6.2 VPN Protocols

There are various ways to achieve the encryption needs of a VPN. Certain network protocols are frequently used for VPNs. The two most commonly used protocols for this purpose are Point-to-Point Tunnelling Protocol (PPTP) and Layer 2 Tunnelling Protocol (L2TP). The part of the connection in which the data is encapsulated is referred to as the tunnel. L2TP is often combined with IPSec to achieve a high level of security.

6.2.1 PPTP

PPTP is a tunnelling protocol that enables an older connection protocol, PPP (Point-to-Point Protocol), to have its packets encapsulated within Internet Protocol (IP) packets and forwarded over any IP network, including the Internet itself. PPTP is often used to create VPNs. PPTP is an older protocol than L2TP or IPSec. Some experts consider PPTP to be less secure than L2TP or IPSec, but it consumes fewer resources and is supported by almost every VPN implementation. It is basically a secure extension to PPP.

PPTP was originally proposed as a standard in 1996 by the PPTP Forum—a group of companies that included Ascend Communications, ECI Telematics, Microsoft, 3Com, and U.S. Robotics. This group’s purpose was to design a protocol that would allow remote users to communicate securely over the Internet.

Although newer VPN protocols are available, PPTP is still widely used, because almost all VPN equipment vendors support PPTP. Another important benefit of PPTP is that it operates at layer 2 of the OSI model (the data link layer), allowing different networking protocols to run over a PPTP tunnel.

When users connect to a remote system, one of the facets of security is encrypting the data transmissions. However, this is not the only one. You must also authenticate the user. PPTP supports two separate technologies for accomplishing this: Extensible Authentication Protocol (EAP) and Challenge Handshake Authentication Protocol (CHAP).

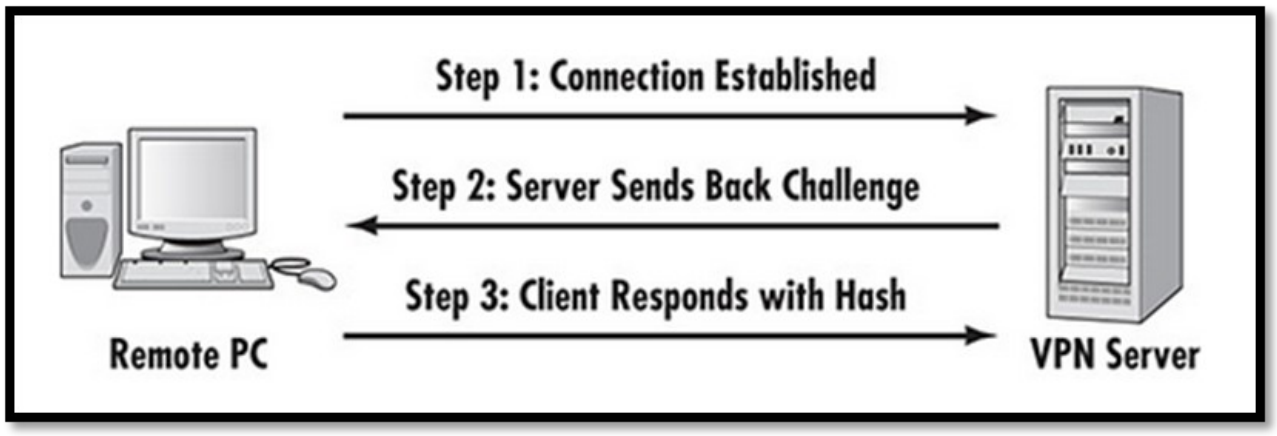
6.2.1.1 Extensible Authentication Protocol (EAP)

EAP was designed specifically with PPTP and is meant to work as part of PPP. EAP works from within PPP’s authentication protocol. It provides

a framework for several different authentication methods. EAP is meant to supplant proprietary authentication systems and includes a variety of authentication methods to be used, including passwords, challenge-response tokens, and public key infrastructure certificates.

6.2.1.2 Challenge Handshake Authentication Protocol (CHAP)

CHAP is actually a three-part handshaking (a term used to denote authentication processes) procedure. After the link is established, the server sends a challenge message to the client machine originating the link. The originator responds by sending back a calculated value, using a one-way hash function. The server checks the response against its own calculation of the expected hash value. If the values match, the authentication is acknowledged; otherwise the connection is usually terminated. This means that the authorization of a client connection has three stages.



What makes CHAP particularly interesting is that it periodically repeats the process. This means that even after a client connection is authenticated, CHAP repeatedly seeks to re-authenticate that client, providing a robust level of security.

6.2.2 L2TP

Layer 2 Tunnelling Protocol is an extension or enhancement of the Point-to-Point Tunnelling Protocol. This is often used to operate virtual private networks over the Internet. Essentially, it is a new and improved version of PPTP. As its name suggests, it operates on the data link layer of the OSI model (like PPTP). Many experts consider both the PPTP and L2TP to be less secure than IPSec. However, it is not uncommon to use IPSec together with L2TP in order to create a secure VPN connection.

Like PPTP, L2TP supports EAP and CHAP. However, it also offers support to six additional authentication methods:

- EAP
- CHAP
- MS-CHAP
- PAP
- SPAP
- Kerberos

6.2.2.1 MS-CHAP

As the name suggests, MS-CHAP is a Microsoft-specific extension to CHAP. Microsoft created MS-CHAP to authenticate remote Windows workstations. The goal is to make the functionality available on the LAN to remote users while integrating the encryption and hashing algorithms used on Windows networks.

