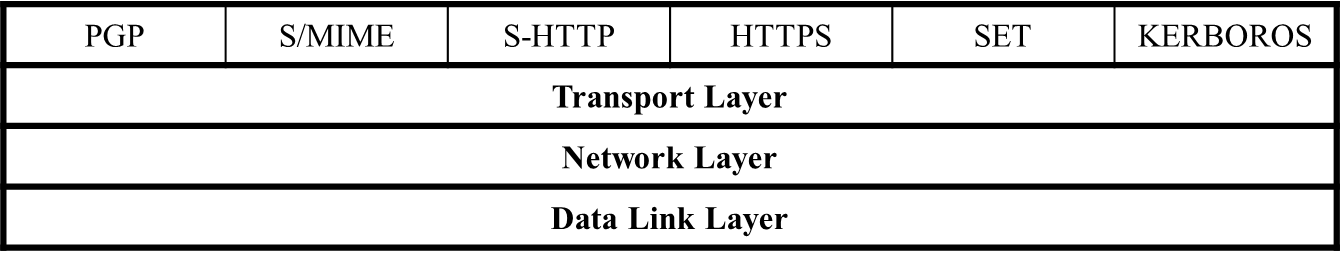
COMPARISON OF NETWORK SECURITY PROTOCOLS

First A. Author, *Fellow, IEEE*, Second B. Author, and Third C. Author, Jr., *Member, IEEE*

**Fig 1.** Application layer security protocols and standard



***Abstract*—** **Security is a continuous process of protecting an object from unauthorized access. It is a state of being or feeling protected from harm. That object in that state may be a person, an organization such as a business, or property such as a computer system or a file. The rapid growth of the Internet and corresponding Internet communities have 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.**

***Index Terms*— security, protocols, IPSEC, SSL, TLS**

# I. INTRODUCTION

S

ecurity and privacy are essential if individual communication is to continue and e-commerce is to thrive in cyberspace. The call and desire for security and privacy have 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. We will discuss these protocols and standards within the framework of the network protocol stack as follows:

* Application-level security – PGP, S/MIME, S-HTTP, HTTPS, SET, and KERBEROS
* Transport level security – SSL and TLS
* Network level security – IPSec and VPNs
* Link level security – PPP and RADIUS

# II. Application-Level Security

In the last few 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 look at four of them here, all in the application layer of the network stack. There are PGP and Secure/Multipurpose Internet Mail Extension (S/MIME), S-HTTP, HTTPS, and Secure Electronic Transaction (SET) standard. These four protocols and the standards are shown in the application layer of the network stack in Fig 1.

## 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 [1].

1. *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 [2]. 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 [1].

1. **Encryption:** S/MIME supports three public-key algorithms to encrypt session keys for transmission with the message. These include Diffie–Hellman as the preferred algorithm, RSA for both signature and session keys, and triple DES.
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.
3. *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 protocols 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 protocols. S-HHTP extended HTTP protocols 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 [3].

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 protocol, 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 protocol 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 1 and 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 negotiates for information exchange. All headers in S-HTTP are optional, except “Content Type” and “Content-Privacy-Domain.”

TABLE I

S-HTTP protocol 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 bits |
| **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 II

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 |  |

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 discover a common set of algorithms, appropriate actions are then taken. Adam Shostack [2] gives the following example as a negotiation line:

SHTTP-Key-Exchange-Algorithms: recv-required = RSA, Kerb-5

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 lengths specifications are necessary in case of variable key length ciphers, this is then specifically referred to as cipher[length], or cipher[L1-L2], where the length of key is length, or in the case of L1-L2, is between L1 and L2, inclusive [2]. Other headers in the S-HTTP negotiations could be [2], [3]:

* + 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

1. *Hypertext Transfer Protocol over Secure Socket Layer (HTTPS)*

HTTPS is the use of Secure Socket Layer (SSL) as a sublayer 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.

1. *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 [1]:

* + Confidentiality of payment and ordering information
  + Integrity of all transmitted data
  + Authentication of all card holders
  + 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 [1], [4].

1. *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 [5], 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 [2]:

* + 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.

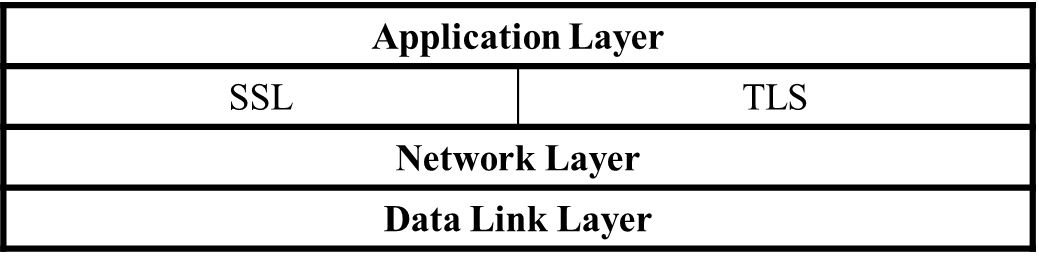
To meet these requirements, Kerberos designers proposed a third-party-trusted authentication service to arbitrate between the client and server in their mutual authentication.

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 drivers 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.

