Computer Network Security Protocols 17

17.1 Introduction

The rapid growth of the Internet and corresponding Internet community has fueled a rapid growth of both individual and business communications leading to the growth of e-mail and e-commerce. In fact, studies now show that the majority of the Internet communication content is e-mail content. The direct result of this has been the growing concern and sometimes demand for security and privacy in electronic communication and e-commerce. Security and privacy are essential if individual communication is to continue and e-commerce is to thrive in cyberspace. The call for and desire for security and privacy has led to the advent of several proposals for security protocols and standards. Among these are Secure Socket Layer (SSL) and Transport Layer Security (TLS) protocols, secure IP (IPsec), Secure HTTP (S-HTTP), secure e-mail (PGP and S/MIME), DNDSEC, SSH, and others. Before we proceed with the discussion of these and others, we want to warn the reader of the need for a firm understanding of the network protocol stack; otherwise, go back and look over the material in Chap. 1 before continuing. To make sure that the reader understands the security of networks based on protocol stacks, we will consider these protocols in both the ISO and TCP/IP stacks. Figure 17.1 shows the relationships between the ISO and TCP/IP stacks.

We will discuss these protocols and standards within the framework of the network protocol stack as follows:

- Application-level security (for TCP/IP) and application and presentation (for ISO)—RADIUS, TACACS, PGP, S/MIME, S-HTTP, HTTPS, SET, SSH, and Kerberos
- Transport-level security (for TCP/IP) and session and transport (for ISO)—SSL and TLS
- Network-level security for both TCP/IP and ISO—PPTP, L2TP, IPsec, and VPNs

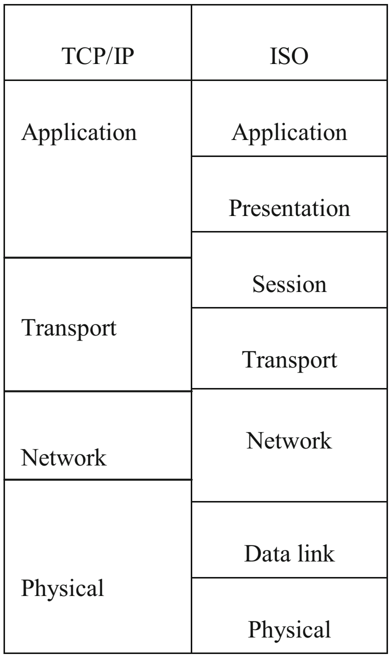

Fig. 17.1 Relationships between ISO and TCP/IP Network Protocol stacks

- Physical link-level security (for TCP/IP) and data link and physical (for ISO)—packet filters, NAT, CHAP, and PAP

17.2 Application-Level Security

All the protocols in this section are application layer protocols, which means that they reside on both ends of the communication link. They are all communication protocols ranging from simple text to multimedia including graphics, video, audio, and so on. In the last 10 years, there has been almost an explosion in the use of electronic communication, both mail and multimedia content, that has resulted in booming e-commerce and almost unmanageable personal e-mails, much of it private or intended to be private anyway, especially e-mails. Along with this explosion, there has been a growing demand for confidentiality and authenticity of private communications. To meet these demands, several schemes have been developed to offer both confidentiality and authentication of these communications. We will give a brief description of each. These four protocols and the standards are shown in the application layer of the network stack in Fig. 17.2.

TCP/IP	ISO	Security Protocols
Application	Application Presentation	RADIUS, TACAS, PGP, S/MIME, S-HTTP, HTTPS, SET, SSH, and KERBEROS

Fig. 17.2 Application layer security protocols and standard

17.2.1 Remote Authentication Dial-In User Service (RADIUS)

RADIUS is a server for remote user authentication and accounting. It is one of the most, if not the most, widely used dial-up authentication protocols. During dial-up, it offers authentication and authorization for those intending to use the network. It is one of a class of Internet dial-in security protocols that include Password Authentication Protocol (PAP) and Challenge-Handshake Authentication Protocol (CHAP). Although it is mainly used by Internet service providers (ISPs) to provide authentication and accounting for remote users, it can be used also in private networks to centralize authentication and accounting services on the network for all dial-in connections for service. A number of organizations are having their employees work off-site, and many of these employees may require to dial-in for services. Vendors and contractors may need to dial-in for information and specifications. RADIUS is a good tool for all these types of off-site authentication and accounting.

Let us look at RADIUS's main components: authentication and accounting protocols.

17.2.1.1 Authentication Protocols

Upon contact with the RADIUS server, a user is authenticated from the data supplied by the user to the server either directly by answering the terminal server's log-in/password prompts or using PAP or CHAP protocols. The user's personal data can also be obtained from one of the following places [1]:

- *System database*: The user's log-in ID and password are stored in the password database on the server.
- *Internal database*: The user's log-in ID and password can also be encrypted by either MD5 or DES hash and then stored in the internal RADIUS database.
- *SQL authentication*: The user's details can also be stored in an SQL database.

17.2.1.2 Accounting Protocols

RADIUS has three built-in accounting schemes: Unix accounting, detailed accounting, and SQL accounting.

17.2.1.3 Key Features of RADIUS

RADIUS has several features including [2]:

- *Client/Server Model*: In client/server model, the client is responsible for passing user information to designated RADIUS servers and then acting on the response which is returned. RADIUS servers, on their part, are responsible for receiving user connection requests, authenticating the user, and then returning all configuration information necessary for the client to deliver service to the user.
- *Network Security*: All correspondence between the client and RADIUS server is authenticated through the use of a shared secret, which is never sent over the network. User passwords are sent encrypted between the client and RADIUS server.
- *Flexible Authentication Mechanisms*: The RADIUS server can support a variety of methods to authenticate a user. When it is provided with the username and the original password given by the user, it can support PPP PAP or CHAP, Unix log-in, and other authentication mechanisms.
- *Extensible Protocol*: RADIUS is a fully extensible system. It supports two extension languages: the built-in Rewrite language and Scheme. Based on RFC 2138, all transactions are comprised of variable length Attribute-Length-Value 3-tuples. New attribute values can be added without disturbing existing implementations of the protocol.

17.2.2 Terminal Access Controller Access Control System (TACACS+)

This protocol is commonly referred to as “tac-plus” is a commonly used method of authentication protocol. Developed by Cisco Systems, TACACS+ is a strong protocol for dial-up, and it offers the following [3]:

- Authentication—arbitrary length and content authentication exchange which allows many authentication mechanisms to be used with it.
- Authorization.
- Auditing—a recording of what a user has been doing and in TACACS+, it serves two purposes:
  – To account for services used
  – To audit for security services

TACACS+ has a “+” sign because Cisco has extended it several times and has derivatives that include the following:

- TACACS—the original TACACS, which offers combined authentication and authorization
- XTACACS, which separated authentication, authorization, and accounting
- TACACS+, which is XTACACS plus extensions of control and accounting attributes

17.2.3 Pretty Good Privacy (PGP)

The importance of sensitive communication cannot be underestimated. Sensitive information, whether in motion in communication channels or in storage, must be protected as much as possible. The best way, so far, to protect such information is to encrypt it. In fact, the security that the old snail mail offered was based on a seemingly protective mechanism similar to encryption when messages were wrapped and enclosed in envelopes. There was, therefore, more security during the days of snail mail because it took more time and effort for someone to open somebody's mail. First, one had to get access to it, which was no small task. Then, one had to steam the envelope in order to open it and seal it later so that it looks unopened after. There were more chances of being caught doing so. Well, electronic communication has made it easy to intercept and read messages in the clear.

So encryption of e-mails and any other forms of communication is vital for the security, confidentiality, and privacy of everyone. This is where PGP comes in and this is why PGP is so popular today. In fact, currently PGP is one of the popular encryption and digital signatures schemes in personal communication.

Pretty Good Privacy (PGP), developed by Phil Zimmermann, is a public key cryptosystem. As we saw in Chap. 9, in public key encryption, one key is kept secret and the other key is made public. Secure communication with the receiving party (with a secret key) is achieved by encrypting the message to be sent using the recipient's public key. This message then can be decrypted only using the recipient's secret key.

PGP works by creating a *circle of trust* among its users. In the circle of trust, users, starting with two, form a key ring of public key/name pairs kept by each user. Joining this "trust club" means trusting and using the keys on somebody's key ring. Unlike the standard PKI infrastructure, this circle of trust has a built-in weakness that can be penetrated by an intruder. However, since PGP can be used to sign messages, the presence of its digital signature is used to verify the authenticity of a document or file. This goes a long way in ensuring that an e-mail message or file just downloaded from the Internet is both secure and untampered with.

