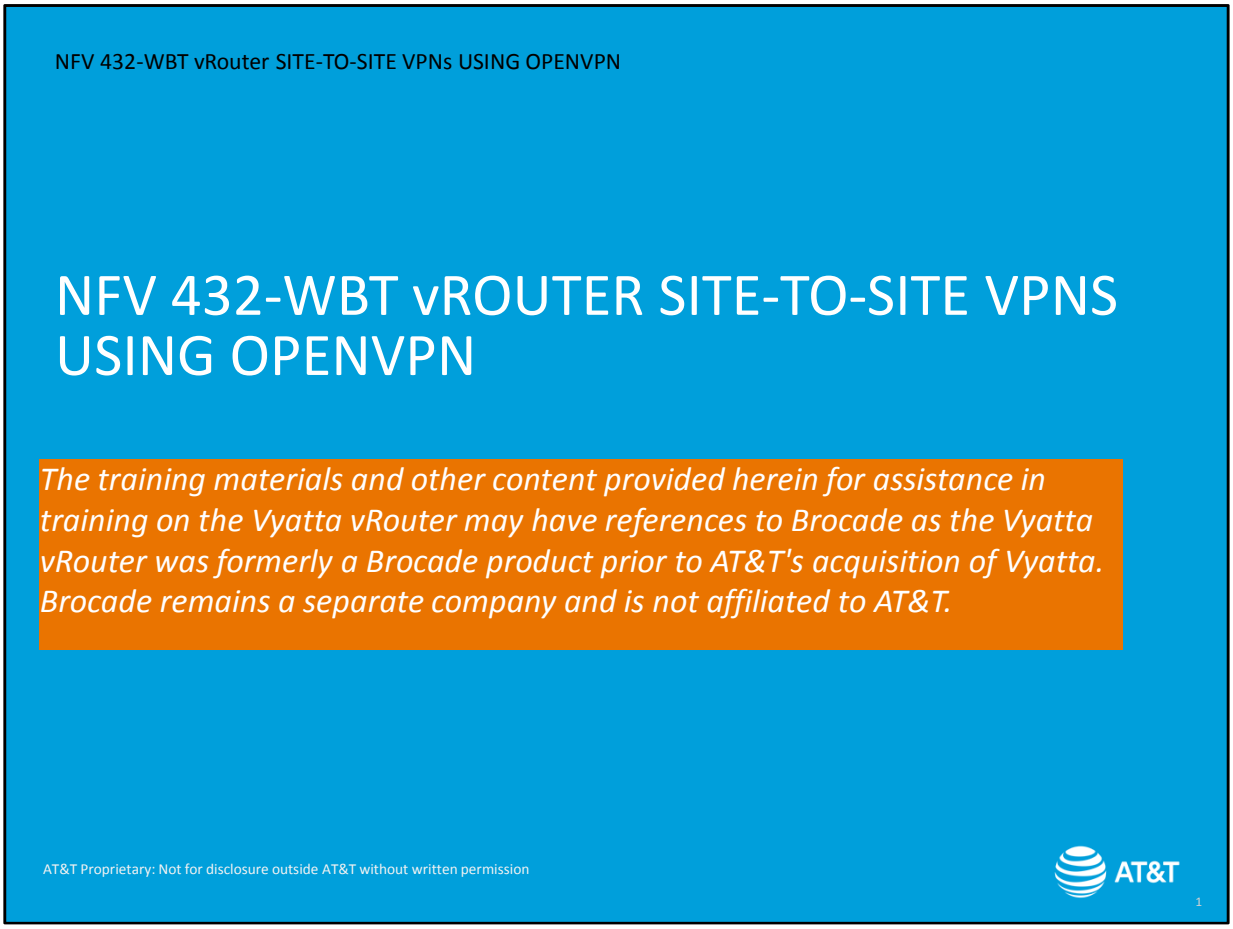


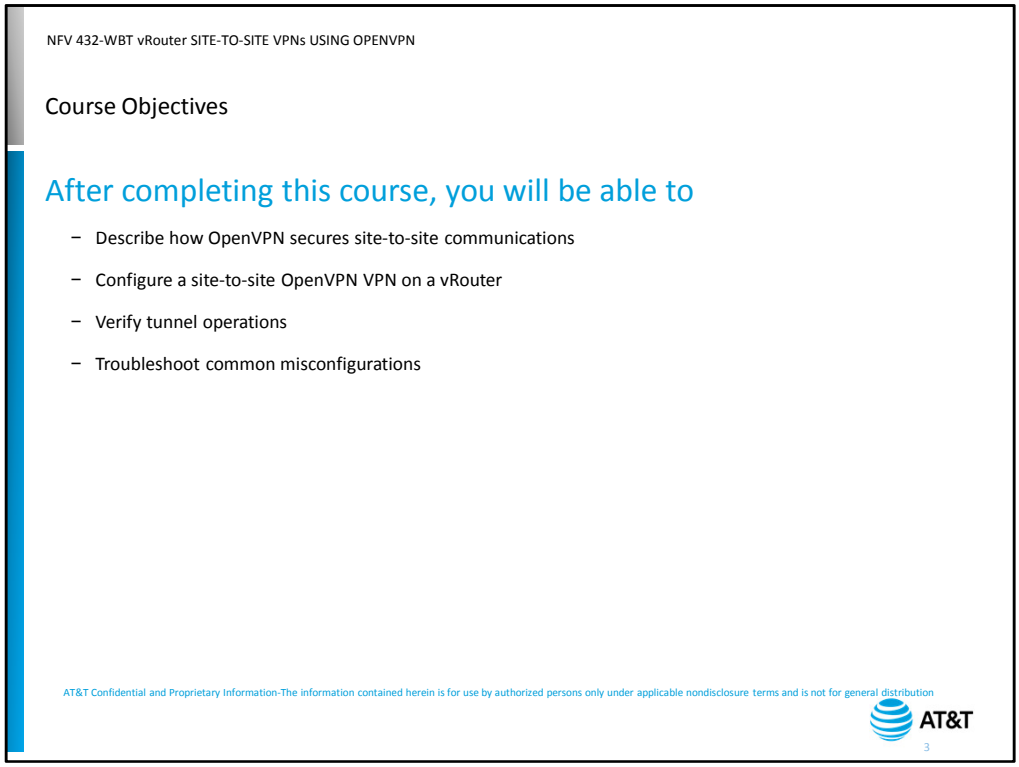
Welcome to the AT&T Vyatta 5600 vRouter Site-to-site VPNs using OpenVPN course.

NFV 432-WBT vRouter SITE-TO-SITE VPNs USING OPENVPN

Legal Disclaimer

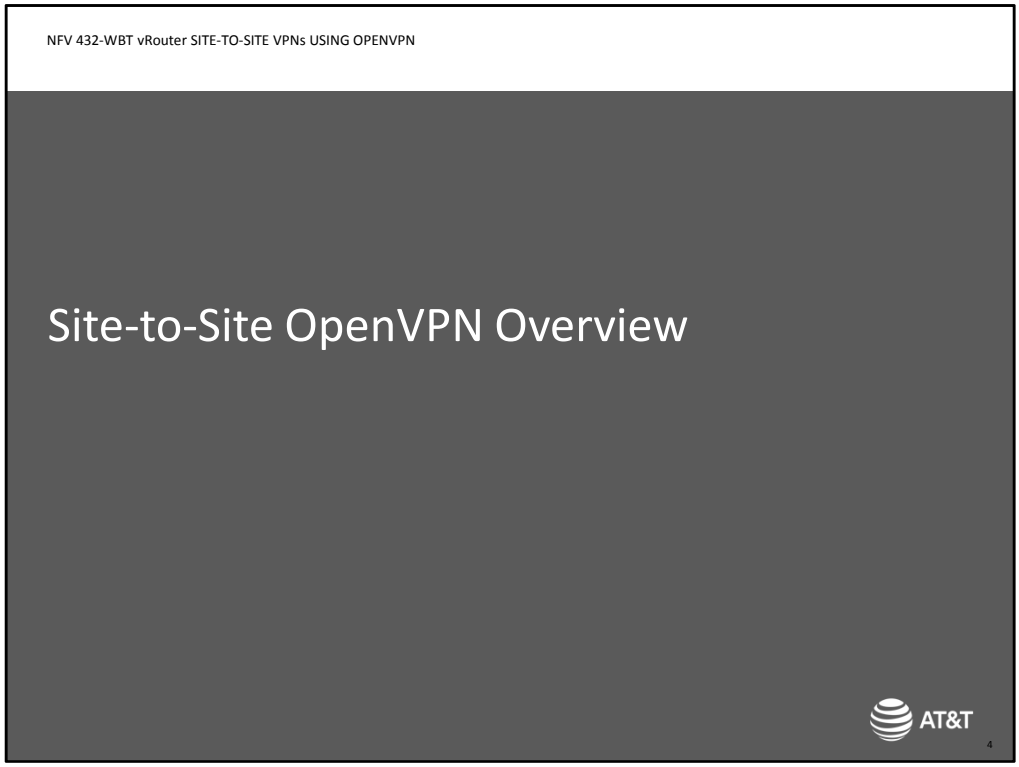
All or some of the products detailed in this presentation may still be under development and certain specifications may be subject to change. Nothing in this presentation shall be deemed to create a warranty of any kind.

Before we begin the course, please take a moment to read our legal disclaimer.

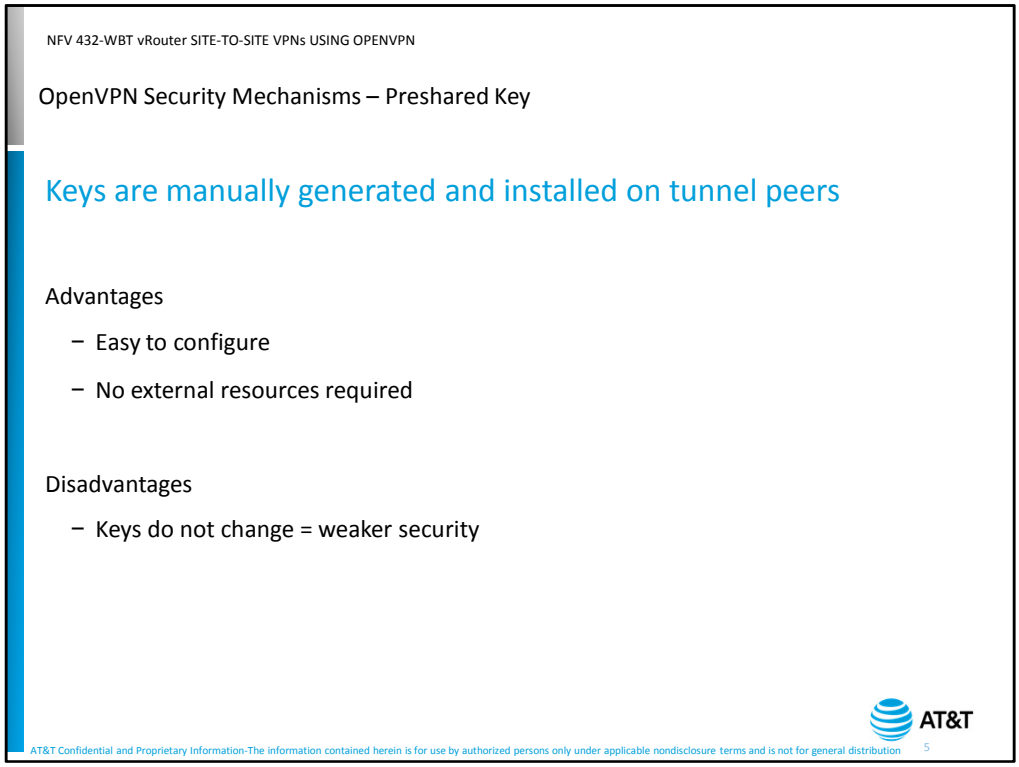


After completing this course, you will be able to:

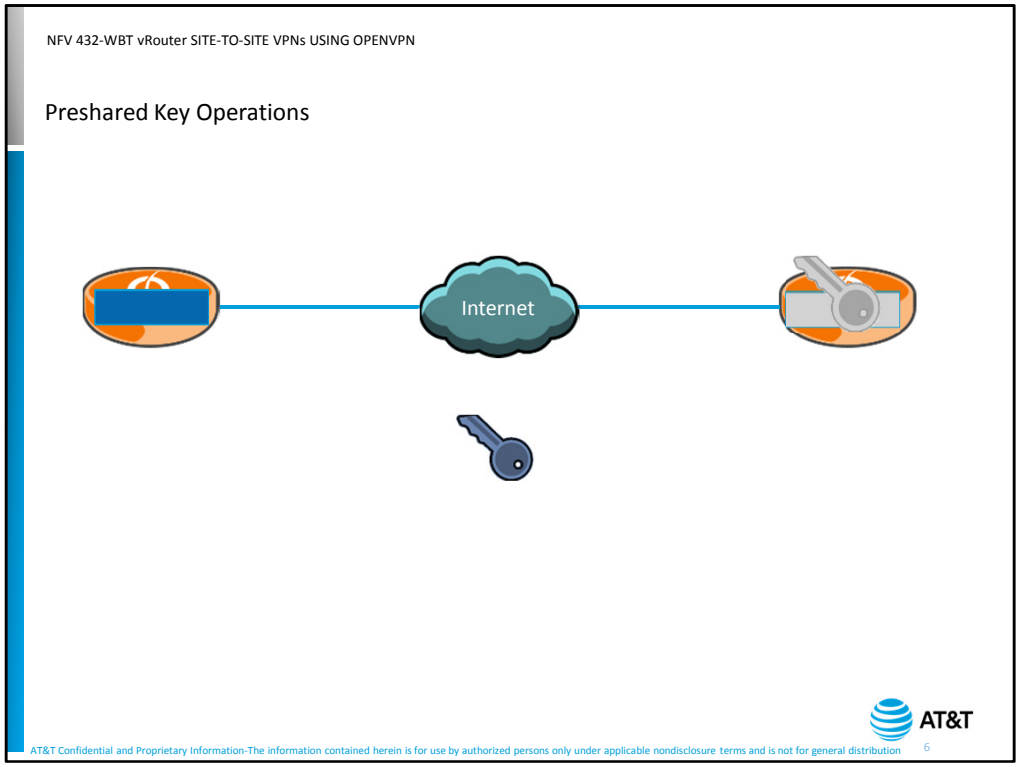
- Describe how OpenVPN secures site-to-site communications
- Configure a site-to-site VPN using OpenVPN on a vRouter device
- Verify tunnel operations
- Troubleshoot common misconfigurations



