php.com/opensource/ez-pine-gpg/>. Once you have installed *ez-pine-gpg* you will need to modify your `~/ .pinerc` file. You need to:

1. `/home/username/bin` should be replaced with the installation path that you specified.
2. In two places, the `gpg-identifier` after `_RECIPIENTS_` should be replaced with your GPG public key's identifier. The reason you include your own GPG identifier here is so that if you send an encrypted message to "Alice", that message is also encrypted with your public key -- if you don't do this, then you will not be able to open that message in your sent-mail folder and remind yourself of what you wrote.

It should look something like this:

```
# This variable takes a list of programs that message text is piped into
# after MIME decoding, prior to display.
display-filters=_LEADING("-----BEGIN PGP")_ /home/max/bin/ez-pine-gpg-incoming

# This defines a program that message text is piped into before MIME
# encoding, prior to sending
sending-filters=/home/max/bin/ez-pine-gpg-sign _INCLUDEALLHDRS_,
    /home/username/bin/ez-pine-gpg-encrypt _RECIPIENTS_ gpg-identifier,
    /home/username/bin/ez-pine-gpg-sign-and-encrypt _INCLUDEALLHDRS_ _RECIPIENTS_ gpg-
identifier
```

4.2.5.5. Using GPG with Evolution

4.2.5.5.1. Configuring GPG for use with Evolution

To configure GPG for use in **Evolution** select from the **Evolution** Main Menu, select Tools, Settings... In the left pane, select Mail Accounts. In the right pane, select the email account you use for correspondence. Then select the Edit button. The **Evolution** Account Editor dialog appears. Select the Security tab.

In the PGP/GPG Key ID field, enter the GPG key ID matching this account's email address. If you are not sure what your key ID is, use this command: **gpg --fingerprint EMAIL_ADDRESS**. The key ID is the same as the last eight characters (4 bytes) of the key fingerprint. It is a good idea to click the option Always encrypt to myself when sending encrypted mail. You may also want to select Always sign outgoing messages when using this account.



Notice

If you do not mark public keys as trusted in your keyring, you will not be able to encrypt email to their owners unless you select the option Always trust keys in my keyring when encrypting. You will instead receive a dialog indicating that a trust check has failed.

4.2.5.5.2. Verifying email with Evolution

Evolution will automatically check any incoming GPG-signed messages for validity. If Evolution cannot GPG verify a message due to a missing public key (or tampering), it will end with a red banner. If the message is verified but you have not signed the key either locally or globally, the banner will be yellow. If the message is verified and you have signed the key, the banner will be green. When you click the seal icon, Evolution displays a dialog with more security information about the signature. To add a public key to your keyring, use the search function along with the key owner's email address: **gpg --keyserver pgp.mit.edu --search email address**. To import the correct key, you may need to match the key ID with the information provided by Evolution.

4.2.5.5.3. Signing and Encrypting email with Evolution

Signing email allows the recipients to verify that the email actually came from you.

While composing your email, choose the Security menu, and then select PGP Sign to sign your message. To encrypt your message, select PGP Encrypt. You may sign an encrypted message as well, which is good practice. When you send the message, Evolution will ask you to enter your GPG key passphrase. (After three unsuccessful attempts Evolution generates an error.) If you select the option Remember this password for the remainder of this session, you will not need to use your passphrase again to sign or decrypt, unless you quit and restart Evolution.

4.2.5.6. Using GPG with Thunderbird

Fedora includes Mozilla Thunderbird in the thunderbird package, and the mozilla-mail package for the Mozilla Suite email application. Thunderbird is the recommended Mozilla email application. This appears on your desktop as Applications > Internet > Thunderbird Email.

Mozilla products support extensions, plugins that add new features to the main application. The Enigmail extensions provide GPG support to email products from Mozilla. Versions of Enigmail exist for both Mozilla Thunderbird, and the Mozilla Suite (Seamonkey). Netscape software from AOL is based on the Mozilla products, and may also use this extension.

To install Enigmail on Fedora systems, follow the instructions given below.