PGP is regarded as hard encryption, which is impossible to crack in the foreseeable future. Its strength is based on algorithms that have survived extensive public review and are already considered by many to be secure. Among these algorithms are RSA which PGP uses for encryption, DSS, and Diffie-Hellman for public key encryption; CAST-128, IDEA, and 3DES for conventional encryption; and SHA-1 for hashing. The actual operation of PGP is based on five services: authentication, confidentiality, compression, e-mail compatibility, and segmentation [4].

17.2.3.1 Authentication

PGP provides authentication via a digital signature scheme. The hash code (MAC) is created using a combination of SHA-1 and RSA to provide an effective digital signature. It can also create an alternative signature using DSS and SHA-1. The signatures are then attached to the message or file before sending. PGP, in addition, supports unattached digital signatures. In this case, the signature may be sent separately from the message.

17.2.3.2 Confidentiality

PGP provides confidentiality by encrypting messages before transmission. PGP encrypts messages for transmission and storage using conventional encryption schemes such as CAST-128, IDEA, and 3DES. In each case, a 64-bit cipher feedback mode is used. As in all cases of encryption, there is always a problem of key distribution; so PGP uses a conventional key once. This means for each message to be sent, the sender mints a brand new 128-bit session key for the message. The session key is encrypted with RSA or Diffie-Hallman using the recipient's public key; the message is encrypted using CAST-128 or IDEA or 3DES together with the session key. The combo is transmitted to the recipient. Upon receipt, the receiver uses RSA with his or her private key to encrypt and recover the session key which is used to recover the message. See Fig. 17.8.

17.2.3.3 Compression

PGP compresses the message after applying the signature and before encryption. The idea is to save space.

17.2.3.4 E-mail Compatibility

As we have seen above, PGP encrypts a message together with the signature (if not sent separately) resulting into a stream of arbitrary 8-bit octets. But since many e-mail systems permit only use of blocks consisting of ASCII text, PGP accommodates this by converting the raw 8-bit binary streams into streams of printable ASCII characters using a radix-64 conversion scheme. On receipt, the block is converted back from radix-64 format to binary. If the message is encrypted, then a session key is recovered and used to decrypt the message. The result is then decompressed. If there is a signature, it has to be recovered by recovering the transmitted hash code and comparing it to the receiver's calculated hash before acceptance.

17.2.3.5 Segmentation

To accommodate e-mail size restrictions, PGP automatically segments e-mail messages that are too long. However, the segmentation is done after all the housekeeping is done on the message, just before transmitting it. So the session key and signature appear only once at the beginning of the first segment transmitted. At receipt, the receiving PGP strips off all e-mail headers and reassembles the original mail.

PGP's popularity and use have so far turned out to be less than anticipated because of two reasons: first, its development and commercial distribution after Zimmermann sold it to Network Associates, which later sold it to another company, did not do well; second, its open-source cousin, the OpenPGP, encountered market problems including the problem of ease of use. Both OpenPGP and commercial PGP are difficult to use because it is not built into many e-mail clients. This implies that any two communicating users who want to encrypt their e-mail using PGP have to manually download and install PGP, a challenge and an inconvenience to many users.

17.2.4 Secure/Multipurpose Internet Mail Extension (S/MIME)

Secure/Multipurpose Internet Mail Extension (S/MIME) extends the protocols of Multipurpose Internet Mail Extensions (MIME) by adding digital signatures and encryption to them. To understand S/MIME, let us first make a brief digression and look at MIME. MIME is a technical specification of communication protocols that describes the transfer of multimedia data including pictures, audio, and video. The MIME protocol messages are described in RFC 1521; a reader with further interest in MIME should consult RFC 1521. Because Web contents such as files consist of hyperlinks that are themselves linked onto other hyperlinks, any e-mail must describe this kind of interlinkage. That is what a MIME server does whenever a client requests for a Web document. When the Web server sends the requested file to the client's browser, it adds a MIME header to the document and transmits it [5]. This means, therefore, that such Internet e-mail messages consist of two parts: the header and the body.

Within the header, two types of information are included: *MIME type* and *subtype*. The MIME type describes the general file type of the transmitted content type such as image, text, audio, application, and others. The subtype carries the specific file type such as *jpeg* or gif, tiff, and so on. For further information on the structure of a MIME header, please refer to RFC 822. The body may be unstructured or it may be in MIME format which defines how the body of an e-mail message is structured. What is notable here is that MIME does not provide any security services.

S/MIME was then developed to add security services that have been missing. It adds two cryptographic elements: encryption and digital signatures [4].

17.2.4.1 Encryption

S/MIME supports three public key algorithms to encrypt session keys for transmission with the message. These include Diffie-Hallman as the preferred algorithm, RSA for both signature and session keys, and Triple DES.

17.2.4.2 Digital Signatures

To create a digital signature, S/MIME uses a hash function of either 160-bit SHA-1 or MD5 to create message digests. To encrypt the message digests to form a digital signature, it uses either DSS or RSA.

17.2.5 Secure HTTP (S-HTTP)

Secure HTTP (S-HTTP) extends the Hypertext Transfer Protocol (HTTP). When HTTP was developed, it was developed for a Web that was simple, that did not have dynamic graphics, and that did not require, at that time, hard encryption for end-to-end transactions that have since developed. As the Web became popular for businesses, users realized that current HTTP needed more cryptographic and graphic improvements if it were to remain the e-commerce backbone it had become.

Responding to this growing need for security, the Internet Engineering Task Force called for proposals that will develop Web protocols, probably based on current HTTP, to address these needs. In 1994, such protocol was developed by Enterprise Integration Technologies (EIT). IET's protocols were, indeed, extensions of the HTTP. S-HHTP extended HTTP by extending HTTP's instructions and added security facilities using encryptions and support for digital signatures. Each S-HTTP file is either encrypted, contains a digital certificate, or both. S-HTTP design provides for secure communications, primarily commercial transactions, between a HTTP client and a server. It does this through a wide variety of mechanisms to provide for confidentiality, authentication, and integrity while separating policy from mechanism. The system is not tied to any particular cryptographic system, key infrastructure, or cryptographic format [6].

HTTP messages contain two parts: the header and the body of the message. The header contains instructions to the recipients (browser and server) on how to process the message's body. For example, if the message body is of the type like MIME, Text, or HTML, instructions must be given to display this message accordingly. In the normal HTTP, for a client to retrieve information (text-based message) from a server, a client-based browser uses HTTP to send a request message to the server that specifies the desired resource. The server, in response, sends a message to the client that contains the requested message. During the transfer transaction, both the client browser and the server use the information contained in the HTTP header to negotiate formats they will use to transfer the requested information. Both the server and client browser may retain the connection as long as it is needed: otherwise, the browser may send message to the server to close it.

The S-HTTP extends this negotiation between the client browser and the server to include the negotiation for security matters. Hence, S-HTTP uses additional headers for message encryption, digital certificates, and authentication in the HTTP format which contains additional instructions on how to decrypt the message body. Tables 17.1 and 17.2 show header instructions for both HTTP and S-HTTP. The HTTP headers are encapsulated into the S-HTTP headers. The headers give a variety of options that can be chosen from as a client browser and the server negotiate for information exchange. All headers in S-HTTP are optional, except "Content Type" and "Content-Privacy-Domain."

To offer flexibility, during the negotiation between the client browser and the server, for the cryptographic enhancements to be used, the client and server must agree on four parts: property, value, direction, and strength. If agents are unable to

Table 17.1 S-HTTP headers

S-HTTP header	Purpose	Options
Content-Privacy-Domain	For compatibility with PEM-based secure HTTP	RSA's PKCS-7 (Public Key Cryptography Standard 7), "Cryptographic Message Syntax Standard," RFC-1421 style PEM, and PGP 2.6 format
Content-transfer-encoding	Explains how the content of the message is encoded	7, 8 bit
Content-type	Standard header	HTTP
Prearranged-Key-Info	Information about the keys used in the encapsulation	DEK (data exchange key) used to encrypt this message

Table 17.2 HTTP headers

HTTP header	Purpose	Options
Security scheme	Mandatory, specifies protocol name and version	S-HTTP/1.1
Encryption identity	Identity names the entity for which a message is encrypted. Permits return encryption under public key without others signing first	DN-1485 and Kerberos
Certificate info	Allows a sender to send a public key certificate in a message	PKCS-7, PEM
Key assign (exchange)	The message used for actual key exchanges	Krb-4, Krb-5 (Kerberos)
Nonces	Session identifiers, used to indicate the freshness of a session	

S-HTTP-Key-Exchange-Algorithms: *recv-required* = RSA, Kerb-5

discover a common set of algorithms, appropriate actions are then be taken. Adam Shastack [5] gives the following example as a negotiation line:

This means that messages *received* by this machine are *required* to use Kerberos 5 or RSA encryption to exchange keys. The choices for the *(recv-required)* modes are *(recv || orig)-(optional || required || refused)*. Where key length specifications are necessary in case of variable key length ciphers, this is then specifically referred to as cipher[length], or cipher[L1–L2], where length of key is length or, in the case of L1–L2, is between L1 and L2, inclusive [5].

