- Overview of IPSEC -

Virtual Private Networks (VPNs)

A Virtual Private Network (VPN) provides a secure tunnel across a public (and thus, insecure) network. This provides a mechanism for organizations to connect users and offices together, without the high costs of dedicated leased lines.

VPNs are most often used across the Internet, the world's largest public network, providing users with access to email, documents, printers, and systems as if they were actually at their central office.

VPNs are generally used for two purposes:

- Client VPNs - connect home or "roaming" users to an office.
- Site-to-Site VPNs - connect remote offices to a main office.

What is IPSEC?

IPSEC, short for **IP Security**, is a suite of protocols, standards, and algorithms to secure traffic over an untrusted network, such as the Internet. IPSEC is supported on both Cisco IOS devices and PIX Firewalls.

IPSEC provides three core services:

- **Confidentiality** – prevents the theft of data, using **encryption**.
- **Integrity** – ensures that data is not tampered or altered, using a **hashing algorithm**.
- **Authentication** – confirms the identity of the host sending data, using **pre-shared keys** or a **Certificate Authority (CA)**.
- **Anti-replay** – prevents duplication of encrypted packets, by assigning a unique sequencing number.

The IPSEC standard is outlined in **RFC 2401**.

Confidentiality and Encryption

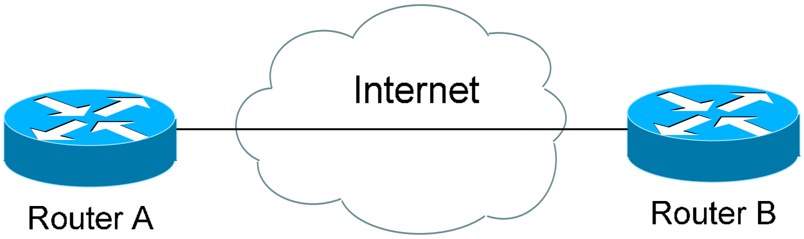
Data sent in clear-text across the Internet can easily be intercepted and stolen. Because of this, sensitive data should be **encrypted** when sent across an untrusted network or domain.

Keys are generated values used to both encrypt and decrypt data. The *longer* the key, the *more secure* that key is. The length of a key is measured in bits. Two "types" of keys exist:

Symmetric keys can be used to both encrypt and decrypt data. More specifically, the *same* key is used to both encrypt a packet (at the sending device) and then decrypt that packet (at the receiving device). Symmetric key encryption is efficient, but does not scale well in large environments.

Symmetric keys are not openly shared during data transmit, and must instead be installed on each machine *prior* to the transfer of data. This can be accomplished using a variety of (inefficient and insecure) methods: email, sneaker-net, and even snail-mail. Each device on a network would require every other device's symmetric key, and thus the lack of scalability.

Asymmetric keys require a separate key for encryption (the *public* key) and decryption (the *private* key). Public keys are openly exchanged between devices to encrypt data during transfer. Private keys are *never* exchanged.



Consider the above diagram. Assume we are using a public/private key infrastructure:

- Both Router A and Router B have their own unique **private** key.
- Both Router A and Router B exchange unique **public** keys.
- When Router B encrypts data destined for Router A, it uses Router A's **public** key. (and vice versa)
- Router A decrypts the data using its **private** key.

Only the private keys can decrypt the data. Thus, even if the data and the public key were intercepted, confidentiality is ensured.

Confidentiality and Encryption (continued)

Diffie-Hellman (D-H) Public Key Exchange is the most common standard used to create and exchange keys across insecure mediums. D-H is *not* used to encrypt data, but rather to *generate* the keys that are used to encrypt and decrypt data.

A variety of popular standards and protocols utilize D-H key exchange, including SSL (Secure Socket Layer), SSH (Secure Shell), and IPSEC.

The generated public keys encrypt data payload using one of several available encryption algorithms:

- **DES (Data Encryption Standard) – 56-bit key**
- **3DES (Triple Data Encryption Standard) – 168-bit key**
- **AES (Advanced Encryption Standard) - 128, 192, or 256-bit key**
- **Blowfish – up to a 448-bit key**

Additionally, the strength of a key is determined by the **D-H group** used to generate that key. There are several D-H groups:

- **Group 1 – 768 bits**
- **Group 2 – 1024 bits**
- **Group 5 – 2048 bits**

Data Integrity and Hashing

Data sent across the Internet can not only be stolen, but can also be maliciously altered.

To combat this, a **hashing algorithm** computes and appends a specific **hash value** as each packet is sent. Once the data is received, it is run through the hashing algorithm again. If the hash value is *different*, the packet was altered in transit.

Hashed Message Authentication Code (HMAC) is used to perform this hashing function. HMAC utilizes a *secret key* when computing the hash value, thus preventing an attacker from altering the packet and then *recomputing* the correct hash.

Two HMAC algorithms are commonly used:

- **HMAC-MD5 (Message-Digest 5)** – 128-bit hashed key
- **HMAC-SHA1 (Secure Hash Algorithm)** – 160-bit hashed key

Authentication

Another concern when sending data across the Internet is the *source* or *origin* of that data. It is possible to masquerade or spoof one's identity or address.

For an IPSEC VPN tunnel to be established, both sides of the tunnel must be authenticated. To accomplish this, either **pre-shared keys** or **RSA digital signatures** are used.

When using **pre-shared keys,** a secret string of text is used on each device to authenticate each other. This string must be pre-agreed upon and identical on each device. This string is then hashed into a digital signature.

When using **RSA Digital signatures,** a **Certificate Authority (CA)** is used to apply a verified digital signature.

One of the above options must be correctly configured before the VPN tunnel will become active.

Certificate Authorities

Remember, two methods exist to authenticate an IPSEC tunnel:

When using **pre-shared keys,** a secret string of text is used on each device to authenticate each other. This string must be pre-agreed upon and identical on each device. This string is then hashed into a digital signature.

When using **RSA Digital signatures,** a **Certificate Authority (CA)** is used to apply a verified digital signature. This provides a more scalable solution than pre-shared keys.

The certificate process works as follows:

1. First, a client creates a “blank” or **unsigned** certificate, and sends it to the CA. Included on this blank certificate is the client’s ID. This communication is secured using a D-H private/public key exchange.
2. Next, the CA computes an encrypted hash, which is applied to the blank certificate. Thus, the certificate is now **signed** with the CA’s **digital signature.** The signed certificate is sent back to the client, where it is stored until it is deleted or expires.
3. The client then sends the signed certificate, along with its keys, to any VPN peers, “authenticating” its origin.

REMEMBER: Digital signatures, and Certificate Authority servers, are *not* used to encrypt data. Instead, the digital signatures are used to **authenticate** a device’s keys. Essentially, the digital signature gives the key a stamp of authenticity.

Obviously, one must “trust” the CA that signs these digital certificates. This is why third-party CA’s are often used, such as VeriSign or Entrust.

Cisco IOS devices can function with several CA vendors:

- Microsoft Windows Certificate Services
- Entrust
- VeriSign

Regardless of the vendor chosen, the CA must support the **Simple Certificate Enrollment Protocol (SCEP)** to work with Cisco IOS devices. This is available on the server Resource Kit for Windows Certificate Servers.