Enigmail uses the term OpenPGP in menu items and options. GPG is an implementation of OpenPGP, and you may treat the terms as equivalent.

The homepage for Enigmail is: <http://enigmail.mozdev.org/download.html>.

This page provides screenshots of Enigmail and GPG in action: <http://enigmail.mozdev.org/screenshots.html>.

4.2.5.6.1. Installing Enigmail

Enigmail is now available in fedora repository. It can be installed by typing: **yum install thunderbird-enigmail** at a command line. Alternatively, you can install *thunderbird-enigmail* using by going to **System -> Administration -> Add/Remove Software**.

4.2.5.7. About Public Key Encryption

1. [Wikipedia - Public Key Cryptography](http://en.wikipedia.org/wiki/Public-key_cryptography)⁹
2. [HowStuffWorks - Encryption](http://computer.howstuffworks.com/encryption.htm)¹⁰

⁹ http://en.wikipedia.org/wiki/Public-key_cryptography

¹⁰ <http://computer.howstuffworks.com/encryption.htm>

General Principles of Information Security

The following general principals provide an overview of good security practices:

- encrypt all data transmitted over networks to help prevent man-in-the-middle attacks and eavesdropping. It is important to encrypt authentication information, such as passwords.
- minimize the amount of software installed and running services.
- use security-enhancing software and tools, for example, Security-Enhanced Linux (SELinux) for Mandatory Access Control (MAC), Netfilter iptables for packet filtering (firewall), and the GNU Privacy Guard (GnuPG) for encrypting files.
- if possible, run each network service on a separate system to minimize the risk of one compromised service being used to compromise other services.
- maintain user accounts: create and enforce a strong password policy; delete unused user accounts.
- routinely review system and application logs. By default, security-relevant system logs are written to **/var/log/secure** and **/var/log/audit/audit.log**. Note: sending logs to a dedicated log server helps prevent attackers from easily modifying local logs to avoid detection.
- never log in as the root user unless absolutely necessary. It is recommended that administrators use **sudo** to execute commands as root when required. Users capable of running **sudo** are specified in **/etc/sudoers**. Use the **visudo** utility to edit **/etc/sudoers**.

Secure Installation

Security begins with the first time you put that CD or DVD into your disk drive to install Fedora. Configuring your system securely from the beginning makes it easier to implement additional security settings later.

6.1. Disk Partitions

The NSA recommends creating separate partitions for `/boot`, `/`, `/home`, `/tmp`, and `/var/tmp`. The reasons for each are different and we will address each partition.

`/boot` - This partition is the first partition that is read by the system during boot up. The boot loader and kernel images that are used to boot your system into Fedora are stored in this partition. This partition should not be encrypted. If this partition is included in `/` and that partition is encrypted or otherwise becomes unavailable then your system will not be able to boot.

`/home` - When user data (`/home`) is stored in `/` instead of in a separate partition, the partition can fill up causing the operating system to become unstable. Also, when upgrading your system to the next version of Fedora it is a lot easier when you can keep your data in the `/home` partition as it will not be overwritten during installation. If the root partition (`/`) becomes corrupt your data could be lost forever. By using a separate partition there is slightly more protection against data loss. You can also target this partition for frequent backups.

`/tmp` and `/var/tmp` - Both the `/tmp` and the `/var/tmp` directories are used to store data that doesn't need to be stored for a long period of time. However if a lot of data floods one of these directories it can consume all of your storage space. If this happens and these directories are stored within `/` then your system could become unstable and crash. For this reason, moving these directories into their own partitions is a good idea.

6.2. Utilize LUKS Partition Encryption

The implementation of *Linux Unified Key Setup-on-disk-format*¹ (LUKS) encryption has become a lot easier in recent years. During the installation process an option to encrypt your partitions will be presented to the user. The user must supply a passphrase that will be the key to unlock the bulk encryption key that will be used to secure the partition's data.

¹ http://fedoraproject.org/wiki/Security_Guide/9/LUKSDiskEncryption