Other headers in the S-HTTP negotiations could be [5, 6]:

- S-HTTP-Privacy-Domains
- S-HTTP-Certificate-Types
- S-HTTP-Key-Exchange-Algorithms
- S-HTTP-Signature-Algorithms
- S-HTTP-Message-Digest-Algorithms
- S-HTTP-Symmetric-Content-Algorithms
- S-HTTP-Symmetric-Header-Algorithms

- S-HTTP-Privacy-Enhancements
- Your-Key-Pattern

We refer a reader interested in more details of these negotiations to Adam Shastack's paper.

We had pointed out earlier that S-HTTP extends HTTP by adding message encryption, digital signature, and message and sender authentication. Let us see how these are incorporated into HTTP to get S-HTTP.

17.2.5.1 Cryptographic Algorithm for S-HTTP

S-HTTP uses a symmetric key cryptosystem in the negotiations that prearranges symmetric session keys and a challenge-response mechanism between communicating parties. Before the server can communicate with the client browser, both must agree upon an encryption key. Normally the process would go as follows: The client's browser would request the server for a page. Along with this request, the browser lists encryption schemes it supports and also includes its public key. Upon receipt of the request, the server responds to the client browser by sending a list of encryption schemes it also supports. The server may, in addition, send the browser a session key encrypted by the client's browser's public key, now that it has it. If the client's browser does not get a session key from the server, it then sends a message to the server encrypted with the server's public key. The message may contain a session key or a value the server can use to generate a session key for the communication.

Upon the receipt of the page/message from the server, the client's browser, if possible, matches the decryption schemes in the S-HTTP headers (recall this was part of the negotiations before the transfer), which include session keys, and then decrypts the message [6]. In HTTP transactions, once the page has been delivered to the client's browser, the server would disconnect. However, with S-HTTP, the connection remains until the client browser requests the server to do so. This is helpful because the client's browser encrypts each transmission with this session key.

Cryptographic technologies used by S-HTTP include Privacy-Enhanced Mail (PEM), Pretty Good Privacy (PGP), and Public Key Cryptography Standard 7 (PKGS-7). Although S-HTTP uses encryption facilities, non-S-HTTP browsers can still communicate with an S-HTTP server. This is made possible because S-HTTP does not require that the user preestablishes public keys in order to participate in a secure transaction. A request for secure transactions with an S-HTTP server must originate from the client browser. See Fig. 17.1.

Because a server can deal with multiple requests from client browsers, S-HTTP supports multiple encryptions by supporting two transfer mechanisms: one that uses public key exchange, usually referred to as *in-band*, and one that uses a third-party Public Key Authority (PKA) that provides session keys using public keys for both clients and servers.

17.2.5.2 Digital Signatures for S-HTTP

S-HTTP uses *SignedData* or *SignedAndEnvelopedData* signature enhancement of PKCS-7 [6]. S-HTTP allows both certificates from a certificate authority (CA) and a self-signed certificate (usually not verified by a third party). If the server requires a digital certificate from the client's browser, the browser must attach a certificate then.

17.2.5.3 Message and Sender Authentication in S-HTTP

S-HHTP uses an authentication scheme that produces a MAC. The MAC is actually a digital signature computed from a hash function on the document using a shared secret code.

17.2.6 Hypertext Transfer Protocol over Secure Socket Layer (HTTPS)

HTTPS is the use of Secure Sockets Layer (SSL) as a sub-layer under the regular HTTP in the application layer. It is also referred to as Hypertext Transfer Protocol over Secure Socket Layer (HTTPS) or HTTP over SSL, in short. HTTPS is a Web protocol developed by Netscape, and it is built into its browser to encrypt and decrypt user page requests as well as the pages that are returned by the Web server. HTTPS uses port 443 instead of HTTP port 80 in its interactions with the lower layer, TCP/IP. Probably to understand well how this works, the reader should first go over Sect. 17.3.1, where SSL is discussed.

17.2.7 Secure Electronic Transactions (SET)

SET is a cryptographic protocol developed by a group of companies that included Visa, Microsoft, IBM, RSA, Netscape, MasterCard, and others. It is a highly specialized system with complex specifications contained in three books with book one dealing with the business description, book two a programmer's guide, and book three giving the formal protocol description. Book one spells out the business requirements that include the following [4]:

- Confidentiality of payment and ordering information
- Integrity of all transmitted data
- Authentication of all cardholders
- Authenticating that a merchant can accept card transactions based on relationship with financial institution
- Ensuring the best security practices and protection of all legitimate parties in the transaction
- Creating protocols that neither depend on transport security mechanism nor prevent their use

- Facilitating and encouraging interoperability among software and network providers

Online credit and debit card activities that must meet those requirements may include one or more of the following: cardholder registration, merchant registration, purchase request, payment authorization, funds transfer, credits reversals, and debit cards. For each transaction, SET provides the following services: authentication, confidentiality, message integrity, and linkage [4, 7].

17.2.7.1 Authentication

Authentication is a service used to authenticate everyone in the transacting party that includes the customer, the merchant, the bank that issued the customer's card, and the merchant's bank, using X.509v3 digital signatures.

17.2.7.2 Confidentiality

Confidentiality is a result of encrypting all aspects of the transaction to prevent intruders from gaining access to any component of the transaction. SET uses DES for this.

17.2.7.3 Message Integrity

Again this is a result of encryption to prevent any kind of modification to the data such as personal data and payment instructions involved in the transaction. SET uses SHA-1 hash codes used in the digital signatures.

17.2.7.4 Linkage

Linkage allows the first party in the transaction to verify that the attachment is correct without reading the contents of the attachment. This helps a great deal in keeping the confidentiality of the contents.

SET uses public key encryption and signed certificates to establish the identity of everyone involved in the transaction and to allow every correspondence between them to be private.

The SET protocols involved in a transaction have several representations, but every one of those representations has the following basic facts: the actors and the purchase-authorization-payment control flow.

The actors involved in every transaction are as follows [4]:

- The buyer—usually the cardholder.
- The merchant—fellow with the merchandise the buyer is interested in.
- The merchant bank—the financial institution that handles the merchant's financial transactions.
- The customer bank—usually the bank that issues the card to the customer. This bank also authorizes electronic payments to the merchant account upon authorization of payment request from the customer. This bank may sometimes set up another entity and charge it with payment authorizations.

- Certificate authority (CA)—that issues X.509v3 certificates to the customer and merchant.

Purchase-authorization-payment control flow. This flow is initiated by the customer placing a purchase order to the merchant and is concluded by the customer bank sending a payment statement to the customer. The key cryptographic authentication element in SET is the *dual signature*. The dual signature links two messages (payment information and order information) intended for two different recipients, the merchant getting merchandise information and the customer bank getting payment information. The dual signature keeps the two bits of information separate letting the intended party see only the part they are authorized to see. The customer creates a dual signature by hashing the merchandise information and also payment information using SHA-1, concatenates the two, hashes them again, and encrypts the result using his or her private key before sending them to the merchant. For more details on dual signatures, the reader is referred to *Cryptography and Network Security: Principles and Practice,* Second Edition, by William Stallings. Let us now look at the purchase-authorization-payment control flow [4, 7].

- Customer initiates the transaction by sending to the merchant a purchase order and payment information together with a dual signature.
- The merchant, happy to receive an order from the customer, strips off the merchant information, verifies customer purchase order using his or her certificate key, and forwards the payment information to his or her bank.
- The merchant bank forwards the payment information from the customer to the customer bank.
- The customer bank, using the customer's certificate key, checks and authorizes the payments and informs the merchant's bank.
- The merchant's bank passes the authorization to the merchant, who releases the merchandise to the customer.
- The customer bank bills the customer.

17.2.8 Kerberos

Kerberos is a network authentication protocol. It is designed to allow users, clients, and servers authenticate themselves to each other. This mutual authentication is done using secret key cryptography. Using secret key encryption or, as it is commonly known, conventional encryption, a client can prove its identity to a server across an insecure network connection. Similarly, a server can also identify itself across the same insecure network connection. Communication between the client and the server can be secure after the client and server have used Kerberos to prove their identities. From this point on, subsequent communication between the two can be encrypted to ensure privacy and data integrity.

In his paper *The Moron's Guide to Kerberos, Version 1.2.2,* Brian Tung [7], in a simple but interesting example, likens the real-life self-authentication we always do with the presentation of driver licenses on demand to that of Kerberos.

Kerberos client/server authentication requirements are as follows [5]:

- Security—that Kerberos is strong enough to stop potential eavesdroppers from finding it to be a weak link.
- Reliability—that Kerberos is highly reliable, employing a distributed server architecture where one server is able to back up another. This means that Kerberos system is fail safe, meaning graceful degradation, if it happens.
- Transparency—that users are not aware that authentication is taking place beyond providing passwords.
- Scalability—that Kerberos systems accept and support new clients and servers.