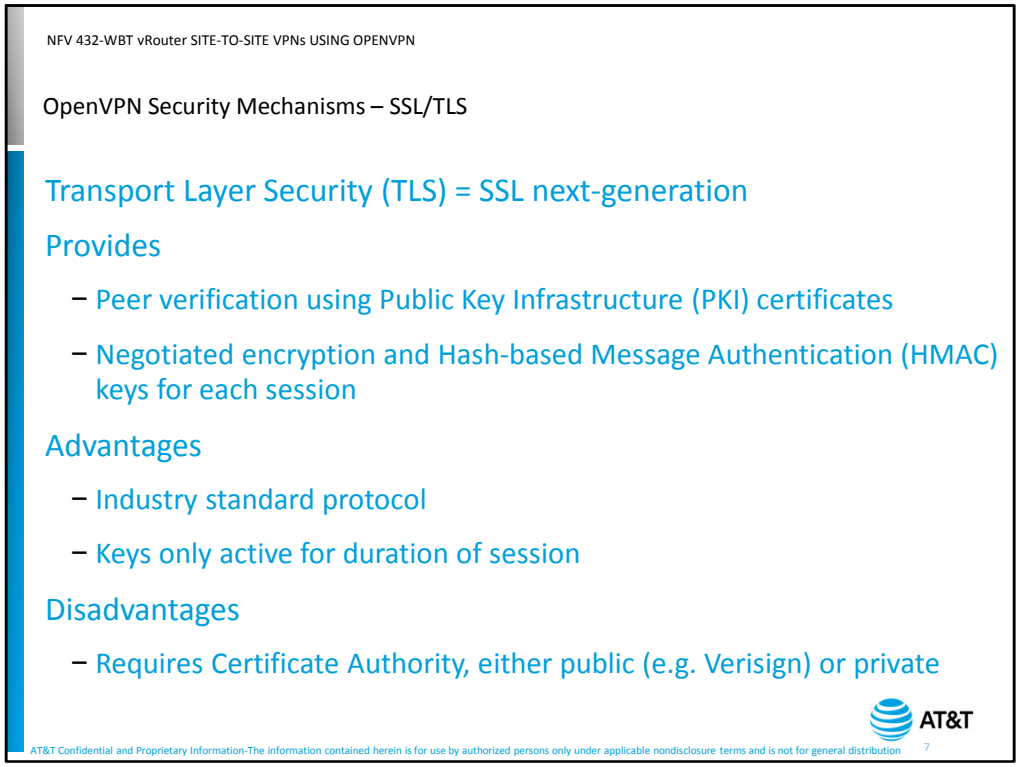
We begin with an overview of site-to-site OpenVPN operations, as well as what common problems can occur.



OpenVPN provides two different mechanisms for securing site-to-site communications. The first is using a preshared key.
A preshared key is a file of random numbers that the device uses to perform encryption and validation of user data. The file is manually generated by the device administrator, and copied to each side of the tunnel connection.
Preshared keys have two advantages: They are relatively simple to configure. All you have to do is copy the key file to the tunnel peer devices. They do not require the use of any external resources, such as a certificate server.
However, there is one disadvantage, preshared keys are static – they do not change over time. If a determined hacker intercepts enough data, they could potentially calculate the encryption key and have full access to the encrypted data. You can overcome this by manually updating your keys at regular intervals, but this rarely happens on production networks.



Let's look at how preshared keys are used.
First, the administrator generates a file that contains a random sequence of numbers. This file is the preshared key.
The administrator then copies this key file to both tunnel peers.
When a tunnel peer receives a packet that must traverse the tunnel, it uses the preshared key to encrypt the packet, then sends the encrypted packet to the tunnel peer.
The tunnel peer uses its own preshared key to decrypt the packet, then forwards it toward the destination.
If the receiver does not have a matching key, it cannot decrypt packets properly. The decrypted packet will be unreadable, and therefore discarded.



The other security mechanism available for site-to-site VPNs using OpenVPN is Transport Layer Security (TLS).

TLS, is simply the next-generation of Secure Socket Layer (SSL). Originally developed to secure Web communications, TLS is now widely used for securing all kinds of data exchanges.

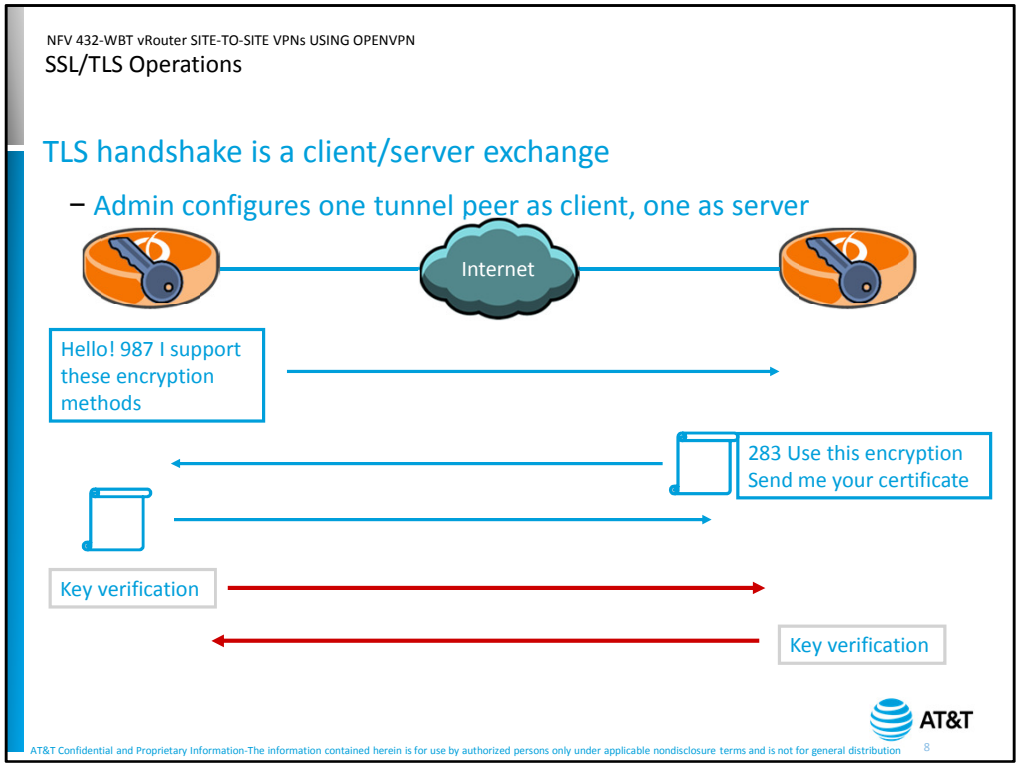
TLS provides peer verification using Public Key Infrastructure (PKI). Devices exchange certificates, which are validated through the use of a Certificate Authority (CA). This validation takes place before any user data is exchanged.

TLS also provides dynamic key negotiation on a per-session basis.

TLS has a couple of advantages: it is an industry-standard protocol, so the peer validation and key exchange mechanisms are well-defined, and the negotiated keys are only active for the duration of the session.

The biggest disadvantage to using TLS is that it requires the use of a Certificate Authority to generate and sign device certificates. This can be a public CA, such as Verisign, or a privately-deployed certificate server, such as a Linux system running OpenSSL.

Please note that the details of PKI are beyond the scope of this course.



Because TLS was originally designed for secure Web-based communications, establishing a connection follows a client/server model.

In our peer-to-peer tunneled environment, you configure one device to be the client and the other to be the server. Once the handshake is completed, the client/server designations do not matter.

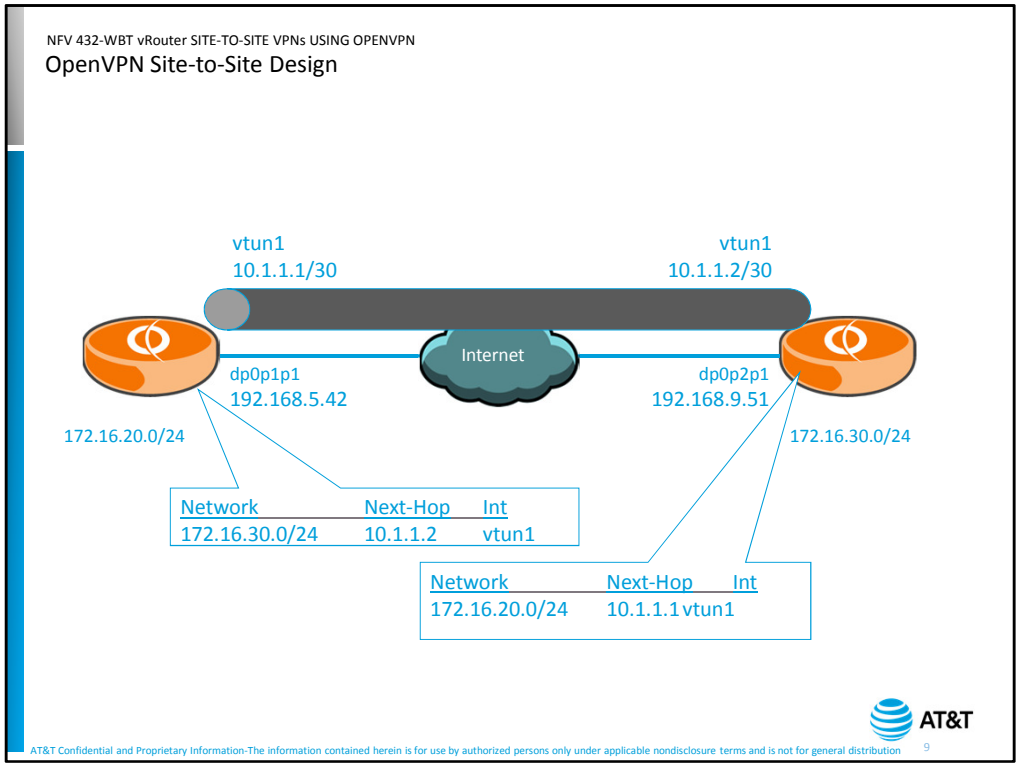
To begin establishing a connection, the client sends a hello message to the server. This message contains a random number and the encryption and HMAC methods the client can support.

The server responds with its certificate, which allows the client to verify the identity of the server. The client uses the Certificate Authority information to verify the server's certificate. If the server certificate cannot be verified, the client drops the connection. The server also sends a random number, the selected encryption and HMAC methods, and a request for the client's certificate.

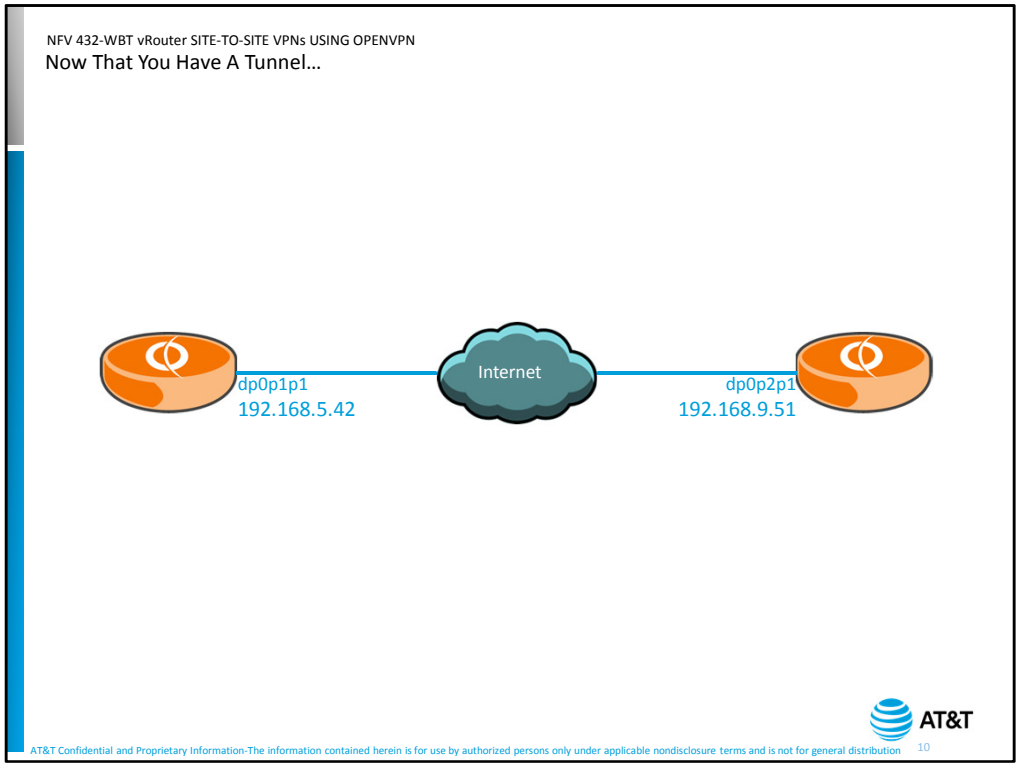
If the server is validated, then the client sends its certificate to the server. The server uses the Certificate Authority information to validate the client's certificate. Again, if the server cannot validate the client certificate, it will drop the connection. Verifying the certificates is where most connection attempts fail, usually because of missing or outdated files relating to certificate verification. We will discuss these required files in detail in the configuration section later in the course.

