Internet Security Protocols

Walter Fumy

Siemens AG, Corporate Technology - Information and Communications
Munich, Germany
Walter.Fumy@mchp.siemens.de

Abstract. This article describes various efforts to address security in three areas of the Internet protocol suite: the Internet Protocol itself (IPsec), the domain between transport and application layer (the Secure Sockets Layer and the Transport Layer Security protocols) and security extensions for the HyperText Transfer Protocol (S-HTTP). For each area the current technology, relevant standardization activities and likely future developments are discussed. In addition, a brief introduction to the Internet standardization process is given.

1 Motivation

Any network connected to the Internet is at risk of intrusion. In 1995, Carnegie Mellon University's CERT (*Computer Emergency Response Team*) responded to more than 2,400 legitimate security breaches in the United States involving more than 12,000 Internet sites. The most common methods of deliberate intrusion identified by CERT are

- IP spoofing, whereby an intruder masquerades as a node whose Internet Protocol address is known and trusted, and
- packet sniffing, which invaders use to extract valid account names and passwords or credit card information from captured, unenciphered IP datagrams.

Part of the reason for the Internet security risks is the lack of effective security mechanisms in the current version of the Internet Protocol (IPv4). Other reasons are lax security practices in many network environments and some network administrators' lack of knowledge about Internet and Intranet security.

On the other hand, the ease of use of the World Wide Web has prompted widespread interest in its employment as a client/server architecture for many applications. As increasing amounts of sensitive information, such as credit card numbers, are being transmitted over the Internet, security issues have become of central importance.

Most companies with connections to the Internet have implemented firewall solutions to protect their corporate network from unauthorized users coming in and unauthorized Internet sessions going out. However, while firewalls can guard against intrusion and unauthorized use, they can guarantee neither the security of the connection between a particular workstation and a particular server on opposite sides of a firewall nor the security of the actual messages being conveyed. To create this level of security, Internet security protocols, such as the IP *Encapsulating Security Payload* (ESP), the *Secure Sockets Layer* (SSL) protocol, or the *Secure HyperText Transfer Protocol* (S-HTTP), ensure that sensitive information traveling through the Internet is safe from eavesdropping.

The primary goal of all network security protocols is to provide confidentiality, data integrity and authentication between communicating parties. The capabilities of a security protocol depend not only on its design but also on its location in the protocol stack.

This article describes various efforts to address security in three areas of the Internet protocol suite: the Internet Protocol itself (IPsec), the domain between transport and application layer (SSL and TLS, the *Transport Layer Security* protocol), and security extensions for the HyperText Transfer Protocol (S-HTTP). For each area the current technology, relevant standardization activities and likely future developments are discussed.

The paper shows that the technology for secure Internet communications is largely available. However, progress in this area is also influenced by several non-technical issues. E.g., the desire to avoid patented technologies, the mistrust of some specific techniques (e.g., key escrow, key recovery), and the existence of export controls and other regulations can hinder or delay the deployment of technologies based on cryptographic techniques.

2 The Internet Protocol Suite

Network protocols are typically developed in layers, with each layer responsible for a different aspect of the communication. Each layer has protocols for communicating with its peer at the same layer. The Internet protocol suite is considered to be a four-layer system, as shown in figure 1.

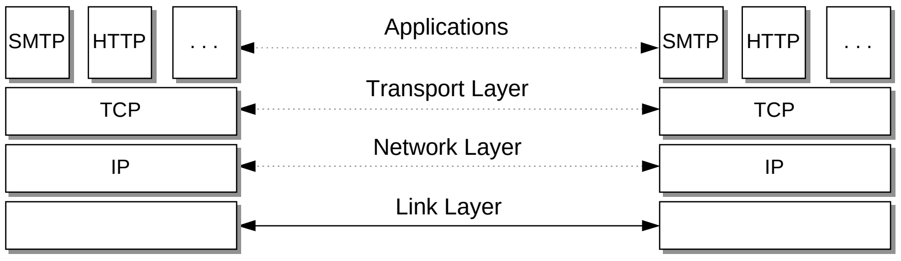


Fig. 1: Internet Protocols

The lowest layer is the link layer which provides the physical interfacing with the communications medium (e.g., the Ethernet cable). The network layer handles the movement of packets around the network. The transport layer provides a data flow between two hosts, for the application layer above. The application layer finally handles the details of the particular application.

IP, the *Internet Protocol* is the basis for the global Internet. Both IP version 4 (IPv4) and the next generation IP[1] (IPv6, [RFC1883]) provide unreliable data transmission between nodes. The network layer of the Internet includes not only IP, but also the *Internet Control Message Protocol* (ICMP) and the *Internet Group Membership Protocol* (IGMP). In the Internet protocol suite there are two different transport protocols. Most Internet applications use the *Transmission Control Protocol* (TCP) layered above IP which provides transmission reliability. Multicast applications and applications that do not need reliable transport use the much simpler *User Datagram Protocol* (UDP) instead.