To meet these requirements, Kerberos designers proposed a third-party-trusted authentication service to arbitrate between the client and server in their mutual authentication. Figure 17.3 shows the interaction between the three parties.

The actual Kerberos authentication process is rather complex, probably more complex than all the protocols we have seen so far in this chapter. So to help the reader grasp the concept, we are going to follow what many writers on Kerberos have done and go via an example and Fig. 17.3. And here we go [5]:

On a Kerberos network, suppose user A wants to access a document on server B. Both principals in the transaction do not trust each other. So the server must demand assurances that A is who he or she says he or she is. So just like in real life, when you are seeking a service from demands that you show proof of what you claim you are by pulling out a driver's license with a picture of you on it, Kerberos also demands proof. In Kerberos, however, A must present a *ticket* to B. The ticket is issued by a Kerberos *authentication server* (AS). Both A and B trust the AS. So A anticipating that B will demand proof works on it by digitally signing the request to access the document held by B with A's private key and encrypting the request with B's public key. A then sends the encrypted request to AS, the trusted server. Upon receipt of the request, AS verifies that it is A who sent the request by analyzing A's digital signature. It also checks A's access rights to the requested document. AS has those lists for all the servers in the Kerberos system. AS then mints a *ticket* that contains a session key and B's access information, uses A's public key to encrypt it, and sends it to A. In addition, AS mints a similar ticket for B which contains the same information as that of A. The ticket is transmitted to B. Now AS's job is almost done after connecting both A and B. They are now on their own. After the connection, both A and B compare their tickets for a match. If the tickets match, the AS has done its job, and A and B start communicating as A accesses the requested document on B. At the end of the session, B informs AS to recede the ticket for this session. Now if A wants to communicate with B again for whatever request, a new ticket for the session is needed.

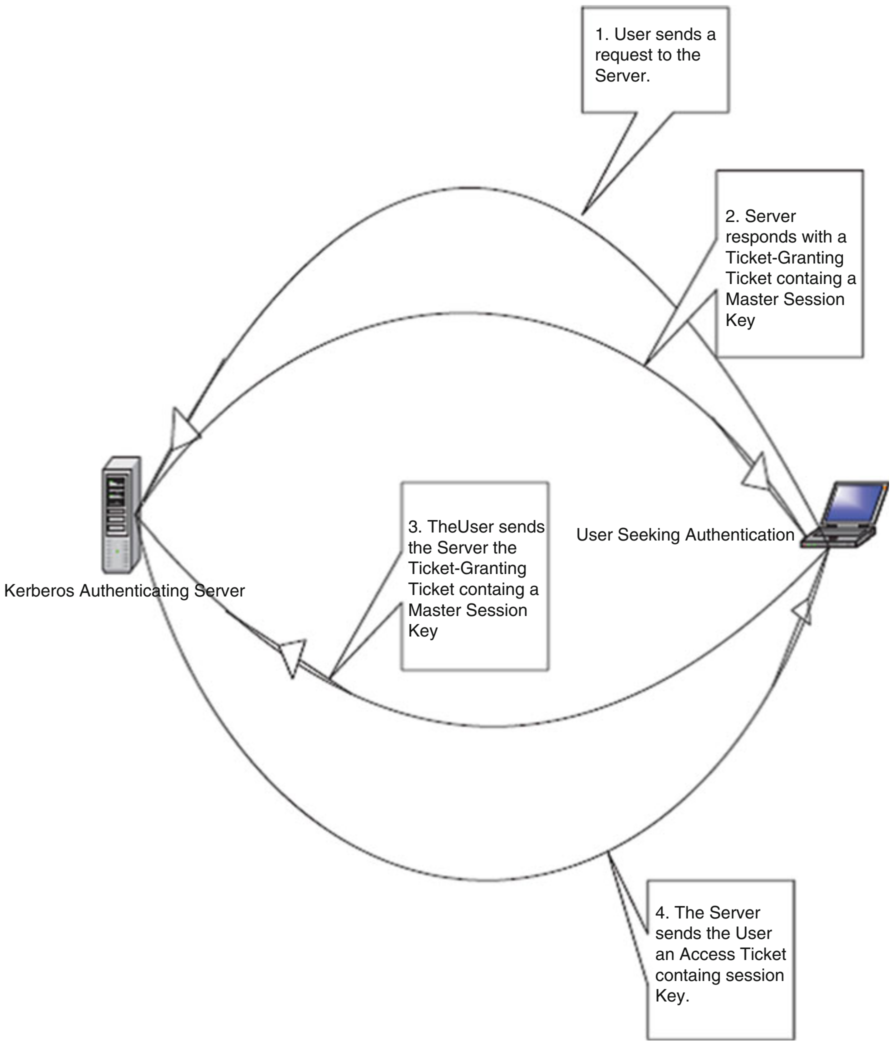
1. User sends a request to the Server.
2. Server responds with a Ticket-Granting Ticket containg a Master Session Key
3. TheUser sends the Server the Ticket-Granting Ticket containg a Master Session Key
User Seeking Authentication
Kerberos Authenticating Server
4. The Server sends the User an Access Ticket containg session Key.

Fig. 17.3 Kerberos authentication system

17.2.8.1 Ticket-Granting Ticket

The Kerberos system may have more than one AS. While this method works well, it is not secure. This is so because the ticket stays in use for some time and is, therefore, susceptible to hacking. To tackle this problem, Kerberos systems use another approach that is more secure. This second approach uses the first approach as the basis. An authentication process goes as follows:

A, in need of accessing a document on server B, sends an access request text in the clear to AS for a *ticket-granting ticket*, which is a master ticket for the log-in process with B. AS, using a shared secret such as a password, verifies A and sends A a ticket-granting ticket. A then uses this ticket-granting ticket instead of A's public key to send to AS for a ticket for any resource from any server A wants. To send the ticket to A, the AS encrypts it with a master session key contained in the ticket-granting ticket.

When a company has distributed its computing by having Kerberos networks in two separate areas, there are ways to handle situations like this. One Kerberos system in one network can authenticate a user in another Kerberos network. Each Kerberos AS operates in a defined area of the network called a *realm.* To extend the Kerberos' authentication across realms, an inter-realm key is needed that allows clients authenticated by an AS in one realm to use that authentication in another realm. Realms sharing inter-realm keys can communicate.

Kerberos can run on top of an operating system. In fact it is a default protocol in Windows 2000 and later versions of Windows. In a nonnative environment, Kerberos must be "kerberized" by making server and client software make calls to Kerberos library.

17.3 Security in the Transport Layer

Unlike the last five protocols we have been discussing in the previous section, in this section we look at protocols that are a little deeper in the network infrastructure. They are at the level below the application layer. In fact they are at the transport layer. Although several protocols are found in this layer, we are only going to discuss two: Secure Socket Layer (SSL) and Transport Layer Security (TLS). Currently, however, these two are no longer considered as two separate protocols but one under the name SSL/TLS, after the SSL standardization was passed over to IETF, by the Netscape consortium, and Internet Engineering Task Force (IETF) renamed it TLS. Figure 17.3 shows the position of these protocols in the network protocol stack.

17.3.1 Secure Socket Layer (SSL)

SSL is a widely used general-purpose cryptographic system used in the two major Internet browsers: Netscape and Explorer. It was designed to provide an encrypted end-to-end data path between a client and a server regardless of platform or OS. Secure and authenticated services are provided through data encryption, server authentication, message integrity, and client authentication for a TCP connection through HTTP, LDAP, or POP3 application layers. It was originally developed by Netscape Communications, and it first appeared in a Netscape Navigator browser in 1994. The year 1994 was an interesting year for Internet security because during the same year, a rival security scheme to SSL, the S-HTTP, was launched. Both systems were designed for Web-based commerce. Both allow for the exchange of multiple messages between two processes and use similar cryptographic schemes such as digital envelopes, signed certificates, and message digest.

Fig. 17.4 Transport layer security protocols and standards

TCP/IP	ISO	Security Protocols
Transport	Session Transport	SSL and TLS

Although these two Web giants had much in common, there are some differences in design goals, implementation, and acceptance. First, S-HTTP was designed to work with only Web protocols. Because SSL is at a lower level in the network stack than S-HTTP, it can work in many other network protocols. Second, in terms of implementation, since SSL is again at a lower level than S-HTTP, it is implemented as a replacement for the sockets API to be used by applications requiring secure communications. On the other hand, S-HTTP has its data passed in named text fields in the HTTP header. Finally, in terms of distribution and acceptance, history has not been so good to S-HTTP. While SSL was released in a free mass circulating browser, the Netscape Navigator, S-HTTP was released in a much smaller and restricted NCSA Mosaic. This unfortunate choice doomed the fortunes of S-HTTP (Fig. 17.4).