Assuming both certificates are validated, each device generates a session key using the random numbers provided in the first two messages. The client then sends an encrypted message to the server. If the server cannot decrypt and verify the encrypted packet, then the handshake fails and the server drops the connection.

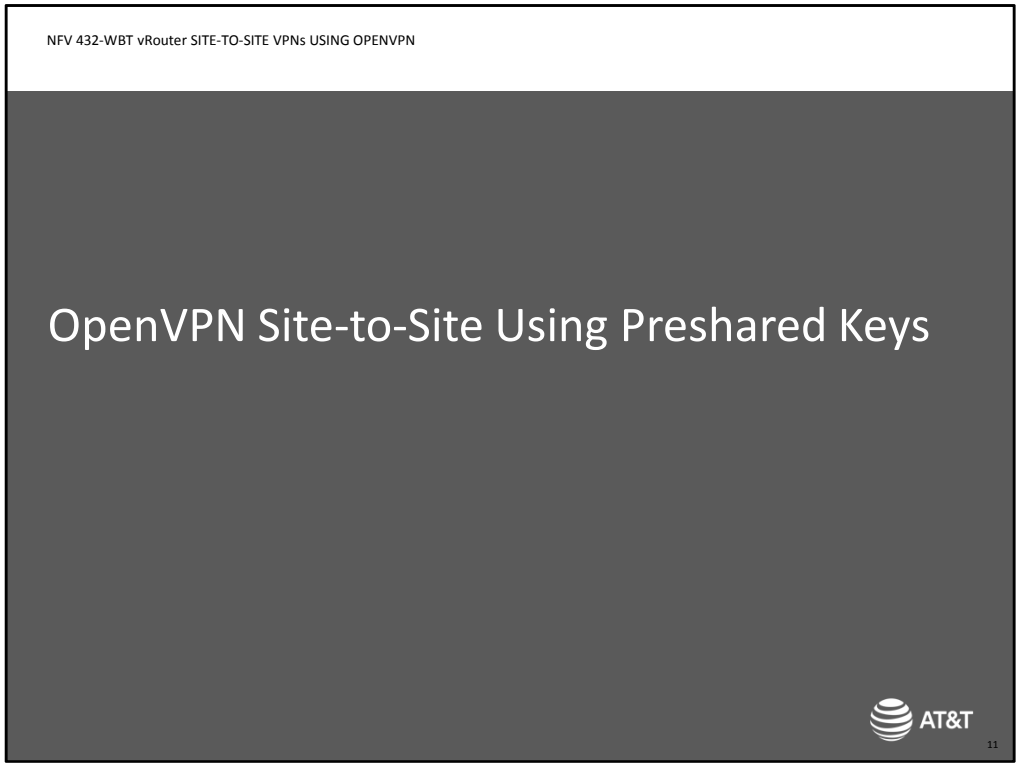
If the server correctly decrypts the packet, it then sends its own key verification. Again, if the client cannot decrypt the packet, then the handshake fails and the server drops the connection. If key verification is successful, then the tunnel peers begin transmitting user data, encrypting it with the negotiated session keys.



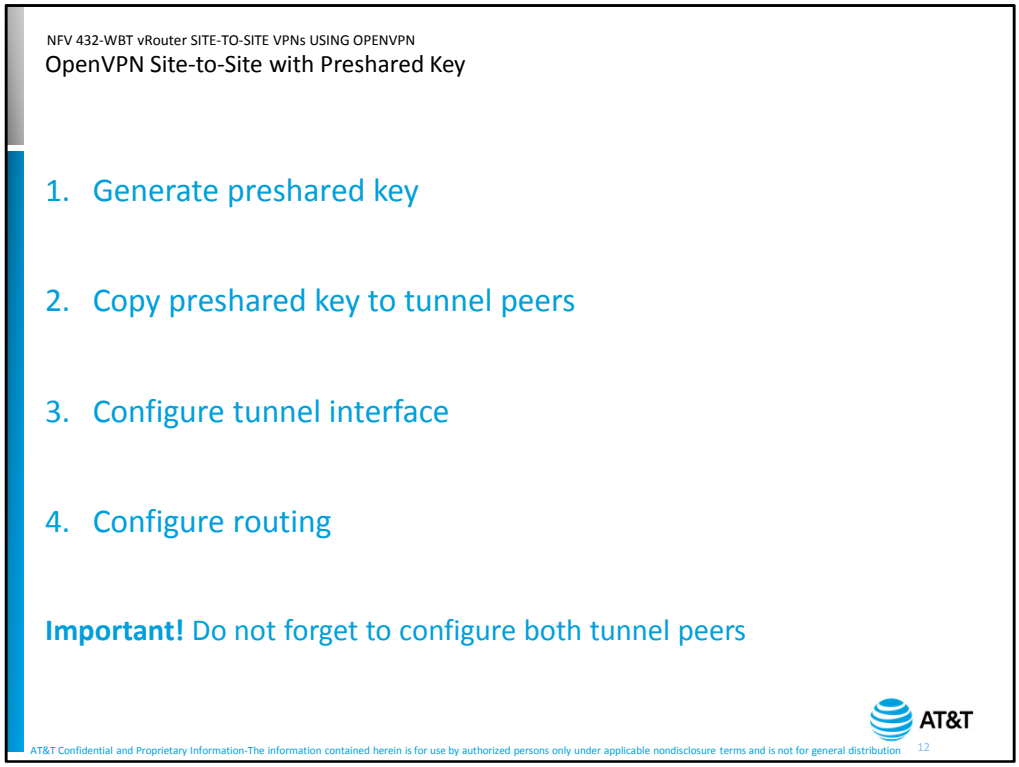
Regardless of the security method you use, an OpenVPN site-to-site network has some basic design requirements.
The tunnel peers must be able to reach other via the transport network, which is usually the Internet.
The tunnel between the two sites appears to be a point-to-point network directly connecting the two sites.
You create this point-to-point network by creating a virtual interface, called a tunnel interface, at each site. Note that these interfaces are on the same subnet.
To send traffic across the tunnel, each side must have routing information about the networks on the other side of the tunnel. This routing information can be manually configured as a static route, or learned via a dynamic routing protocol running over the tunnel.



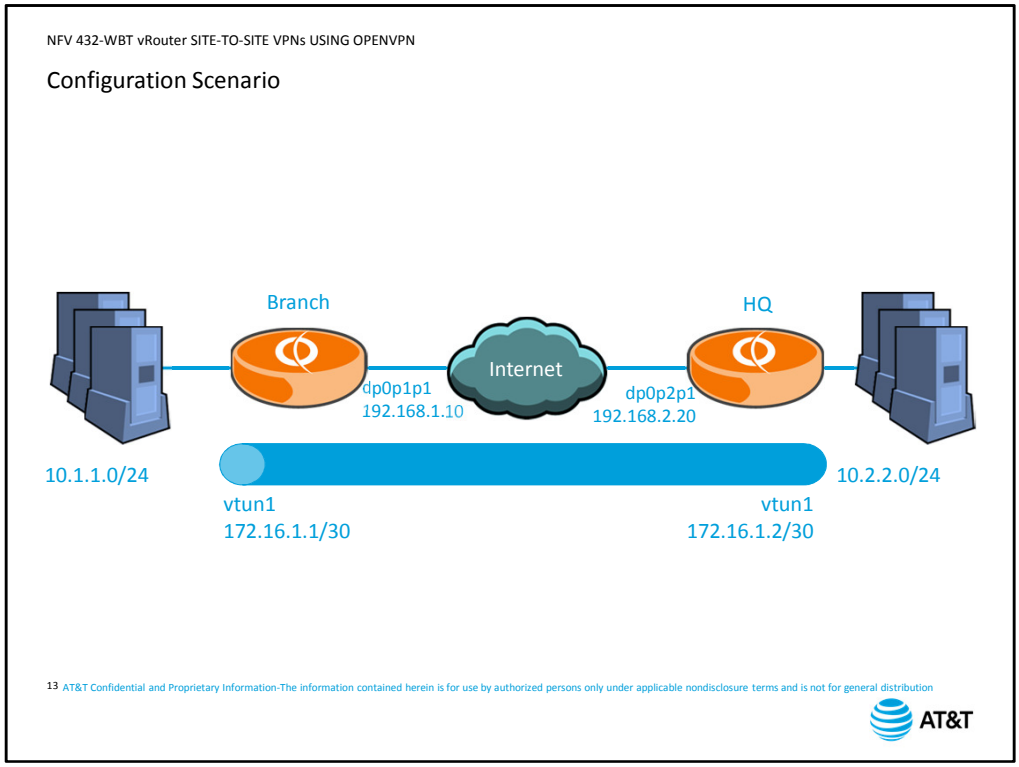
As soon as you have configured the tunnel, the vRouter is ready to process packets.
When the vRouter receives a packet that is routed to the tunnel interface, it uses the encryption key to encrypt the packet, then adds a new IP header, setting the source address to the address of the local physical interface, and the destination address to the address of the remote peer physical interface.
The device then sends the packet across the public network.
The receiver uses its key to decrypt the packet, then forwards it to the original destination.



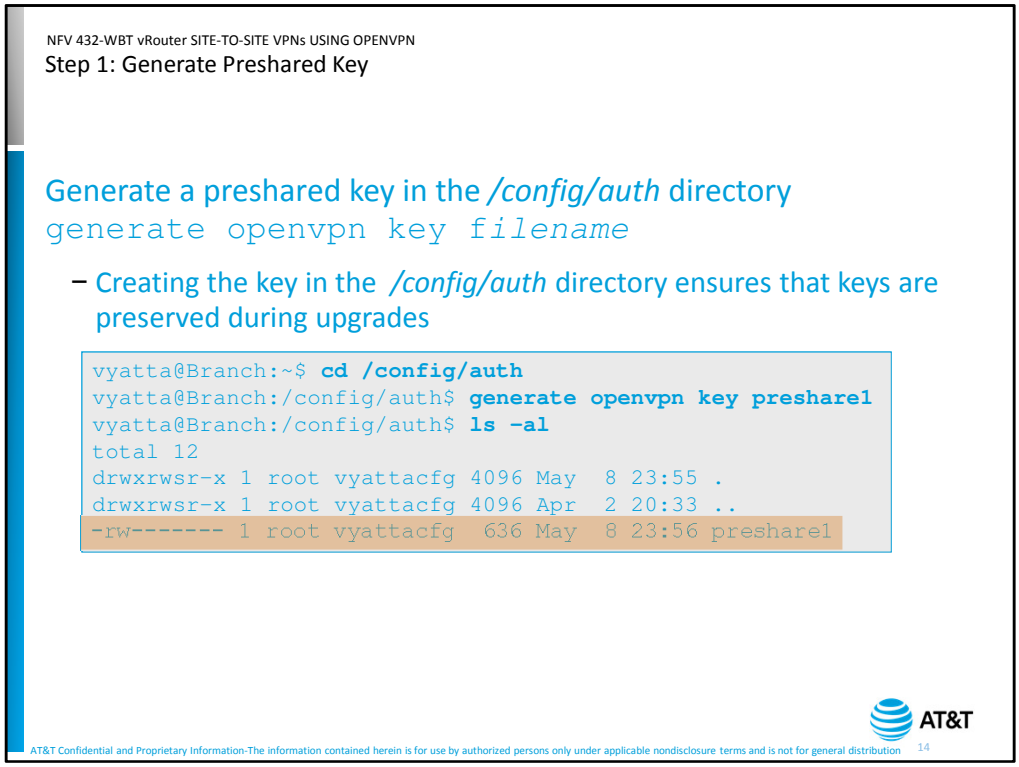
Now we will look at the commands you need to set up a site-to-site VPN using preshared keys.



Configuring a vRouter OpenVPN site-to-site connection consists of four major steps:
The first step is generate the preshared key file.
The second step is to copy the file to the tunnel peers.
The third step is to configure the tunnel interface.
The last step is to configure routing to direct traffic into the tunnel.
Don’t forget, a VPN involves two devices, so you need to configure both tunnel peers.



We will use this configuration scenario as we go through the steps of our configuration.
The vRouters are connected to the Internet using data plane interface 1.
The subnets we want to tunnel are 10.1.1.0 at the Branch site and 10.2.2.0 at the Headquarters site.
The point-to-point tunnel subnet is 172.16.1.0, with host 1 at the Branch site and host 2 at the Headquarters site.