# III. 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. Fig 2 shows the position of these protocols in the network protocol stack.

**Fig 2.** Transport layer security protocols and standards



1. *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 [6].

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.

1. *Transport Layer Security (TLS)*

Transport Layer Security (TLS) is the result of the 1996 Internet Engineering Task Force (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.

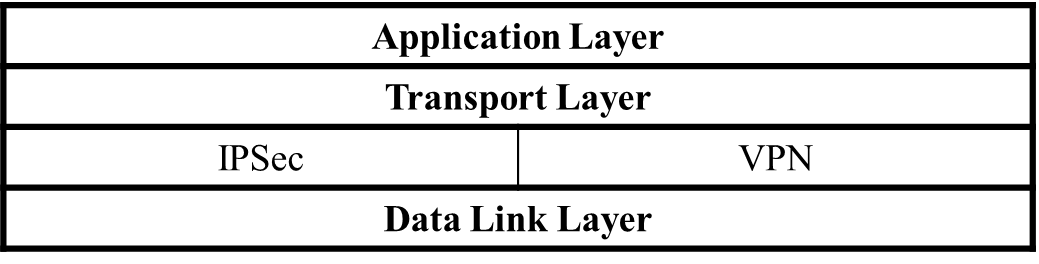
Since then, however, the following additional features have been added [7]:

* + Interoperability – ability to exchange TLS parameters by either party, with no need for one party to know the other’s TLS implementation details
  + Expandability – to plan for future expansions and accommodation of new protocols

# IV. 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 3.

**Fig 3.** Network layer security protocols and standards

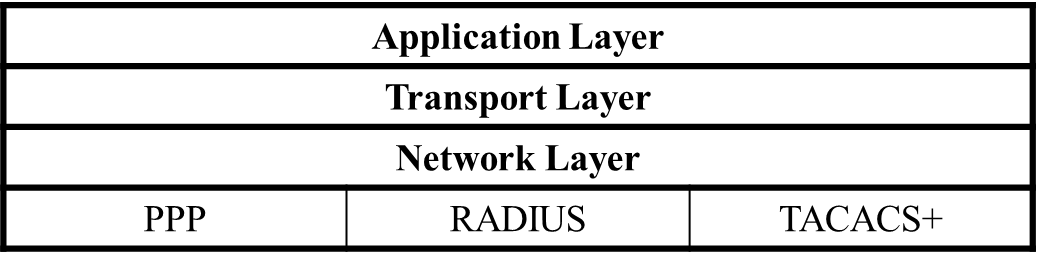


1. *Internet Protocol Security (IPSec)*

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 as well. 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.

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 [8]. 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.



**Fig 4.** Data link layer security protocols and standards

1. *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. Two components of a VPN are [9]:

* + 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.

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.

# V. Security in the Link Layer and over LANS

Finally, our progressive survey of security protocols and standards in the network security stack ends with a look at the security protocols and standards in the data link layer. In this layer, although there are several protocols including those applied in the LAN technology, we will look at only two: PPP and RADIUS. Fig 4 shows the position of these protocols in the stack.

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.

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.

1. *Remote Authentication Dial-In User Service (RADIUS)*

RADIUS is a server for remote user authentication and accounting. 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 [10].

1. *Terminal Access Controller Access Control System (TACACS+)*

This protocol 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:

* + 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 TACASCS+, 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.

# VI. Conclusion

In this paper we have analyzed various network security protocols being used on different network layers. We have also discussed their utilization like IPSec is really transparent and the best way to use it is by implementing it in a hardware, SSL/TLS are only conceptually secure, but not really in practice, SSH, PGP are stand-alone applications which are immediately and easy to deploy and use. Network security is solved in principle but practically solving it still requires some research.

# **References**

|  |  |
| --- | --- |
| [1] | W. Stallings, "Cryptography and network security: principles and practice, 2nd edn," *Prentice Hall, Upper Saddle River,* 1999. |
| [2] | A. Shostack, "An Overview of SHTTP," vol. https://shostack.org/files/essays/shttp/, May 1995. |
| [3] | Jasma, Hacker proof: the ultimate guide to network security, 2nd edn, OnWord Press, Albany, 2002. |
| [4] | L. D. Stein, "Web security," *Addison-Wesley, Massachusetts,* vol. 26, pp. 1-4, 1998. |
| [5] | B. Tung, "The Moron's Guide to Kerberos, Version 2.0," 2007. [Online]. Available: https://wpollock.com/AUnixSec/MoronsGuideToKerberos.htm. |
| [6] | A. Jones, "SSL Demystified," 2000. [Online]. Available: https://www.itprotoday.com/security/ssl-demystified. |
| [7] | T. Onyszko, "Secure Socket Layer," 2002. [Online]. Available: https://techgenix.com/Secure\_Socket\_Layer/. |
| [8] | J. Kurose and K. Ross, Computer Networking: A Top-Down Approach, Addison-Wesley, 2003. |
| [9] | G. Holden, Guide to Firewalls and Network Security: Intrusion Detection and VPNs, Thomson Delmar Learning, Clifton Park, 2004. |
| [10] | "Radius," 2019. [Online]. Available: http://www.gnu.org/software/radius/radius.html#TOCintroduction. |