17.3.1.1 SSL Objectives and Architecture

The stated SSL objectives were to secure and authenticate data paths between servers and clients. These objectives were to be achieved through several services that included data encryption, server and client authentication, and message integrity [8]:

- Data encryption—to protect data in transport between the client and the server from interception and could be read only by the intended recipient.
- Server and client authentication—the SSL protocol uses standard public key encryption to authenticate the communicating parties to each other.
- Message integrity—achieved through the use of session keys so that data cannot be either intentionally or unintentionally tampered with.

These services offer reliable end-to-end secure services to Internet TCP connections and are based on an SSL architecture consisting of two layers: the top layer, just below the application layer, that consists of three protocols, namely, the SSL Handshake protocol, the SSL Change Cipher Spec protocol, and the SSL Alert protocol. Below these protocols is the second SSL layer, the SSL Record Protocol layer, just above the TCP layer. See Fig. 17.5.

17.3.1.2 The SSL Handshake

Before any TCP connection between a client and a server, both running under SSL, is established, there must be almost a process similar to a three-way handshake we

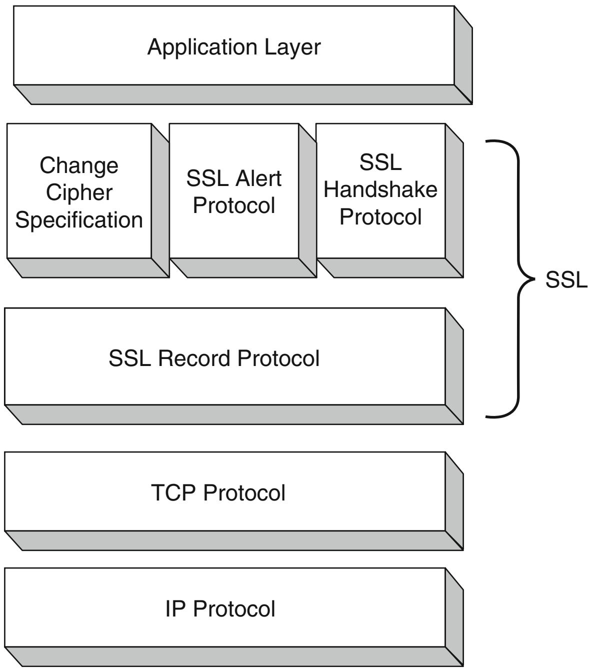

Fig. 17.5 The SSL protocol stack

discussed in Sect. 3.2.2. This get-to-know-you process is similarly called the SSL handshake. During the handshake, the client and server perform the following tasks [9]:

- Establish a cipher suite to use between them.
- Provide mandatory server authentication through the server sending its certificate to the client to verify that the server's certificate was signed by a trusted CA.
- Provide optional client authentication, if required, through the client sending its own certificate to the server to verify that the client's certificate was signed by a trusted CA. The CA may not be the same CA who signed the client's certificate. CAs may come from a list of trusted CAs. The reason for making this step optional was a result of realization that since few customers are willing, know how, or care to get digital certificates, requiring them to do this would amount to locking a huge number of customers out of the system which would not make business sense. This, however, presents some weaknesses to the system.
- Exchange key information using public key cryptography, after mutual authentication, that leads to the client generating a session key (usually a random number) which, with the negotiated cipher, is used in all subsequent encryption or decryption. The customer encrypts the session key using the public key of the merchant server (from the merchant's certificate). The server recovers the session key by decrypting it using its private key. This symmetric key, which now both parties have, is used in all subsequent communication.

17.3.1.3 SSL Cipher Spec Protocol

The SSL Cipher Spec protocol consists of an exchange of a single message in a byte with a value of 1 being exchanged, using the SSL record protocol (see Sect. 17.3.1.4), between the server and client. The bit is exchanged to establish a pending session state to be copied into the current state, thus defining a new set of protocols as the new agreed on session state.

17.3.1.4 SSL Alert Protocol

The SSL Alert protocol, which also runs over the SSL Record protocol, is used by the two parties to convey session warning messages associated with data exchange and functioning of the protocol. The warnings are used to indicate session problems ranging from unknown certificate, revoked certificate, and expired certificate to fatal error messages that can cause immediate termination of the SSL connection. Each message in the alert protocol sits within two bytes, with the first byte taking a value of (1) for a warning and (2) for a fatal error. The second byte of the message contains one of the defined error codes that may occur during an SSL communication session [8]. For further working of these error codes, see [8].

17.3.1.5 SSL Record Protocol

The SSL record protocol provides SSL connections two services: confidentiality and message integrity [5]:

- *Confidentiality* is attained when the handshake protocol provides a shared secret key used in the conventional encryption of SSL messages.
- *Message integrity* is attained when the handshake defines a secret shared key used to form a message authentication code (MAC).

In providing these services, the SSL Record Protocol takes an application message to be transmitted and fragments the data that needs to be sent, compresses it, adds a MAC, encrypts it together with the MAC, adds an SSL header, and transmits it under the TCP protocol. The return trip undoes these steps. The received data is decrypted, verified, and decompressed before it is forwarded to higher layers. The record header that is added to each data portion contains two elementary pieces of information, namely, the length of the record and the length of the data block added to the original data. See Fig. 17.6.

The MAC, computed from a hash algorithm such as MD5 or SHA-1 as MAC = Hash function [secret key, primary data, padding, sequence number], is used to verify the integrity of the message included in the transmitted record. The verification is done by the receiving party computing its own value of the MAC and comparing it with that received. If the two values match, this means that data has not been modified during the transmission over the network.

SSL protocols are widely used in all Web applications and any other TCP connections. Although they are mostly used for Web applications, they are gaining ground in e-mail applications also.

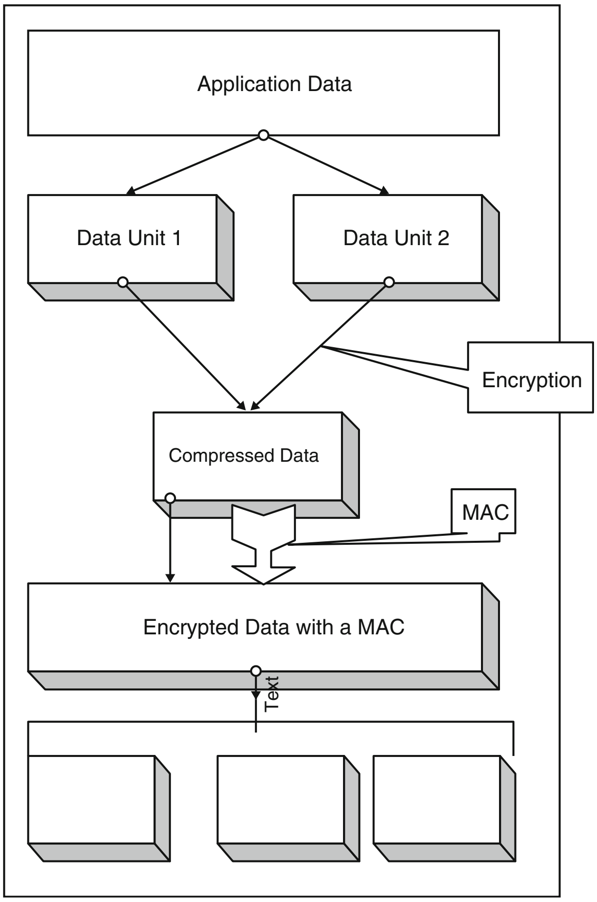

Fig. 17.6 SSL record protocol operation process

17.3.2 Transport Layer Security (TLS)

Transport Layer Security (TLS) is the result of the 1996 Internet Engineering Task Force's (IETF) attempt at standardization of a secure method to communicate over the Web. The 1999 outcome of that attempt was released as RFC 2246 spelling out a new protocol—the Transport Layer Security or TLS. TLS was charged with providing security and data integrity at the transport layer between two applications. TLS version 1.0 was an evolved SSL 3.0. So, as we pointed out earlier, TLS is the successor to SSL 3.0. Frequently, the new standard is referred to as SSL/TLS.

Since then, however, the following additional features have been added [8]:

- *Interoperability*—ability to exchange TLS parameters by either party, with no need for one party to know the other's TLS implementation details

Fig. 17.7 Network layer security protocols and standards

TCP/IP	ISO	Security Protocols
Network	Network	IPsec, VPN, PPTP and L2TP

- *Expandability*—to plan for future expansions and accommodation of new protocols

17.4 Security in the Network Layer

In the previous section, we discussed protocols in the transport part of the stack that are being used to address Internet communication security. In this section, we are going one layer down, to the network layer, and also look at the protocols and probably standards that address Internet communication security. In this layer, we will address IPsec and VPN technologies shown in Fig. 17.7.