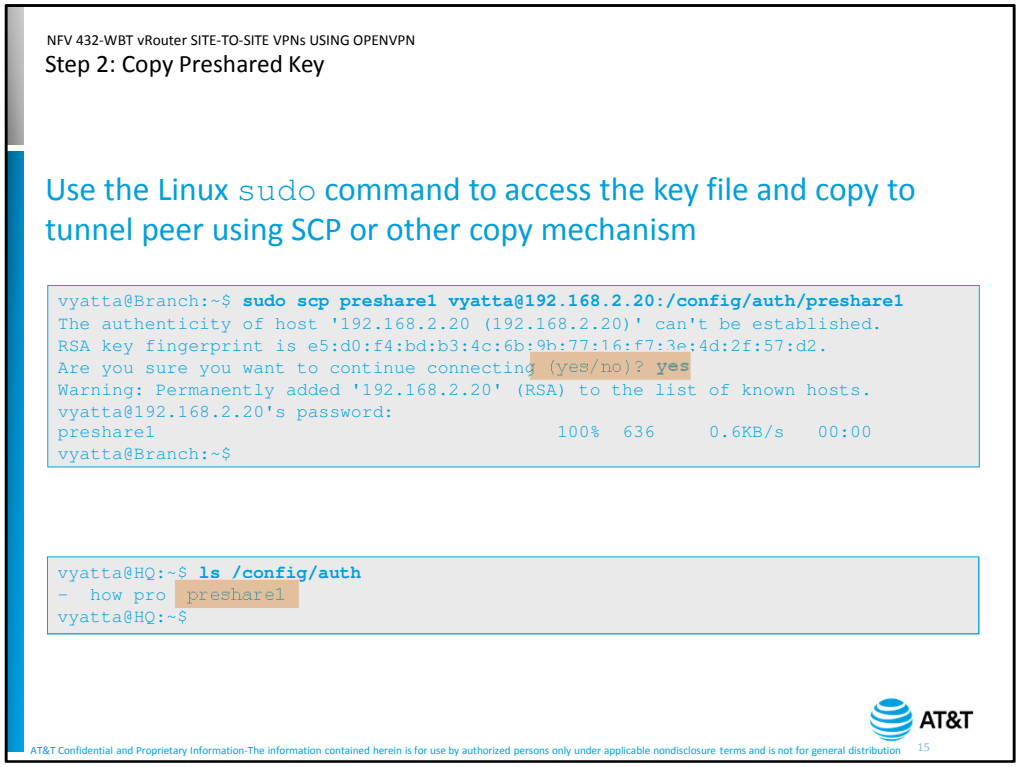


To generate the preshared key file, use the Operational mode command `generate openvpn key`, followed by the filename you want used for the key.

Make sure you change to the */config/auth* directory before entering the command to generate the key. This ensures that the key file is preserved during system upgrades. Other directories on the 5600 vRouter may be overwritten during upgrades.

The screen capture displays the command to change to the */config/auth* directory, followed by the command to generate the key. Finally the screen capture displays the listing of files in the directory.

You can see the file you just created. Note that the file is only readable and writable by the root user.



The next step is to copy the key file to the tunnel peers. If you created the file on one of the tunnel peers, you only have to copy it to the other peer.

As we saw on the previous slide, the file is only accessible by the root user. You can use `sudo` to issue root-level commands to access the file.

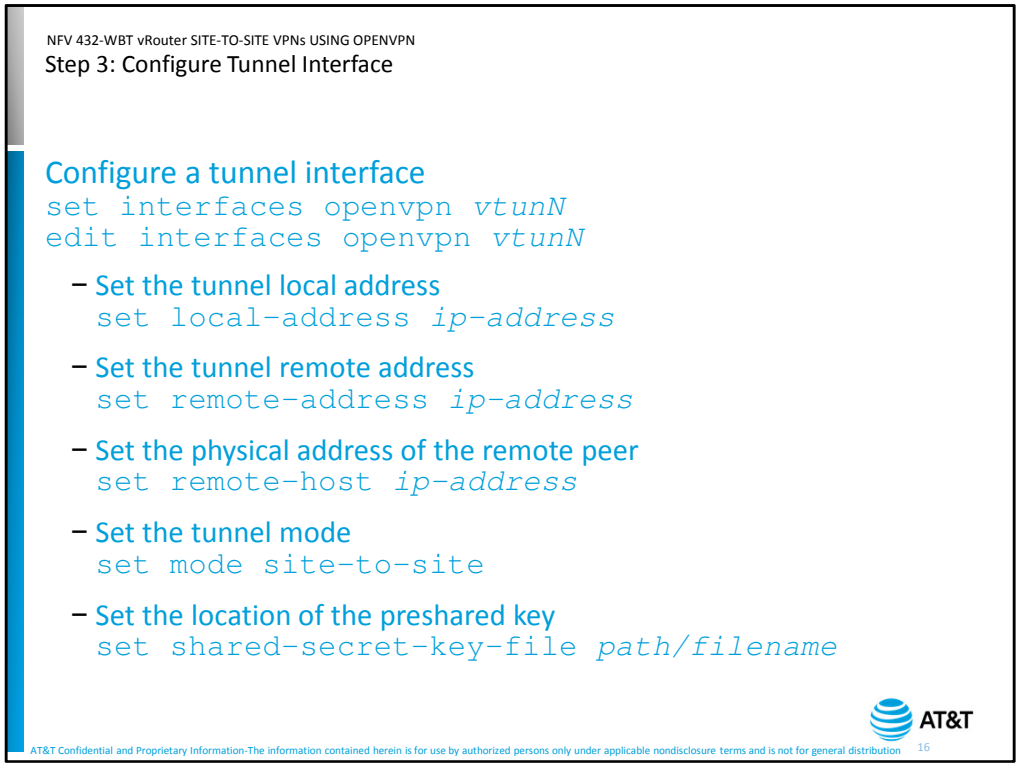
Copy the file using SCP or some other copy mechanism.

In the screen capture, we use sudo to access the secure copy command, specifying the remote address of the peer and the location where we want the file to be copied. Because we have not established a secure connection to this host before, we are prompted to verify the RSA fingerprint.

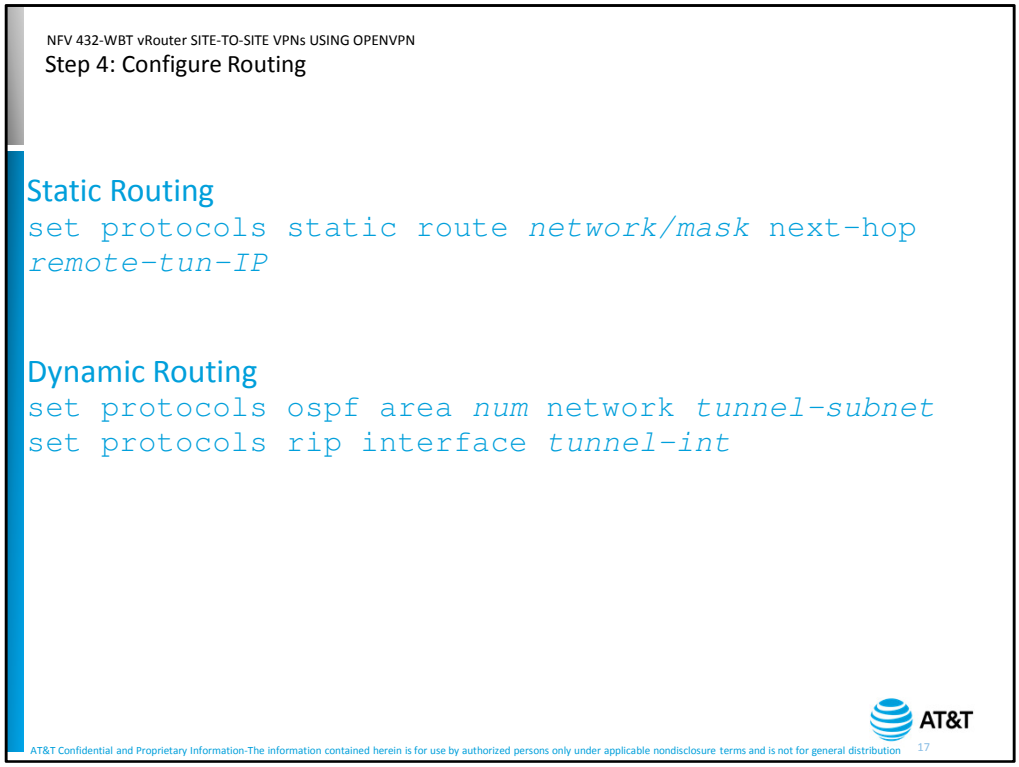
We type *yes*. We are then prompted for the password of the account we specified in the command.

After we type the password, the file is copied to the remote host.

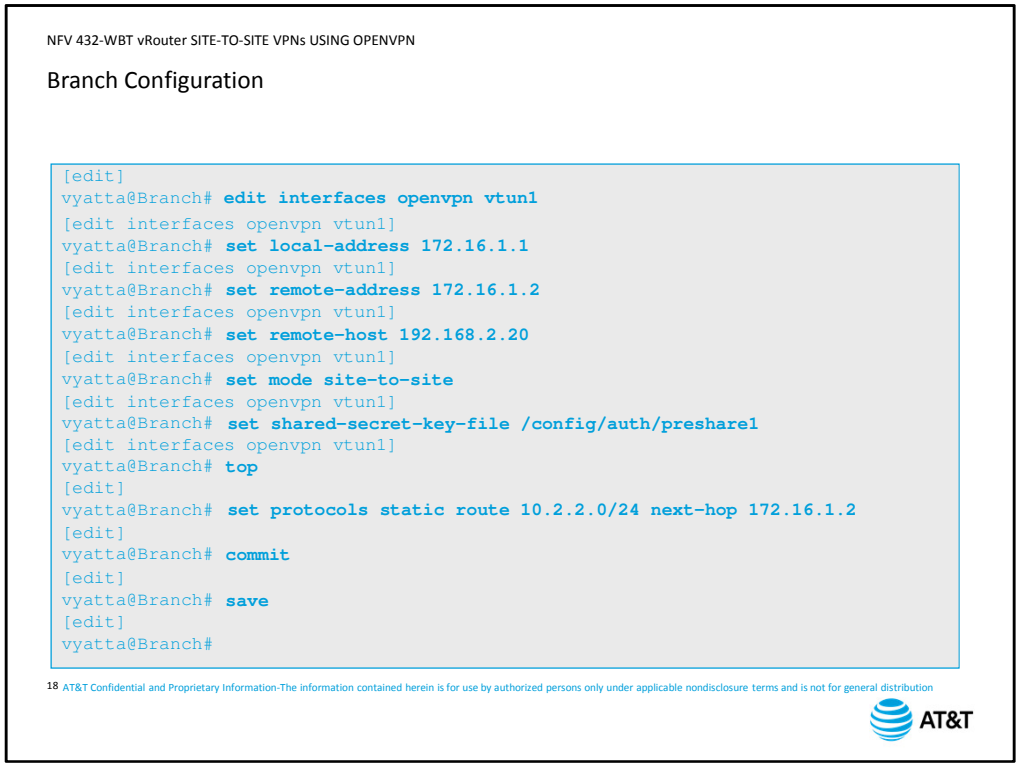
When we list the files at Headquarters, we see that the *preshare1* file is there.



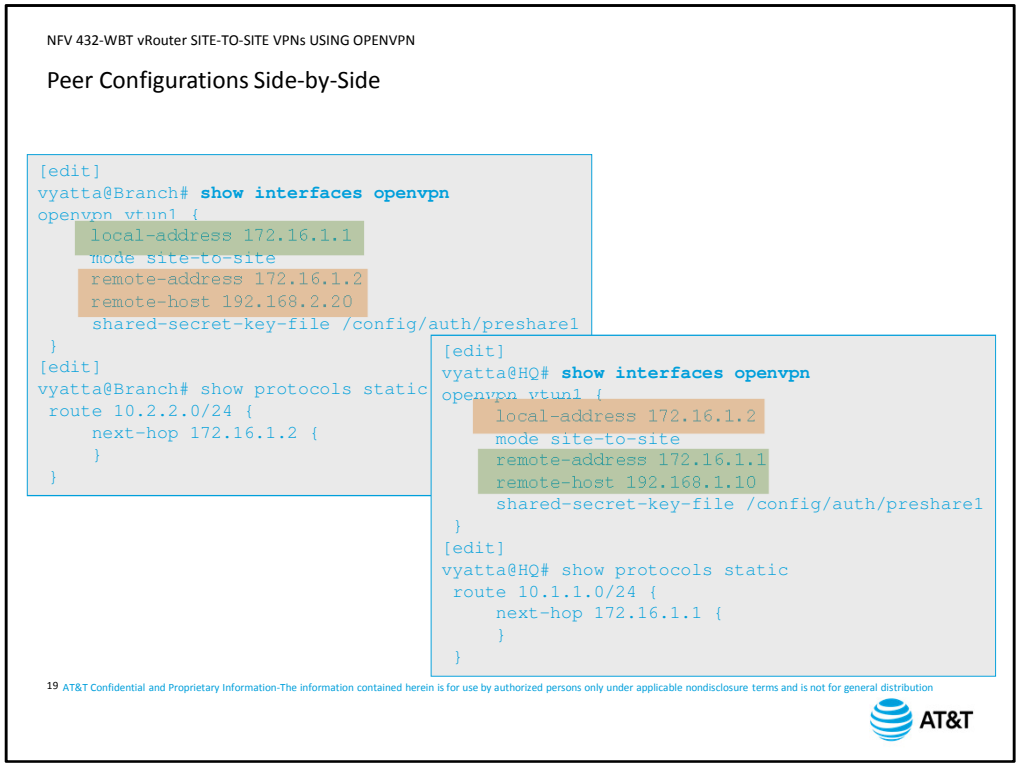
The next step is to configure the tunnel interface on each peer.
First, create the tunnel interface.
Because the rest of the commands must all be associated with the same tunnel interface, we recommend using the `edit` command to move within the configuration hierarchy. The rest of the commands on the screen assume you have used the `edit` command.
Next, set the local IP address for the tunnel interface.
Next, set the IP address of the remote end of the tunnel. This address must be on the same subnet as the local IP address
Next, set the physical address of the remote peer device. This is usually the interface connected to the Internet.
Next, set the tunnel mode to `site-to-site`.
Finally, specify the location of the preshared key file.