Common applications provided by almost every TCP/IP implementation are telnet for remote login, the *File Transfer Protocol* ftp and SMTP, the *Simple Mail Transfer Protocol*. The *World Wide Web* (WWW) is a distributed hypermedia system which has gained widespread acceptance among Internet users. The native and primary protocol used between WWW clients and servers is the *HyperText Transfer Protocol* (HTTP).

3 Internet Standards

The *Internet Engineering Task Force* (IETF) is the technical group responsible for engineering and evolving the Internet, and the principal body engaged in the development of Internet specifications. Most of the technical work takes place in working groups. Currently active IETF working groups in the security area are:

[1] Even after IPv6 is deployed, IP version 4 is likely to remain used for a long time.

- *An Open Specification for Pretty Good Privacy* (OPENPGP)
- *Authenticated Firewall Traversal* (AFT)
- *Common Authentication Technology* (CAT)
- *Domain Name System Security* (DNSSEC)
- *IP Security Protocol* (IPSEC)
- *One Time Password Authentication* (OTP)
- *Public-Key Infrastructure (X.509)* (PKIX)
- *S/MIME Mail Security* (SMIME)
- *Secure Shell* (SECSH)
- *Simple Public Key Infrastructure* (SPKI)
- *Transport Layer Security* (TLS)
- *Web Transaction Security* (WTS)

Specifications for Internet security protocols are developed in three of those working groups: IPSEC, TLS, and WTS.

Documents called *Request for Comments* (RFCs) are the official documents of the *Internet Architecture Board* (IAB) which maintains the standards for the Internet protocol suite. Internet documents less mature or less stable than RFCs are typically available as *Internet-Drafts*. Internet-Drafts have no formal status, are valid for a maximum of six months, and can be changed or deleted at any time.

The RFC series comprises a wide range of documents, ranging from informational documents of general interest to specifications of protocols (*"all Internet standards are published as RFCs, but not all RFCs specify standards"*). Once a document is assigned an RFC number and published, it is never revised or re-issued with the same number. Therefore, the question is not having the most recent version of a particular RFC but having the most recent RFC on a particular protocol. RFCs are available from several Web-sites (e.g., http://ds.internic.net/ds/rfc-index.html).

There are two independent categorizations of RFCs. The first is the maturity level or state of standardization. Protocols which are to become Internet standards go through a series of states (*Proposed Standard*, *Draft Standard*, and *Standard*) involving increasing amounts of scrutiny and testing. At each step, the *Internet Engineering Steering Group* (IESG) must make a recommendation for the advancement of the protocol. Possible states of a RFC are:

- *Proposed Standard* (PS): Advancement from Internet-Draft to Proposed Standard puts a protocol "on the standards track", i.e., these are protocol proposals that may be considered by the IESG for standardization in the future. Implementation and testing by several groups is desirable.
- *Draft Standard* (DS): The IESG is actively considering such protocols as possible standards. It is general practice that no Proposed Standard can be promoted to Draft Standard without at least two independent implementations.

- *Standard* (STD): Promotion from Draft Standard to Standard generally requires operational experience and demonstrated interoperability of two ore more implementations.
- *Historic* (H): These are protocols that have been superseded by later developments or have become obsolete due to lack of interest.
- *Experimental* (E): Typically protocols developed as part of an ongoing research project and not related to an operational service offering.
- *Informational* (I): Other documents, such as protocols developed by other standard organizations, or by particular vendors, which are published as RFCs for the convenience of the Internet community.

The second categorization is the requirement level or status of a protocol, varying from *Required* (i.e., a system must implement it), *Recommended* (i.e., a system should implement it), *Elective*, *Limited Use*, to *Not Recommended*. Only protocols on the standards track (i.e., PS, DS, and STD protocols) are labeled with a status.

4 Security for the Internet Protocol

Cryptographic security mechanisms are not widely available on the Internet layer. In the 1980s, the *Secure Data Network System* (SDNS) project has developed a network security architecture including security protocols for the network and the transport layer [RFC1108]. SP3, the SDNS network security protocol, was designed as a sublayer residing directly below the transport layer. It offered the security services of confidentiality, data integrity and data origin authentication. The basic concepts of the ISO/IEC *Network Layer Security Protocol* (NLSP) [ISO11577] designed for the OSI network layer are largely derived from SP3. Due to the lack of commercial interest in the Internet at that time, the use of SP3 has been very limited.

During the past years, the *IP Security Protocol* (IPSEC) working group of the IETF has made progress in adding cryptographic security techniques to standards for the Internet infrastructure. The security architecture specified for IP provides cryptographic security services that support combinations of authentication, integrity, access control, and confidentiality.