17.4.1 Internet Protocol Security (IPsec)

IPsec is a suite of authentication and encryption protocols developed by the Internet Engineering Task Force (IETF) and designed to address the inherent lack of security for IP-based networks. IPsec, unlike other protocols we have discussed so far, is a very complex set of protocols described in a number of RFCs including RFC 2401 and 2411. It runs transparently to transport layer and application layer protocols which do not see it. Although it was designed to run in the new version of the Internet Protocol, IP version 6 (IPv6), it has also successfully run in the older IPv4. IPsec sets out to offer protection by providing the following services at the network layer:

- Access control—to prevent an unauthorized access to the resource.
- Connectionless integrity—to give an assurance that the traffic received has not been modified in any way.
- Confidentiality—to ensure that Internet traffic is not examined by nonauthorized parties. This requires all IP datagrams to have their data field, TCP, UDP, ICMP, or any other datagram data field segment encrypted.
- Authentication—particularly source authentication so that when a destination host receives an IP datagram, with a particular IP source address, it is possible to be sure that the IP datagram was indeed generated by the host with the source IP address. This prevents spoofed IP addresses.
- Replay protection—to guarantee that each packet exchanged between two parties is different.

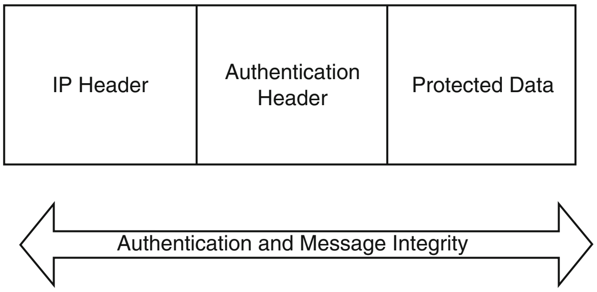

Fig. 17.8 IPsec's AH protocol protection

IPsec protocol achieves these two objectives by dividing the protocol suite into two main protocols: Authentication Header (AH) protocol and the Encapsulation Security Payload (ESP) protocol [10]. The AH protocol provides source authentication and data integrity but no confidentiality. The ESP protocol provides authentication, data integrity, and confidentiality. Any datagram from a source must be secured with either AH or ESP. Figures 17.8 and 17.9 show both IPsec's ESP and AH protections.

17.4.1.1 Authentication Header (AH)

AH protocol provides source authentication and data integrity but not confidentiality. This is done by a source that wants to send a datagram first establishing an SA, through which the source can send the datagram. A source datagram includes an AH inserted between the original IP datagram data and the IP header to shield the data field which is now encapsulated as a standard IP datagram. See Fig. 17.5. Upon receipt of the IP datagram, the destination host notices the AH and processes it using the AH protocol. Intermediate hosts such as routers, however, do their usual job of examining every datagram for the destination IP address and then forwarding it on.

17.4.1.2 Encapsulating Security Payload (ESP)

Unlike the AH protocol, ESP protocol provides source authentication, data integrity, and confidentiality. This has made ESP the most commonly used IPsec header. Similar to AH, ESP begins with the source host establishing an AS which it uses to send secure datagrams to the destination. Datagrams are secured by ESP by surrounding their original IP datagrams with a new header and trailer fields all encapsulated into a new IP datagram. See Fig. 17.6. Confidentiality is provided by DES_CBC encryption. Next to the ESP trailer field on the datagram is the ESP Authentication Data field.

17.4.1.3 Security Associations

In order to perform the security services that IPsec provides, IPsec must first get as much information as possible on the security arrangement of the two communicating hosts. Such security arrangements are called *security associations* (SAs). A security association is a unidirectional security arrangement defining a set of items

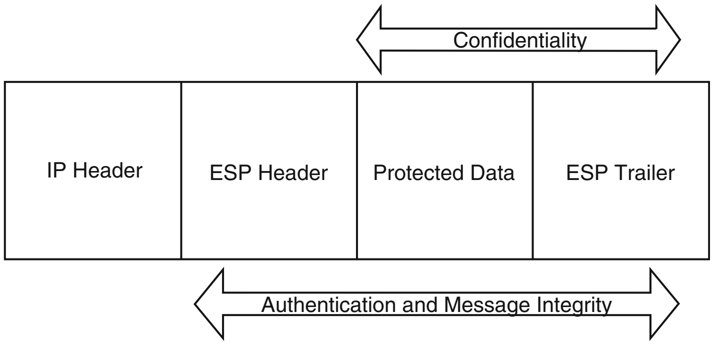

Fig. 17.9 IPsec's ESP protocol protection

and procedures that must be shared between the two communicating entities in order to protect the communication process.

Recall from Chap. 1 that in the usual network IP connections, the network layer IP is connectionless. However, with security associations, IPsec creates logical connection-oriented channels at the network layer. This logical connection-oriented channel is created by a security agreement established between the two hosts stating specific algorithms to be used by the sending party to ensure confidentiality (with ESP), authentication, message integrity, and anti-replay protection.

Since each AS establishes a unidirectional channel, for a full-duplex communication between two parties, two SAs must be established. An SA is defined by the following parameters [5, 11]:

- Security Parameter Index (SPI)—a 32-bit connection identifier of the SA. For each association between a source and destination host, there is one SPI that is used by all datagrams in the connection to provide information to the receiving device on how to process the incoming traffic.
- IP destination address—address of a destination host.
- A security protocol (AH or ESP)—to be used and specifying if traffic is to be provided with integrity and secrecy. The protocol also defines the key size, the key lifetime, and the cryptographic algorithms.
- Secret key—which defines the keys to be used.
- Encapsulation mode—defining how encapsulation headers are created and which parts of the header and user traffic are protected during the communication process.

Figure 17.10 shows the general concept of a security association.

17.4.1.4 Transport and Tunnel Modes

The security associations discussed above are implemented in two modes: transport and tunnel. This means that IPsec is operating in two modes. Let us look at these [5].

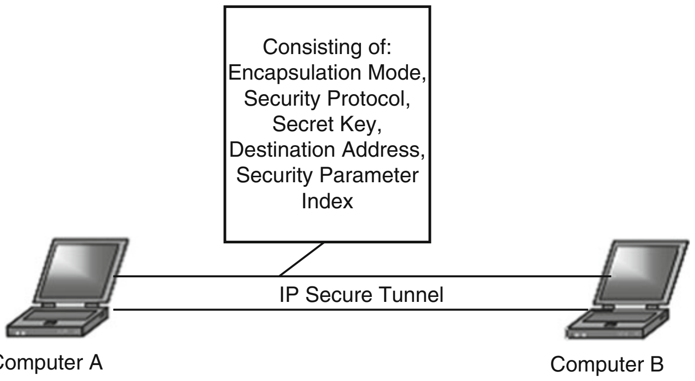

Fig. 17.10 A general concept of IPsec's security association

Transport Mode

Transport mode provides host-to-host protection to higher-layer protocols in the communication between two hosts in both IPv4 and IPv6. In IPv4, this area is the area beyond the IP address as shown in Fig. 17.11. In IPv6, the new extension to IPv4, the protection includes the upper protocols, the IP address, and any IPv6 header extensions as shown in Fig. 17.8. The IP addresses of the two IPsec hosts are in the clear because they are needed in routing the datagram through the network.

Tunnel Mode

Tunnel mode offers protection to the entire IP datagram both in AH and ESP between two IPsec gateways. This is possible because of the added new IP header in both IPv4 and IPv6 as shown in Fig. 17.12. Between the two gateways, the datagram is secure and the original IP address is also secure. However, beyond the gateways, the datagram may not be secure. Such protection is created when the first IPsec gateway encapsulates the datagram including its IP address into a new shield datagram with a new IP address of the receiving IPsec gateway. At the receiving gateway, the new datagram is unwrapped and brought back to the original datagram. This datagram, based on its original IP address, can be passed on further by the receiving gateway, but from this point on unprotected.

17.4.1.5 Other IPsec Issues

Any IPsec compliant system must support single DES, MD5, and SHA-1 as an absolute minimum; this ensures that a basic level of interworking is possible with two IPsec compliant units at each end of the link. Since IPsec sits between the network and transport layers, the best place for its implementation is mainly in hardware.