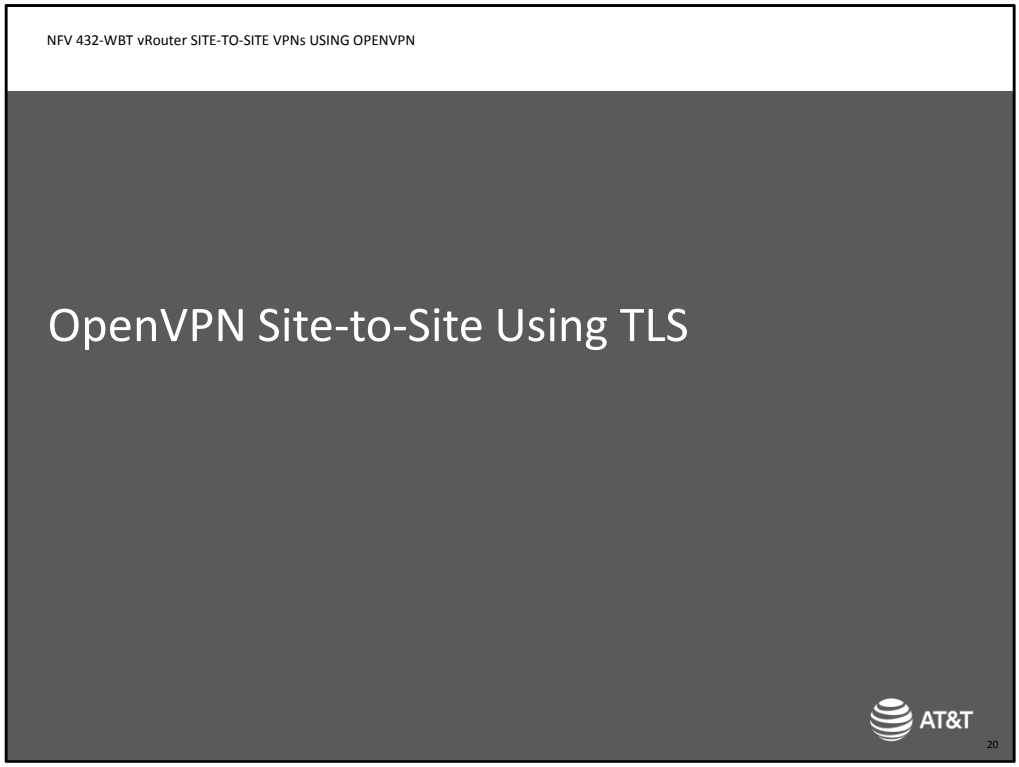
Finally, you configure the routing needed to direct traffic into the tunnel.
If you are using static routes, enter the address of the remote network, setting the next-hop to be the address of the remote tunnel interface.
If you are using dynamic routing, enable routing on the tunnel subnet or tunnel interface, depending on the protocol you are using.



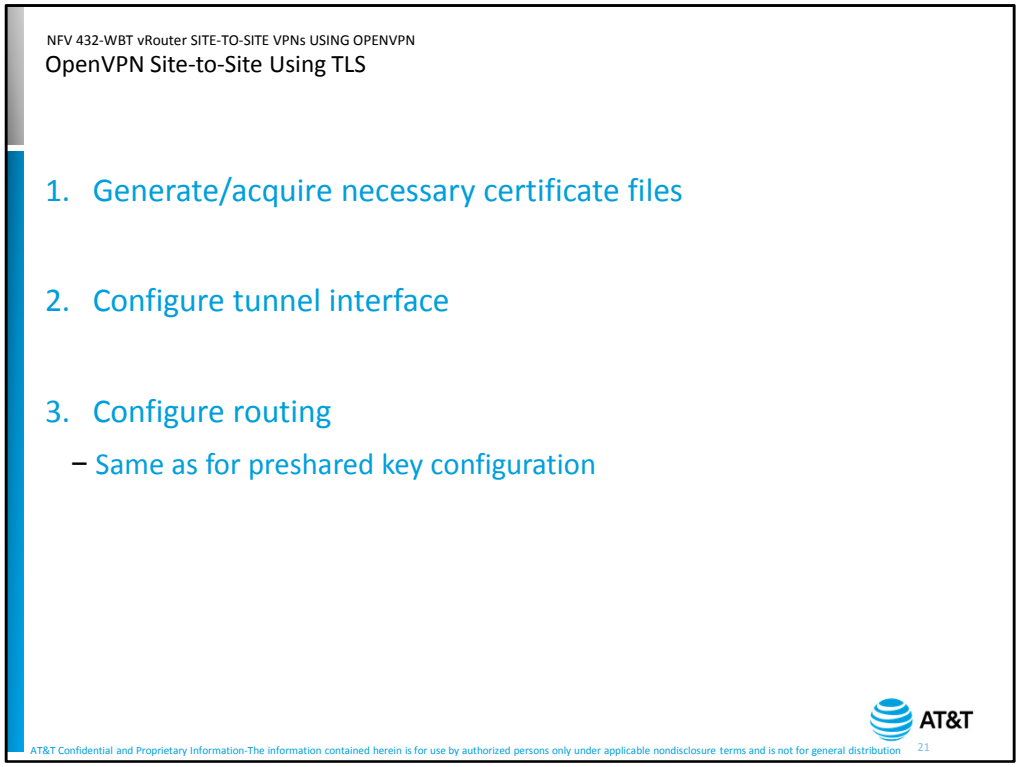
Let's configure the Branch office device first.
In Configuration mode, we use the `edit` command to create the virtual tunnel interface and move us within the configuration hierarchy. Note the prompt changes to show us where we are in the hierarchy.
Next, we set the local address of the tunnel interface.
Then, we set the address of the remote tunnel interface.
Next, we set the physical address of the remote peer.
Then, we set the mode to site-to-site.
Next, we specify the location of the preshared key file.
We are done with the configuration of the tunnel interface, so we use the `top` command to return us to the top of the configuration hierarchy.
We are using static routes in our configuration, so we configure the static route to the remote subnet.
Finally, we commit and save our changes.



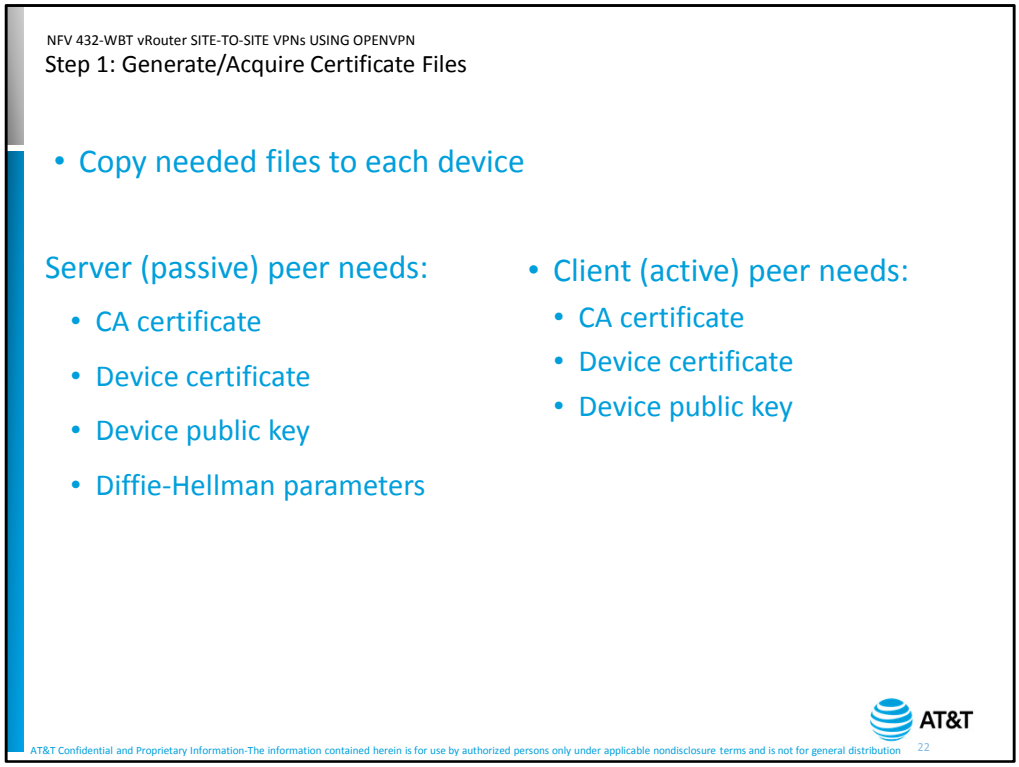
We complete a similar configuration at the Headquarters site, then look at the configurations side-by-side.
We can see the local and remote IP addresses are reversed on each side of the connection.



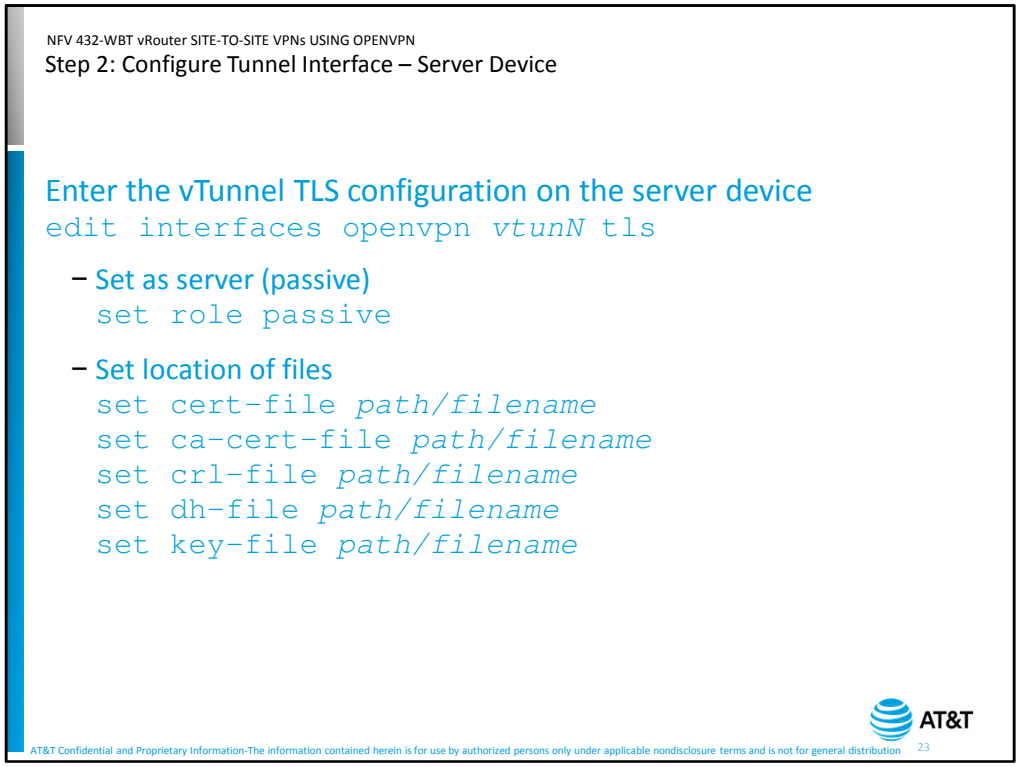
Now we will look at the commands you need to set up site-to-site using TLS.



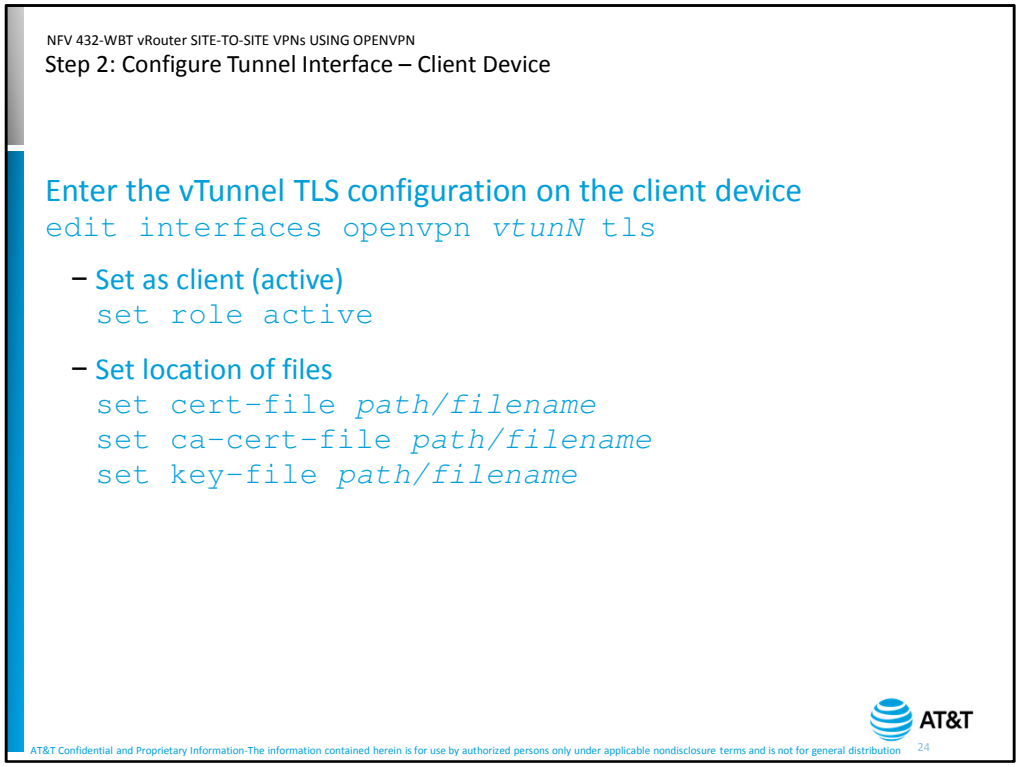
Instead of generating and copying a preshared key file, you acquire the certificate files needed on each device, then copy those files to the devices.
Next, you configure the tunnel interfaces, setting the TLS commands instead of the preshared key file location.
Finally, you configure routing as you did for the preshared key configuration.