Wherever possible, MS-CHAP is consistent with standard CHAP. However, some basic differences between MS-CHAP and standard CHAP include the following:

- The MS-CHAP response packet is in a format which is designed for compatibility with Microsoft's Windows networking products.
- The MS-CHAP format does not require the authenticator to store a clear-text or reversibly encrypted password.
- MS-CHAP provides authenticator-controlled authentication retry and password-changing mechanisms. These retry and password-changing mechanisms are compatible with the mechanisms used in Windows networks.
- MS-CHAP defines a set of reason-for-failure codes that are returned in the failure packet's message field if the authentication fails. These are codes that Windows software is able to read and interpret, thus providing the user with the reason for the failed authentication.

6.2.2.2 PAP

Password Authentication Protocol (PAP) is the most basic form of authentication. PAP allows a user's name and password to be transmitted over a network and compared to a table of name-password pairs. Typically, the passwords stored in the table are encrypted. However, the main weakness of PAP is that the

transmissions of the passwords are in clear text, as well as unencrypted. The basic authentication feature built into the HTTP protocol uses PAP. This method is no longer used and is only presented for historical purposes.

6.2.2.3 SPAP

Shiva Password Authentication Protocol (SPAP) is a proprietary version of PAP. Most experts consider SPAP somewhat more secure than PAP because the username and password are both encrypted when they are sent, unlike PAP.

Due to the fact that SPAP encrypts passwords, someone capturing authentication packets will not be able to read the SPAP password. However, SPAP is still susceptible to playback attacks (that is, a person records the exchange and plays the message back to gain fraudulent access). Playback attacks are possible because SPAP always uses the same reversible encryption method to send the passwords over the wire.

6.2.2.4 Kerberos

Kerberos is one of the most well-known network authentication protocols. It was developed at MIT and it is named after the mythical three-headed dog that guarded the gates to Hades.

Kerberos works by sending messages back and forth between the client and the server. The actual password (or even a hash of the password) is never sent. That makes it impossible for someone to intercept it. What happens instead is that the username is sent. The server then looks up to the stored hash of that password, and uses it as an encryption key to encrypt data and send it back to the client. The client then takes the password the user entered and uses that as a key to decrypt the data. If the user enters the wrong password, then it will never get decrypted. This is a clever way to verify the password without ever being transmitted. Authentication happens with UDP (User Datagram Protocol) on port 88.

After the user’s username is sent to the authentication service (AS), that AS will use the hash of the user password that is stored as a secret key to encrypt the following two messages that get sent to the client:

- **Message A:** Contains Client/TGS (Ticket Granting Service) session key encrypted with secret key of client
- **Message B:** Contains TGT (Ticket Granting Ticket) that includes client ID, client network address, and validity period

Remember, both of these messages are encrypted using the key the AS generated.

Then the user attempts to decrypt message A with the secret key generated by the client hashing the user's entered password. If that entered password does not match the password the AS found in the database, then the hashes will not match, and the decryption won't work. If it does work, then message A contains the Client/TGS session key that can be used for communication with the TGS. Message B is encrypted with the TGS secret key and cannot be decrypted by the client.

Now the user is authenticated into the system. However, when the user actually requests a service, more communication through messages is required. When requesting services, the client sends the following messages to the TGS:

- **Message C:** Composed of the TGT from message B and the ID of the requested service
- **Message D:** Authenticator (which is composed of the client ID and the timestamp), encrypted using the Client/TGS session key

Upon receiving messages C and D, the TGS retrieves message B out of message C. It decrypts message B using the TGS secret key which results to "Client/TGS session key". Using this key, the TGS decrypts message D (Authenticator) and sends the following two messages to the client:

- **Message E:** Client-to-server ticket (which includes the client ID, client network address, validity period, and client/server session key) encrypted using the service's secret key
- **Message F:** Client/server session key encrypted with the Client/TGS session key

Upon receiving messages E and F from TGS, the client has enough information to authenticate itself to the Service Server (SS). The client connects to the SS and sends the following two messages:

- **Message E:** From the previous step (the client-to-server ticket, encrypted using service's secret key)

- **Message G:** A new Authenticator, which includes the client ID and timestamp and is encrypted using the client/server session key

The SS decrypts the ticket (message E) using its own secret key to retrieve the client/server session key. Using the session key, the SS decrypts the Authenticator and sends the following message to the client to confirm its identity and willingness to serve the client:

- **Message H:** The timestamp found in client's Authenticator

The client decrypts the confirmation (message H) using the client/server session key and checks whether the timestamp is correct. If so, then the client can trust the server and can start issuing service requests to the server. The server provides the requested services to the client.

Below are some Kerberos terms to know:

- **Principal:** A server or client that Kerberos can assign tickets to.
- **Authentication Service (AS):** Service that authorizes the principal and connects them to the Ticket Granting Server. Note some books/sources say server rather than service.
- **Ticket Granting Service (TGS):** Provides tickets.
- **Key Distribution Centre (KDC):** A server that provides the initial ticket and handles TGS requests. Often it runs both AS and TGS services.

Realm: A boundary within an organisation. Each realm has its own AS and TGS.

- **Remote Ticket Granting Server (RTGS):** A TGS in a remote realm.
- **Ticket Granting Ticket (TGT):** The ticket that is granted during the authentication process.
- **Ticket:** Used to authenticate to the server. Contains identity of client, session key, timestamp, and checksum. Encrypted with server's key.
- **Session key:** Temporary encryption key.
- **Authenticator:** Proves session key was recently created. Often expires within 5 minutes.