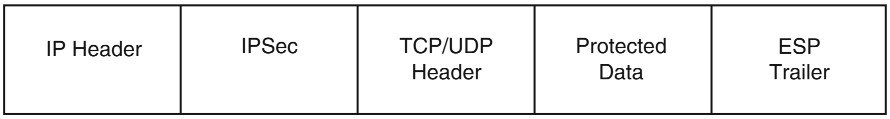

Fig. 17.11 IPsec's transport mode

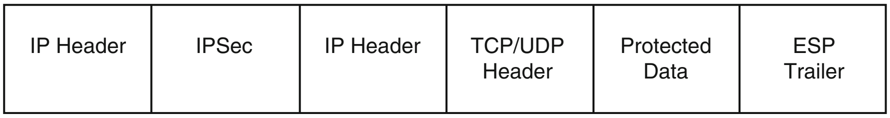

Fig. 17.12 IPsec's tunnel mode

17.4.2 Virtual Private Networks (VPN)

A VPN is a private data network that makes use of the public telecommunication infrastructure, such as the Internet, by adding security procedures over the unsecure communication channels. The security procedures that involve encryption are achieved through the use of a tunneling protocol. There are two types of VPNs: remote access which lets single users connect to the protected company network and site-to-site which supports connections between two protected company networks. In either mode, VPN technology gives a company the facilities of expensive private leased lines at much lower cost by using the shared public infrastructure like the Internet. See Fig. 17.13.

Figure 17.8 shows two components of a VPN [3]:

- Two terminators which are either software or hardware. These perform encryption, decryption, and authentication services. They also encapsulate the information.
- A tunnel—connecting the endpoints. The tunnel is a secure communication link between the endpoints and networks such as the Internet. In fact this tunnel is virtually created by the endpoints.

VPN technology must do the following activities:

- IP encapsulation—this involves enclosing TCP/IP data packets within another packet with an IP address of either a firewall or a server that acts as a VPN endpoint. This encapsulation of host IP address helps in hiding the host.
- Encryption—is done on the data part of the packet. Just like in SSL, the encryption can be done either in transport mode which encrypts its data at the time of generation or tunnel mode which encrypts and decrypts data during transmission encrypting both data and header.
- Authentication—involves creating an encryption domain which includes authenticating computers and data packets by use for public encryption.

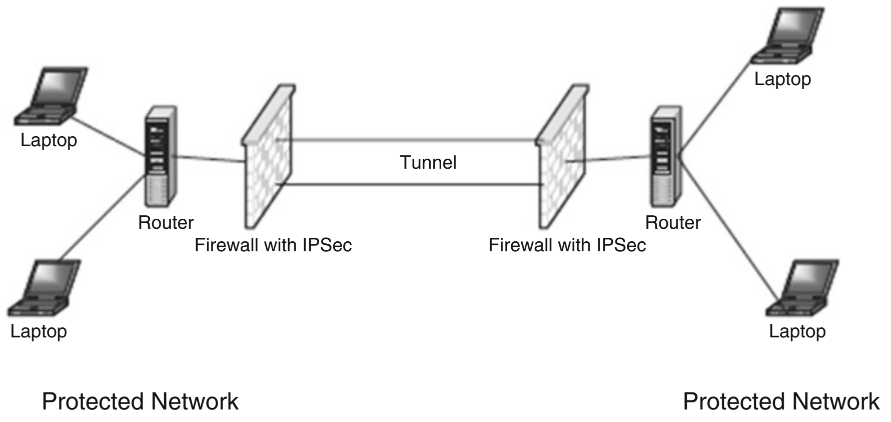

Fig. 17.13 Virtual private network (VPN) model

VPN technology is not new; phone companies have provided private shared resources for voice messages for over a decade. However, its extension to making it possible to have the same protected sharing of public resources for data is new. Today, VPNs are being used for both extranets and wide-area intranets. Probably owing to cost savings, the popularity of VPNs by companies has been phenomenal.

17.4.2.1 Types of VPNs

The security of VPN technologies falls into three types: trusted VPNs, secure VPNs, and hybrid VPNs.

Trusted VPNs

As we have noted above, before the Internet, VPN technology consisted of one or more circuits leased from a communications provider. Each leased circuit acted like a single wire in a network that was controlled by a customer who could use these leased circuits in the same way that he or she used physical cables in his or her local network. So this legacy VPN provided customer privacy to the extent that the communications provider assured the customer that no one else would use the same circuit. Although leased circuits ran through one or more communications switches, making them susceptible to security compromises, a customer trusted the VPN provider to safeguard his or her privacy and security by maintaining the integrity of the circuits. This security based on trust resulted into what is now called *trusted VPNs*.

Trusted VPN technology comes in many types. The most common of these types are what is referred to as *layer 2* and *layer 3* VPNs. Layer 2 VPNs include ATM circuits, frame relay circuits, and transport of layer 2 frames over Multiprotocol Layer Switching (MPLS). Layer 3 VPNs include MPLS with constrained distribution of routing information through Border Gateway Protocol (BGP) [11].

Because the security of trusted VPNs depends only on the goodwill of the provider, the provider must go an extra mile to assure the customers of the security responsibility requirements they must expect. Among these security requirements are the following [11]:

- *No one other than the trusted VPN provider can affect the creation or modification of a path in the VPN.* Since the whole trust and value of trusted VPN security rides on the sincerity of the provider, no one other than the provider can change any part of the VPN.
- *No one other than the trusted VPN provider can change data, inject data, or delete data on a path in the VPN.* To enhance the trust of the customer, a trusted VPN should secure not only a path but also the data that flows along that path. Since this path can be one of the shared paths by the customers of the provider, each customer's path itself must be specific to the VPN and no one other than the trusted provider can affect the data on that path.
- *The routing and addressing used in a trusted VPN must be established before the VPN is created.* The customer must know what is expected of the customer and what is expected of the service provider so that they can plan for maintaining the network that they are purchasing.

Secure VPNs

Since the Internet is a popular public communication medium for almost everything from private communication to businesses and the trusted VPN actually offers only virtual security, security concerns in VPN have become urgent. To address these concerns, vendors have started creating protocols that would allow traffic to be encrypted at the edge of one network or at the originating computer, moved over the Internet like any other data, and then decrypted when it reaches the corporate network or a receiving computer. This way it looks like encrypted traffic has traveled through a tunnel between the two networks. Between the source and the destination points, although the data is in the clear and even an attacker can see the traffic, still one cannot read it and one cannot change the traffic without the changes being seen by the receiving party and therefore rejected. Networks that are constructed using encryption are called *secure VPNs.*

Although secure VPNs are more secure than trusted VPNs, they too require assurance to the customer just like trusted VPNs. These requirements are as follows [11]:

- *All traffic on the secure VPN must be encrypted and authenticated.* In order for VPNs to be secure, there must be authentication and encryption. The data is encrypted at the sending network and decrypted at the receiving network.
- *The security properties of the VPN must be agreed to by all parties in the VPN.* Every tunnel in a secure VPN connects two endpoints who must agree on the security properties before the start of data transmission.
- *No one outside the VPN can affect the security properties of the VPN.* To make it difficult for an attacker, VPN security properties must not be changed by anyone outside the VPN.

Hybrid VPNs

Hybrid VPN is the newest type of VPN technologies that substitutes the Internet for the telephone system as the underlying structure for communications. The trusted VPN components of the new VPN still do not offer security, but they give customers a way to easily create network segments for wide area networks (WANs). On the other hand, the secure VPN components can be controlled from a single place and often come with guaranteed quality of service (QoS) from the provider.

Because of the inherited weaknesses from both components that make up this new hybrid VPN, a number of security requirements must be adhered to. Among the requirements is to clearly mark the address boundaries of the secure VPN within the trusted VPN because in hybrid VPNs, the secure VPN segments can run as subsets of the trusted VPN and vice versa. Under these circumstances, the hybrid VPN is secure only in the parts that are based on secure VPNs.

17.4.3 VPN Tunneling Technology, PPTP, and L2TP

Old VPN firewalls used to come loaded with software that constructed the tunnel. However, with new developments in tunneling technology, this is no longer the case. Let us now look at some different technologies that can be used to construct VPN tunnels:

- *IPsec with encryption* used in either tunnel or transport modes. Since IPsec cannot authenticate users, IPsec alone is not good in some host-to-network VPN [3]. However, this is usually overcome by supplementing IPsec with other authentication methods such as Kerberos. In combination with Internet Key Exchange (IKE) which provides a trusted public key exchange, IPsec is used to encrypt data between networks and between hosts and networks. According to Hoden, the sequence of events in the process of establishing an IPsec/IKE VPN connection goes as follows:
  - The host/gateway at one end of a VPN sends a request to the host/gateway at the other end to establish a VPN connection.
  - The remote host/gateway generates a random number and sends a copy of it to the requesting host/gateway.
  - The requesting host/gateway, using this random number, encrypts its pre-shared key it got from the IKE (shared with the remote host/gateway) and sends it back to the remote host/gateway.
  - The remote host/gateway also uses its random number and decrypts its pre-shared key and compares the two keys for a match. If there is a match with anyone of its keys on the key ring, then it decrypts the public key using this pre-shared key and sends the public key to the requesting host/gateway.
  - Finally, the requesting host/gateway uses the public key to establish the IPsec security association (SA) between the remote host/gateway and itself. This exchange establishes the VPN connection. See Fig. 17.14.