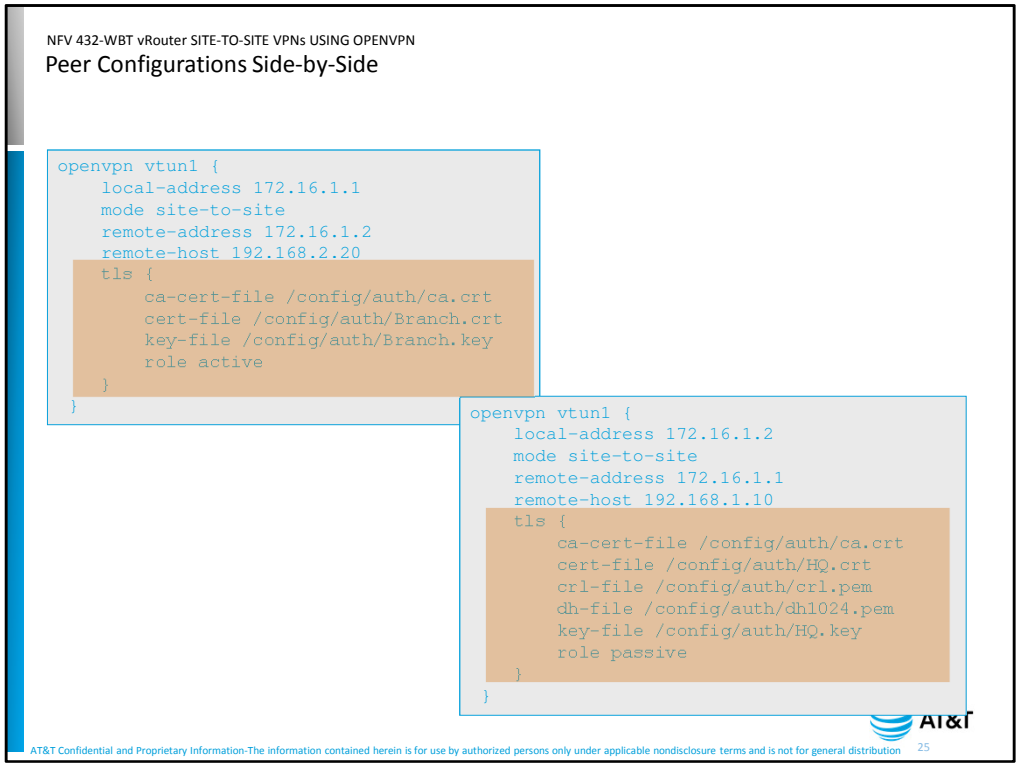
Your first step is to copy the necessary files to each tunnel peer.
The server device needs to be configured with the certificates and parameters in the Server column.
The client device needs to be configured with the certificates and key in the Client column.
How you acquire these files and the file name formats will vary depending on your Certificate Authority.



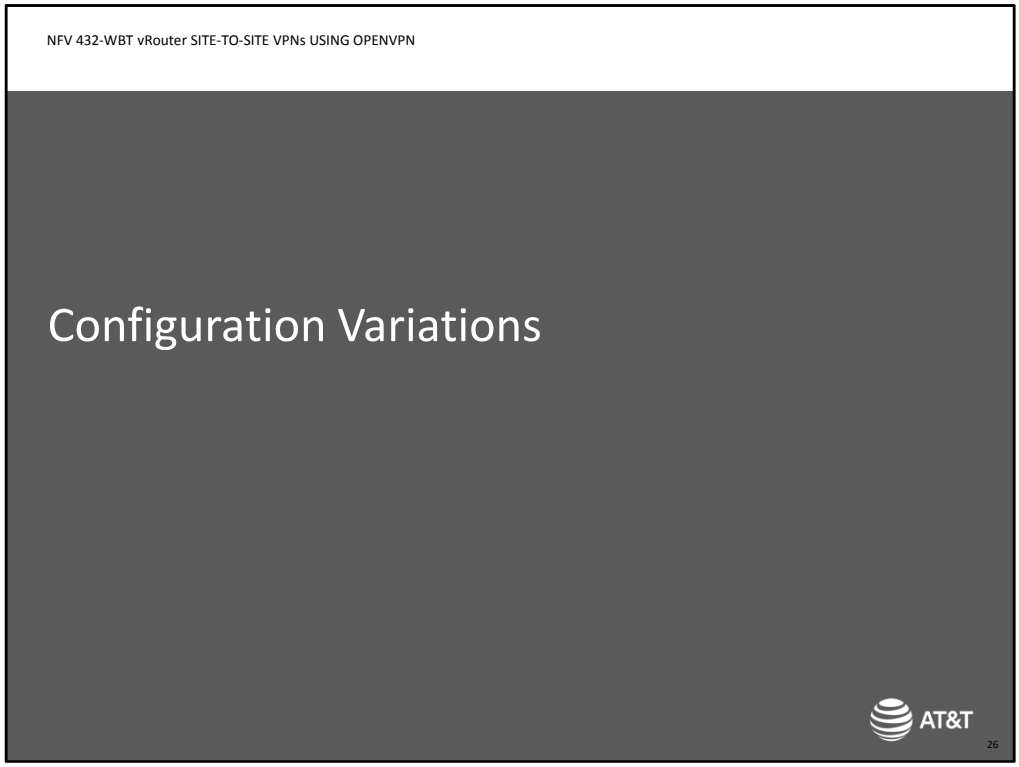
Once you have added the files to each device, configure the tunnel interface on each side.
On the server device, configure all the addressing as for preshared key.
Then, instead of entering the preshared key location, use the `edit` command to move into the TLS commands for the interface.
The server device is passive; that is, it waits for the client to initiate the connection.
Remember, the server/client designations are only used for tunnel setup; once the tunnel is established, the devices operate as peers.
Next, specify the location of all the files needed to perform certificate-based authentication. Note that the crl-file is optional.
To make sure your files are preserved across device upgrades, they should be stored in the */config/auth* directory.



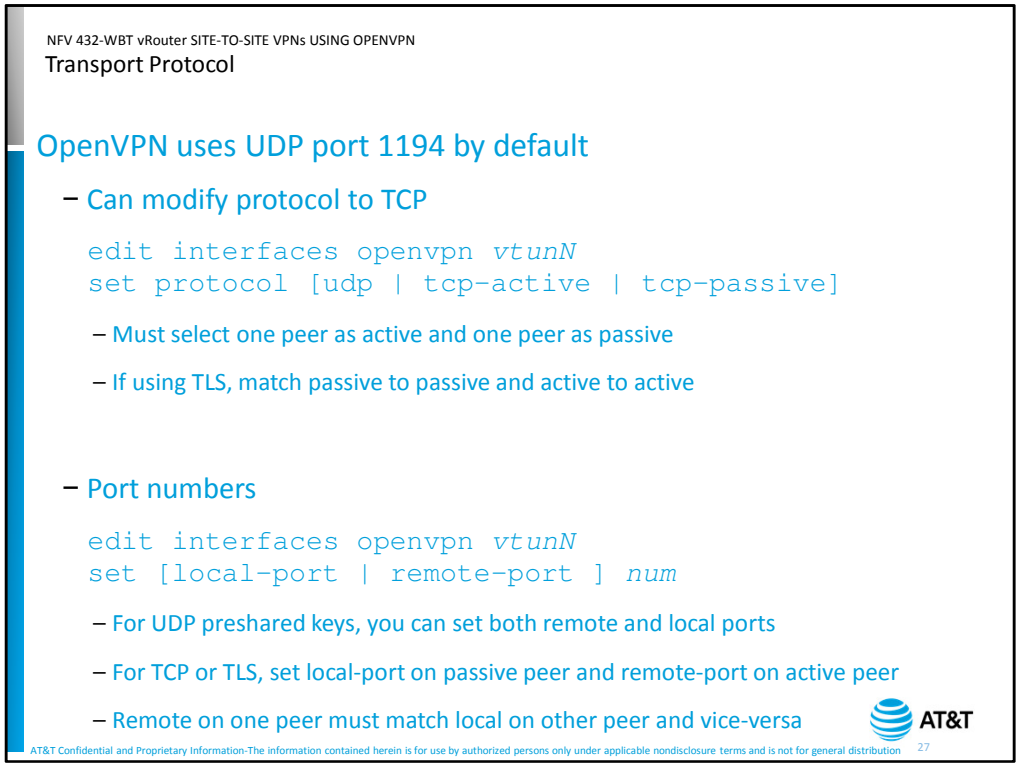
On the client side, configure the addressing as before.
Then, instead of entering the preshared key location, use the edit command to move into the TLS hierachy and enter the commands for the interface.
The client device is *active*; it initiates the connection.
Next, specify the location of all the files needed to perform certificate-based authentication.



In our example, we are using the same scenario as before. We have made the Branch office the client and the Headquarters office the server.
We have specified all the file locations at Headquarters.
and at the Branch office. Note that the rest of the configuration for the interfaces is the same as for preshared key. The screen shots do not show the routing configuration either, but you still need to configure routing in order for traffic to traverse the tunnel.

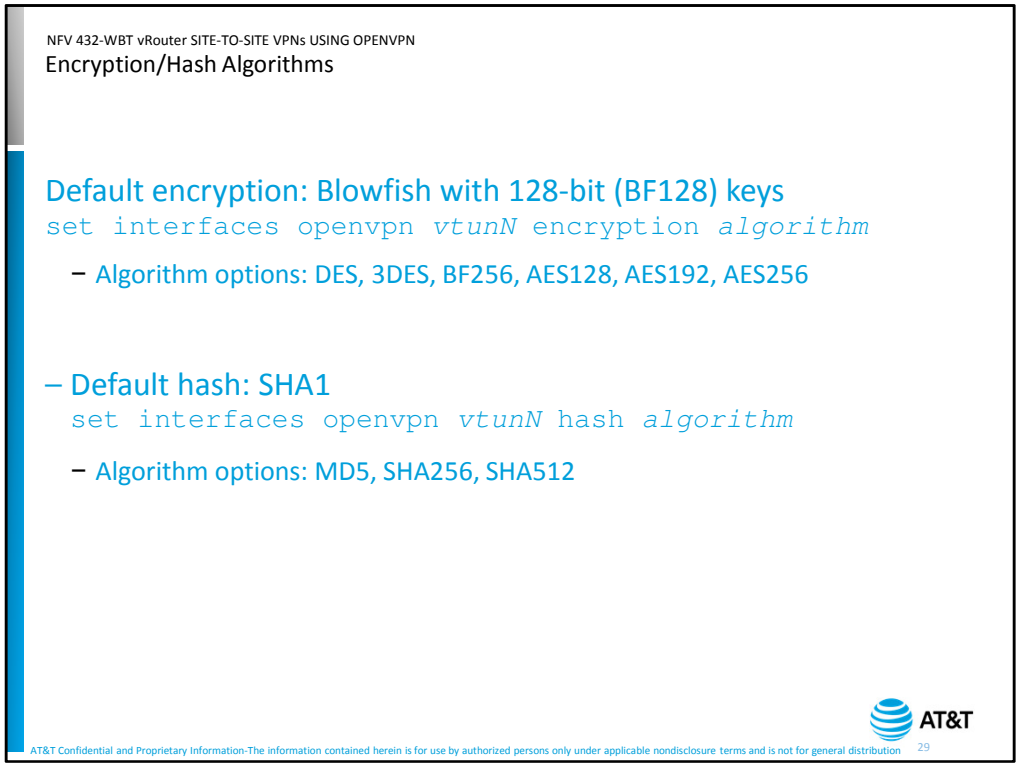


Next, we will look at some configuration variations for site-to-site OpenVPN tunnels.

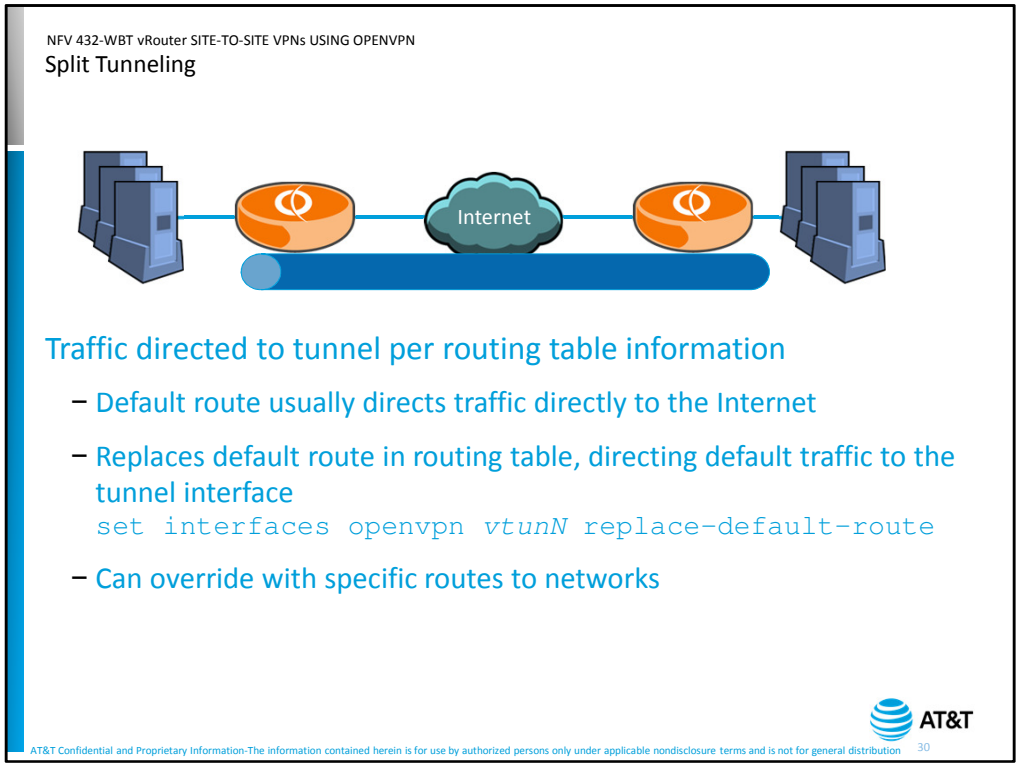


By default, OpenVPN tunnels use UDP on port 1194 as the transport protocol, relying on the original packet protocol and application to take care of any connection-oriented requirements.
You can modify the protocol to use TCP if your network requires it.
However, if you use TCP, one device must assume the active, or client role, while the other device must assume the passive, or server role.
If you are using TLS as your encryption method, you must configure `tcp-passive` on the same device that is passive in the TLS setup exchange. Likewise, you must configure `tcp-active` on the same device that is active in the TLS setup exchange.
You can also modify the port number used for OpenVPN connections.
If you are using UDP as your transport protocol, and are using preshared keys, you can modify both local and remote ports, since either device can initiate the connection.
If you are using TCP as your transport protocol, or if you are using TLS for encryption, you will configure `local-port` on the passive peer, and `remote-port` on the active peer.
Remember that you must configure both sides of the connection, and that the remote setting on one side of the tunnel must match the local setting on the other side.

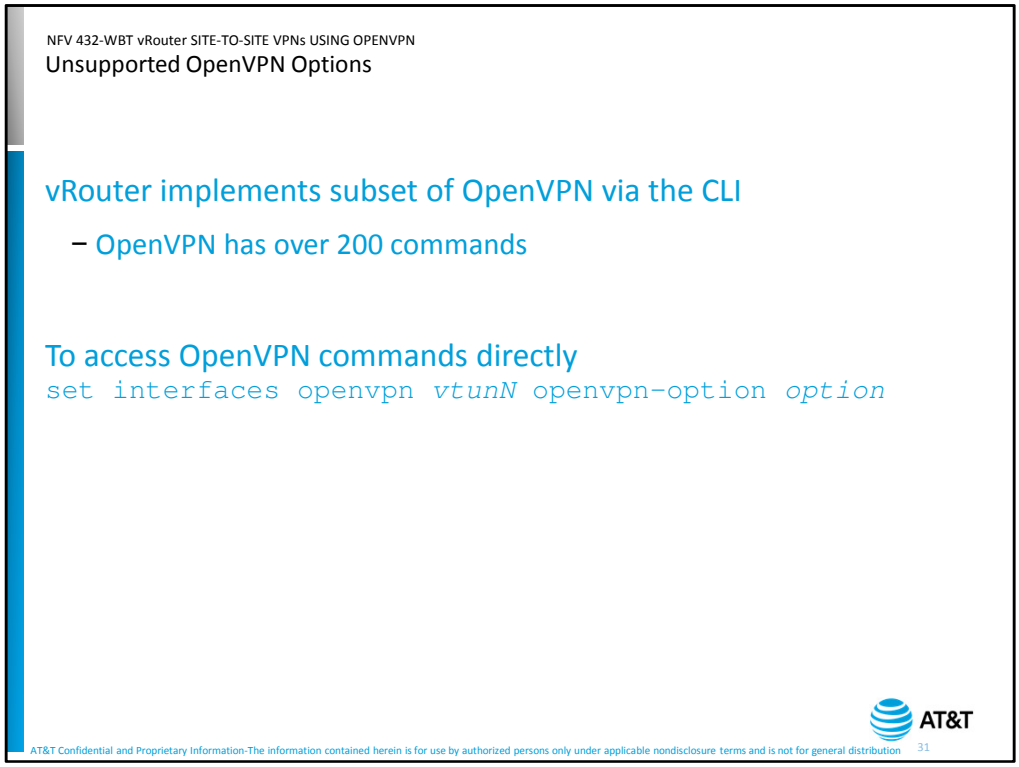
We are still using the same network scenario, with the Branch office device as the active peer and the Headquarters device as the passive peer. In this case we are using preshared keys, but we could also use TLS.
The branch office is set to `tcp-active`.
and Headquarters is set to `tcp-passive`.
The remote port setting at the Branch office matches the local port setting at Headquarters.



You can modify the encryption and hash algorithms used by the VPN.
By default, OpenVPN tunnels use Blowfish with 128 bit keys. Options include DES, Triple DES, Blowfish with 128 bit keys, Blowfish with 256 bit keys, AES with 128 bit keys, AES with 192 bit keys, and AES with 256 bit keys.
You set the encryption algorithm on the tunnel interface. Remember to set both sides of the tunnel to use the same algorithm.
For the hash algorithm, OpenVPN uses SHA1 by default. You can set the hash algorithm to MD5, SHA1, SHA256 or SHA512.
Again, if you modify this setting, you must modify it on both sides of the tunnel or the tunnel will not come up.

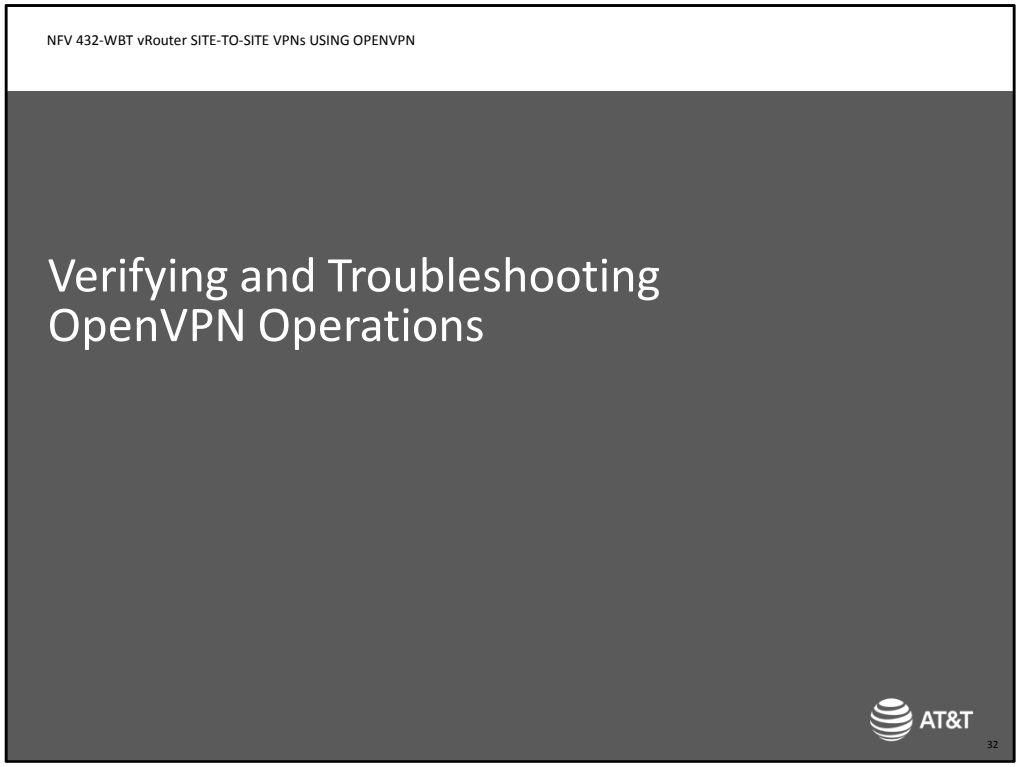


Split tunneling simply means that only traffic explicitly directed into the tunnel by the routing table will use the tunnel.
All other traffic also follows the routing table, which usually has a default route directing traffic directly out the interface connected to the Internet.
You can use this shortcut command to replace the default route with a default route that directs all traffic through the tunnel. You may need this due to security requirements in your environment that want all traffic directed to a central location before being sent to the Internet.
You can still perform split tunneling with this command in place; you simply have to configure routes to specific destinations in the routing table.

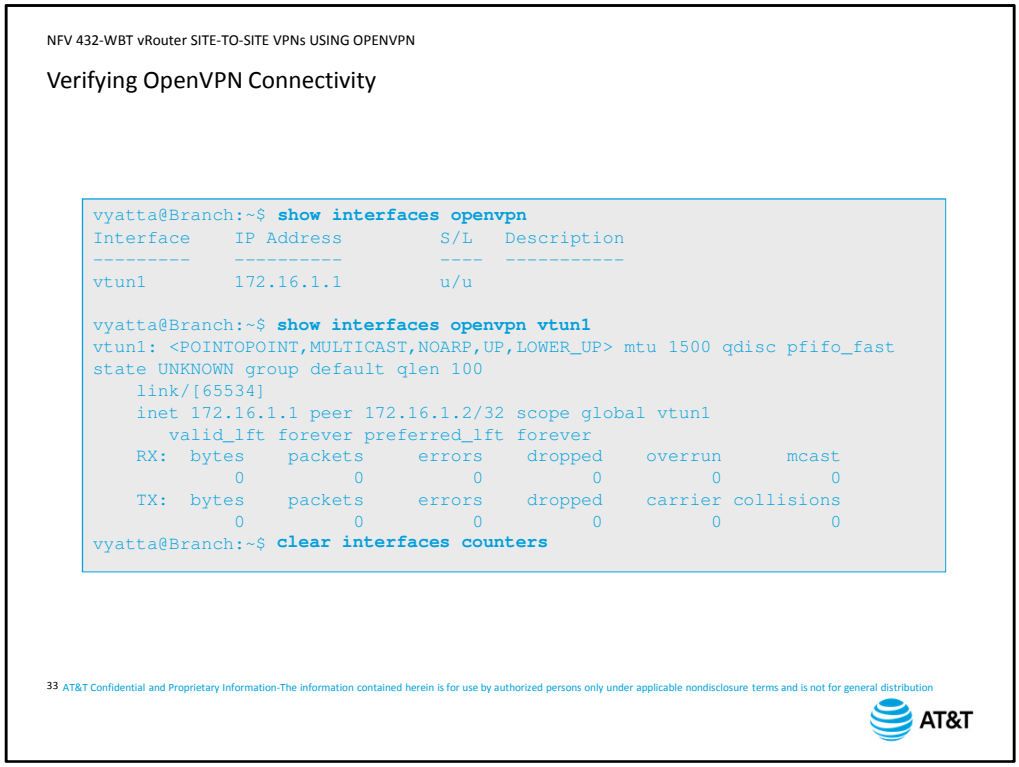


The vRouter CLI only implements a subset of the complete OpenVPN application. With over 200 available commands, it is not feasible for AT&T to incorporate the entire feature set. However, you may have a specific OpenVPN option you want to use that is not integrated into the CLI.

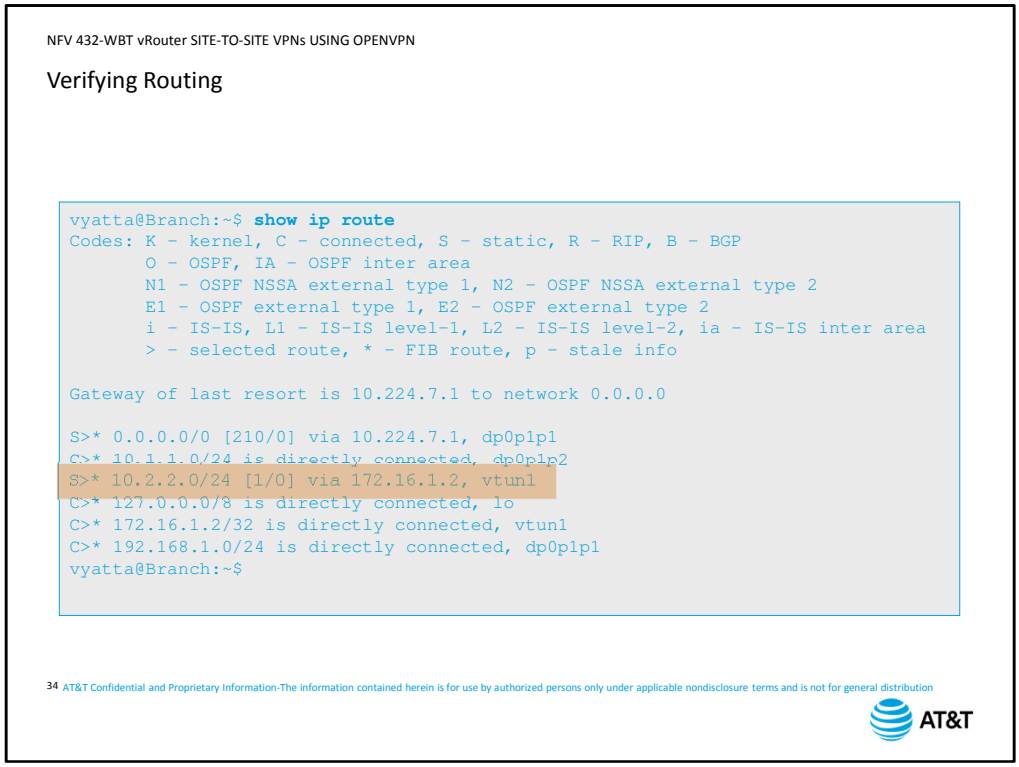
AT&T has added a command that allows you to directly implement OpenVPN options. The option string you enter is passed directly to the OpenVPN application and is not validated by the CLI. Therefore, you need to make sure that the specified option and associated values are valid, and there are no conflicts with any other OpenVPN options configured either through vRouter CLI commands or OpenVPN option commands.