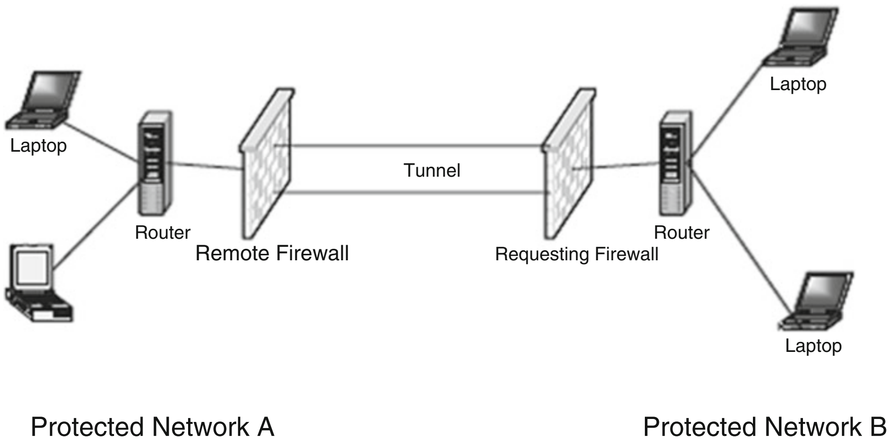

Fig. 17.14 Establishing a VPN using IPsec and IKE

- *Point-to-Point Tunneling Protocol (PPTP)*. This is a Microsoft-based dial-up protocol used by remote users seeking a VPN connection with a network. It is an older technology with limited use.
- *Layer 2 Tunneling Protocol* **[L2TP inside IPsec** (*see RFC 3193*)**].** This is an extension of PPP, a dial-up technology. Unlike PPTP which uses Microsoft dial-up encryption, L2TP uses IPsec in providing secure authentication of remote access. L2TP protocol makes a host connect to a modem, and then it makes a PPP to the data packets to a host/gateway where it is unpacked and forwarded to the intended host.
- PPP over SSL and PPP over SSH. These are Unix-based protocols for constructing VPNs. Please note that PPP also tunnels Internet Protocol (IP) or other network layer 3 data between two directly connected nodes over a physical connection or over a direct link. Since IP and Transmission Control Protocol (TCP) do not support point-to-point connections, the use of PPP can enable them over Ethernet and other physical media (Fig. 17.15).

17.5 Security in the Physical Layer

17.5.1 Point-to-Point Protocol (PPP)

This is an old protocol because early Internet users used to dial into the Internet using a modem and PPP. It is a protocol limited to a single data link. Each call-in went directly to the remote access service (RAS) whose job was to authenticate the calls as they came in.

Fig. 17.15 Physical and data link-layer security protocols and standards

TCP/IP	ISO	Security Protocols
Physical	Data link Physical	PPP, Packet Filters, NAT, CHAP and PAP

A PPP communication begins with a handshake which involves a negotiation between the client and the RAS to settle the transmission and security issues before the transfer of data could begin. This negotiation is done using the Link Control Protocol (LCP). Since PPP does not require authentication, the negotiation may result in an agreement to authenticate or not to authenticate

17.5.1.1 PPP Authentication

If authentication is the choice, then several authentication protocols can be used. Among these are Password Authentication Protocol (PAP), Challenge-Handshake Authentication Protocol (CHAP), and Extensible Authentication Protocol (EAP) [12]:

- *Password Authentication Protocol (PAP)* requires the applicant to repeatedly send to the server authentication request messages, consisting of a username and password, until a response is received or the link is terminated. However, this information is sent in the clear.
- *Challenge-Handshake Authentication Protocol (CHAP)* works on a "shared secret" basis where the server sends to the client a challenge message and waits for a response from the client. Upon receipt of the challenge, the client adds on a secret message, hashes both, and sends the result back to the server. The server also adds a secret message to the challenge, hashes with an agreed-upon algorithm, and then compares the results. Authentication of the client results if there is a match. To harden the authentication process, the server periodically authenticates the client.
- *Extensible Authentication Protocol (EAP)* is open-ended, allowing users to select from among a list of authentication options.

17.5.1.2 PPP Confidentiality

During the negotiations, the client and server must agree on the encryption that must be used. IETF has recommended two such encryptions that include DES and 3DES.

17.5.2 Other Network Physical Layer Security Protocols Include [13]

(a) *Packet Filters*—A packet filter is designed to sit between the internal and external network. As packets enter or leave the network, they are compared to a set of rules. This determines if they are passed, rejected, or dropped. A router ACL is an example of a packet filter.

(b) *NAT* (network address translation) is a means of translating addresses. Most residential high-speed Internet users use NAT. It provides security as it hides the internal address from external networks.

(c) *CHAP* (Challenge-Handshake Authentication Protocol) is an authentication protocol that is used as an alternative to passing clear-text usernames and passwords. CHAP uses the MD5 hashing algorithm to encrypt passwords.

(d) *PAP* (Password Authentication Protocol)—this may not be the best security mechanism at the physical layer; however, it does provide some protection as it requires a user to present a username and password. Its Achilles heel is that it transmits this information in clear text.

Exercises

1. PGP has been a very successful secure communication protocol. Why is this so? What features brought it that success?
2. Discuss five benefits of IPsec as a security protocol.
3. Discuss the differences between the transport mode and the tunnel mode of IPsec. Is one mode better than the other? Under what conditions would you use one over the other?
4. Study the IPv4 and IPv6 and discuss the differences between them? By the mid-1990s, it was thought that IPv6 was destined to be a replacement of IPv4 in less than 5 years. What happened? Is there a future for IPv6?
5. Discuss the differences between RADIUS, as a remote authentication protocol, and Kerberos when using realms.
6. What are Kerberos authentication path? How do they solve the problem of remote authentication?
7. The Kerberos system has several bugs that pose potential security risks. Study the Kerberos ticketing service and discuss these bugs.
8. The Kerberos system is built on the principle that only a limited number of machines on any network can possibly be secure. Discuss the validity of this statement.
9. Investigate how Netscape Navigator and Internet Explorer implemented SSL technology.
10. Study both SSL and S-HTTP. Show that these two protocols have a lot in common. However, the two protocols have some differences. Discuss these differences. Discuss what caused the decline in the use of S-HTTP.

Advanced Exercises

1. X.509 is a good security protocol. Study X.509 and discuss how it differs from S-HTTP and IPsec.
2. SSL3.0 has been transformed into TLS 1.0. Study the TLS protocol specifications and show how all are met by SSL. There are some differences in the protocol specifications of the two. Describe these differences.
3. S/MIME and PGP are sister protocols; they share a lot. Study the two protocols and discuss the qualities they share. Also look at the differences between them. Why is PGP more successful? Or is it?
4. Study the SET protocols and one other payment system protocol such as Dig-Cash. What sets SET above the others? Is SET hacker proof? What problems does SET face that may prevent its becoming a standard as desired by Netscape?
5. Both S/MIME and PGP are on track for standardization by the IETF. In your judgment, which one of the twos likely to become the standard and why?

References

1. *Radius*. http://www.gnu.org/software/radius/radius.html#TOCintroduction
2. *RFC 2138*. http://www.faqs.org/rfcs/rfc2138.html
3. Hoden G (2004) Guide to firewalls and network security: intrision detection and VPNs. Thomson Delmar Learning, Clifton Park
4. Stallings W (1999) Cryptography and network security: principles and practice, 2nd edn. Prentice Hall, Upper Saddle River
5. Shastack A. An overview of S-HTTP. http://www.homeport.org/~adam/shttp.html
6. Jasma K (2002) Hacker proof: the ultimate guide to network security, 2nd edn. OnWord Press, Albany
7. Tung B. The Moron’s Guide to Kerberos, Version 2.0. https://wpollock.com/AUnixSec/MoronsGuideToKerberos.htm
8. Onyszko T. Secure socket layer: authentication, access control and encryption. http://www.windowsecurity.com/articles-tutorials/authentication_and_encryption/Secure_Socket_Layer.html
9. *SSL Demystified: The SSL encryption method in IIS*. http://windowsitpro.com/security/ssl-demystified
10. Kurose J, Keith WR (2003) Computer networking: a top-down approach featuring the internet. Addison-Wesley, Reading.
11. VPN Consortium (2003) VPN technologies: definitions are requirements. http://www.hit.bme.hu/~jakab/edu/litr/VPN/vpn-technologies.pdf
12. Panko R (2004) Corporate computer and network security. Prentice Hall, Upper Saddle River
13. TechTarget. Security and the TCP/IP stack. http://searchnetworking.techtarget.com/tip/Security-and-the-TCP-IP-stack