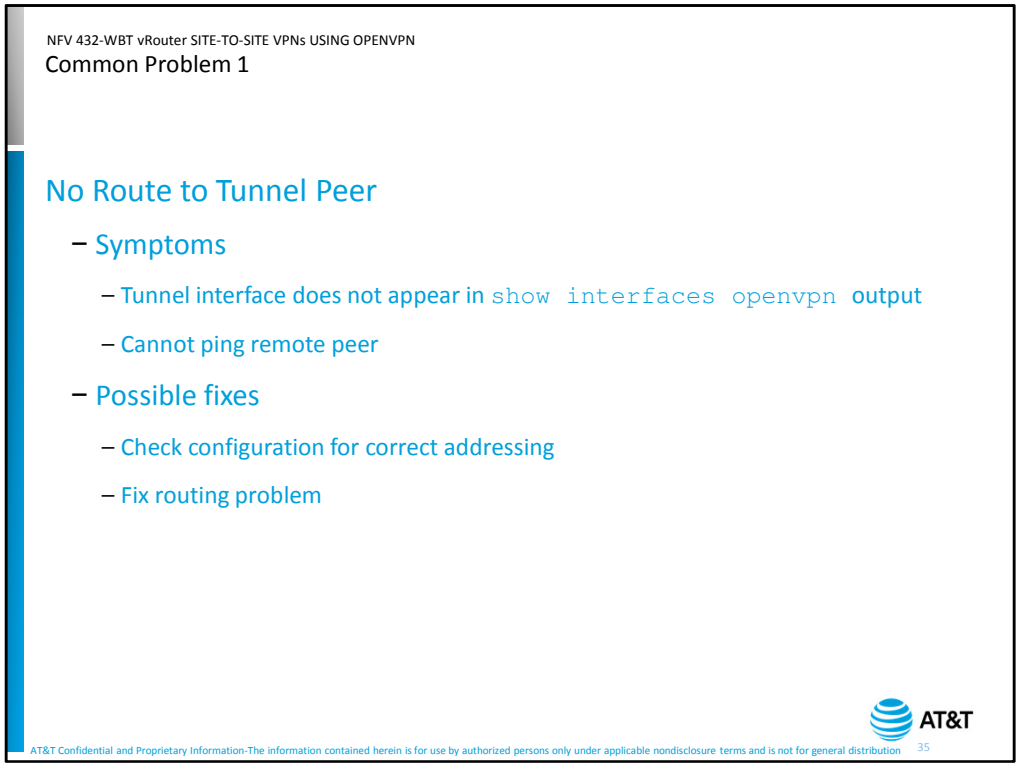
Now let’s look at the commands you can use to verify your VPN is up and operational, as well as what to look for when it is not.



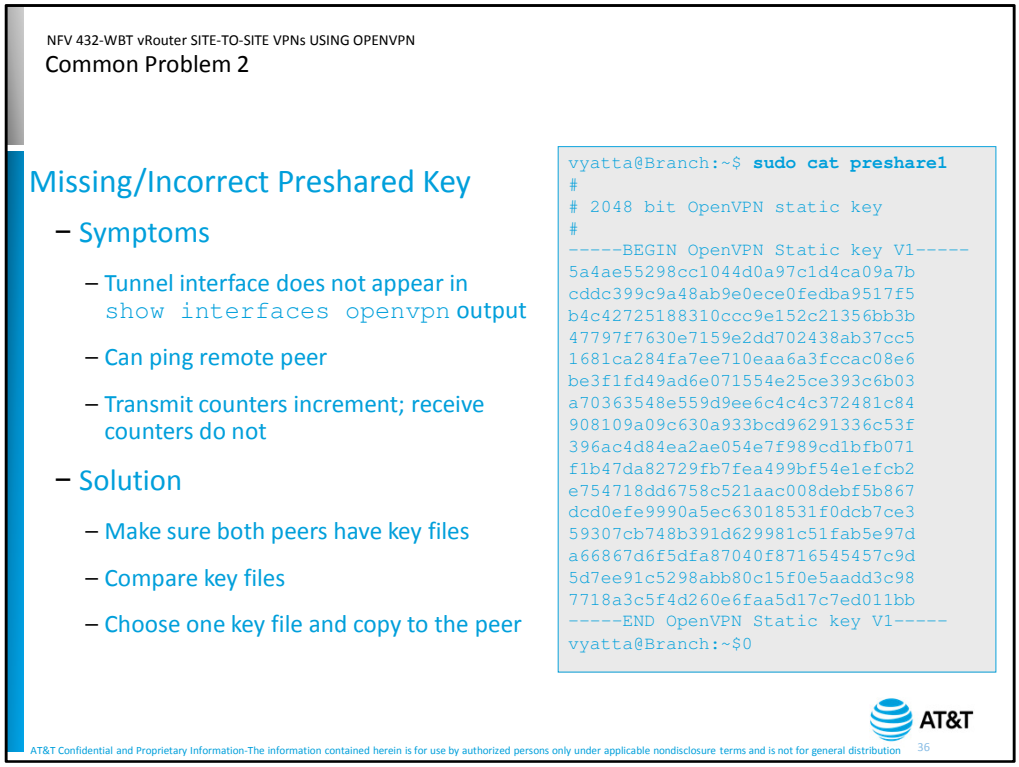
To verify that the tunnel is up, use the command `show interfaces openvpn`. The output displays all configured tunnel interfaces and status.
You can view the details of a specific interface by adding the interface name to the command. The output displays packet counters for both directions of transmission. In this case, the tunnel is up and running, meaning that tunnel negotiations have completed successfully, but no traffic has traversed the tunnel yet.
You can reset counters using the command `clear interfaces counters`.



Use the command `show ip route` to verify the routing information. Note the static route via the tunnel interface is to the remote site network.



One common problem that will prevent the tunnel from being established is that the tunnel peers cannot communicate with one another.
The symptoms of the problem are: the tunnel interface will not appear in the list when you run the `show interfaces openvpn` command, and you cannot ping the remote peer.
To fix the problem, first check your configuration. You may have mistyped the address of the peer, or you may not have the correct address.
Next, troubleshoot the routing as much as you can between the two sites.



If you are using preshared keys, a common problem is that the peers do not have the same key file.

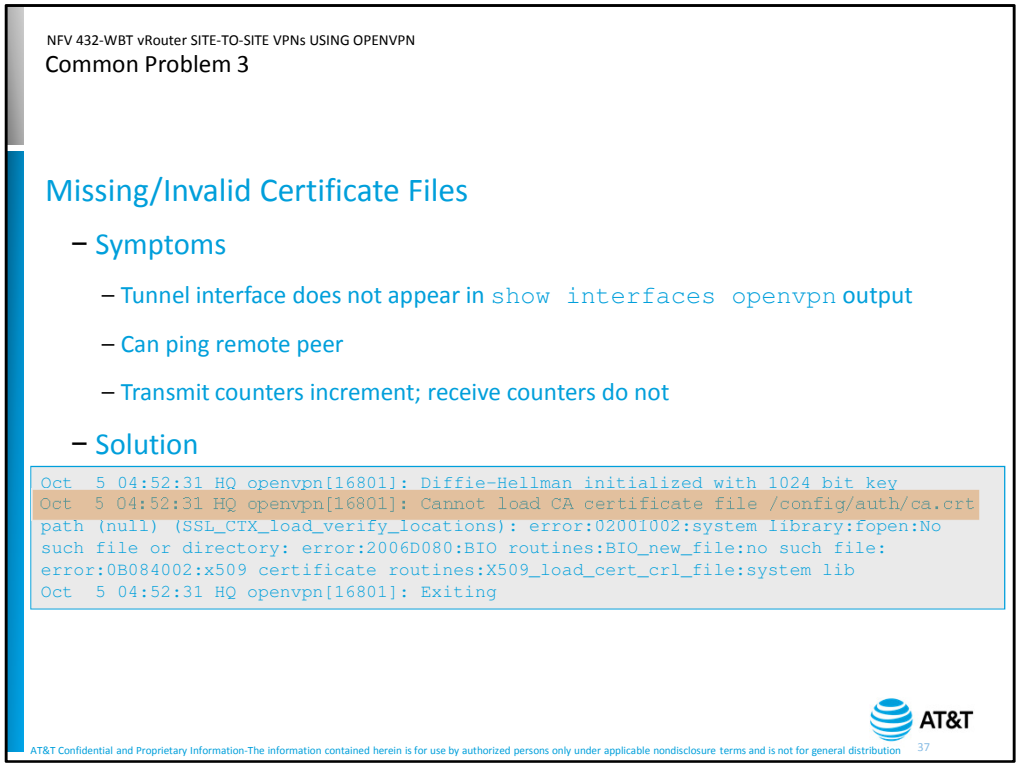
The symptoms of this problem are: the tunnel interface may not appear in the list of openvpn interfaces. However, you can ping the remote peer, so you know connectivity is not a problem.

You may also see transmit counters incrementing on the tunnel, but receive counters remaining at zero. This means that the device is transmitting to the peer, but has been unable to successfully decrypt any received packets.

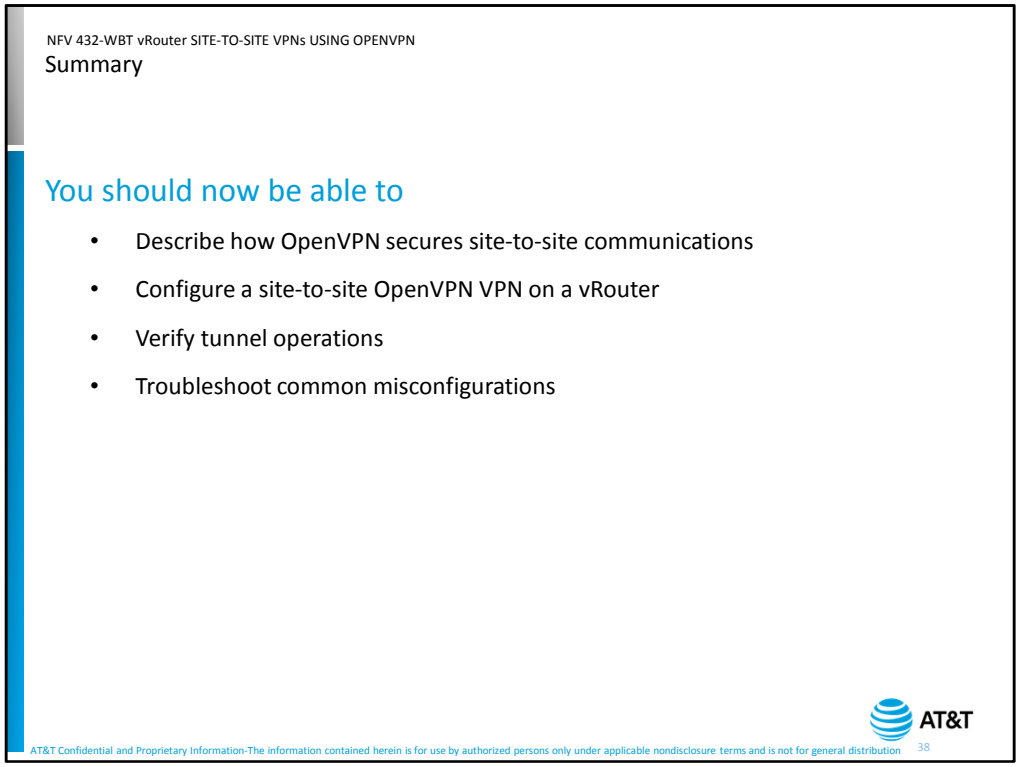
The solution to the problem is to first check that both peers have key files.

Next, you can use the Linux `cat` command to view the key files on each device. Both files should be the same.

If they are not, choose one of the files, and copy it to the other peer. Which file you choose does not matter, as long as both peers have the same key file.



If you are using TLS, the most common problem is missing or invalid certificate files.
The symptoms are the same as for a missing preshared key file.
To solve the problem, run the command `show log tail` on each peer and examine the output for information about which files are causing the problem.
In this example, the output indicates that we are missing the CA certificate. This output is from the Headquarters peer, as indicated by *HQ* in the output, so we know we need to add the CA certificate file to the Headquarters device.



This concludes the AT&T Vyatta 5600 vRouter course on configuring site-to-site VPNs using OpenVPN.

You should now be able to:

- Describe how OpenVPN secures site-to-site communications.
- Configure a site-to-site VPN using OpenVPN on a vRouter.
- Verify tunnel operations, and troubleshoot common misconfigurations.

We hope that this information has been useful to you, and that you will take additional AT&T University courses in the future.

Thank you.

NFV 432-WBT vRouter SITE-TO-SITE VPNs USING OPENVPN

End of Course – SITE-TO-SITE VPNS USING OPENVPN