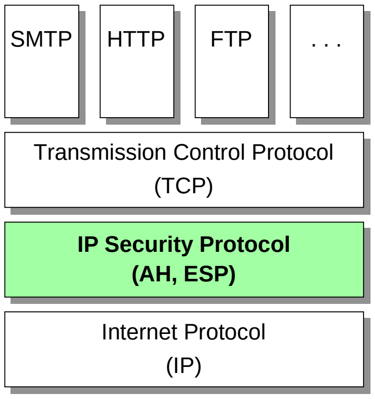


Fig. 2: IP Security Protocol

The *IP Security Architecture* [RFC1825], [KA98a][2] describes security mechanisms for IP version 4 (IPv4) and IP version 6 (IPv6). It defines two specific security headers. The first is the *Authentication Header* which provides integrity and authentication [RFC1826], [KA98b]. The second is the *Encapsulating Security Payload* (ESP) which provides confidentiality, and (depending on algorithms and modes) may also provide integrity and authentication [RFC1827], [KA98c].

The two IP security mechanisms can be used independently of one another, in combination with one another, or in an iterated (nested) fashion. They are defined algorithm-independent so that the cryptographic algorithms can be replaced without affecting the other parts of an implementation. Standard default algorithms are specified to ensure interoperability.

Both IP security mechanisms can provide security services between a pair of communicating hosts, between a pair of communicating security gateways, or between a security gateway and a host. Depending on a suitable key management, AH and ESP also are able to support multicast communications.

Support for key management methods where the keying material is carried in-line within the IP layer was not a design objective. Instead AH and ESP are intended to primarily use key management methods where the keying material will be carried by an upper layer protocol, or will be distributed manually.

Protocols and cryptographic techniques to support the key management requirements of the IP layer security are under development. The *Internet Key Management Protocol* (IKMP) is specified as an application layer protocol that is

[2] [KA98a], [KA98b] and [KA98b] are revised versions of the Proposed Standards [RFC1825], [RFC1826] and [RFC1827]. It is intended that future versions of these documents be submitted to the IESG for publication as Draft Standard RFCs.

independent of lower layer security protocols. IKMP is based on the ISAKMP/Oakley work [MSST97], [Orm96]. The Oakley key establishment protocol is a variant of the well-known Diffie-Hellman key exchange which provides perfect forward secrecy [Fum98].

Many of the basic concepts of the IP Security Architecture are derived from or were influenced by the SDNS security protocol specifications [SDNS], the NLSP specification [ISO11577], and from the SNMPv2 Security Protocol [RFC1446].

4.1 The IP Authentication Header

The IP *Authentication Header* (AH) [RFC1826], [KA98b] is designed to provide data integrity and data origin authentication to both IPv4 and IPv6 datagrams. In addition, it may provide non-repudiation, depending on which cryptographic algorithms are used and how key management is performed. The lack of a confidentiality mechanism is to ensure that AH implementations can be widely deployed, also in locations where the export, import, or use of encryption to provide confidentiality is regulated.

The IP Authentication Header adds authentication information (typically a *Message Authentication Code*) to an IP datagram. The authentication information is calculated using a secret authentication key and the fields in the IP datagram which do not change during transmission (including the IP Header, other headers and the user data). Fields or options which need to change in transit (e.g., "hop count", or "time to live") are zeroed for the calculation. The default length of the authentication information is 96 bits.

If a symmetric authentication algorithm is used and intermediate authentication is desired, then the nodes performing such intermediate authentication are able to forge or modify traffic. In the case of public-key techniques, intermediate nodes could authenticate the traffic without being able to forge or modify messages.

All IPv6-capable nodes and all IPv4 systems claiming to implement the Authentication Header must implement mandatory message authentication techniques. As of this writing these are HMAC-MD5 [MG98a] and HMAC-SHA [MG98b] (see below). Other algorithms may additionally be supported.

The revised specification of AH [KA98b] differs from RFC 1826 in several respects. The default algorithms required for interoperability have been changed from keyed MD5 (as specified in RFC 1828) to HMAC based on MD5 or SHA-1. Furthermore, an anti-replay service is now an integral part of AH and carriage of a 32-bit field that contains a monotonically increasing counter value (sequence number) to support this service is mandatory. The sequence number is used to guarantee that each packet exchanged between two parties is different. An authentication key must not be active for a period of time that allows the counter to wrap, that is, for more than 2^{32} packets.

IP Authentication Using Keyed MD5

A common suggestion for the calculation of authentication information is to construct a *Message Authentication Code* (MAC) from a hash-function by including a secret key as part of the hash-function input. RFC 1828 specifies how to use keyed MD5 for the Authentication Header. The shared authentication key is involved in both the start and the end of the MAC computation, a technique known as *envelope method with padding*.

MD5 hashes its input by iterating a basic compression function that operates on 512-bit blocks of data. To compute the keyed MAC,

$$\text{MD5}(\ K \,\|\, \text{pad} \,\|\, \text{data} \,\|\, K\ )$$

the key K is filled to the next 512-bit boundary, using the padding technique defined for MD5 [RFC1321]. The result is concatenated with the relevant fields of the IP datagram, and with the authentication key again. A trailing padding for the entire message to the next 512-bit boundary is added by MD5. The 128-bit hash-value is calculated as described in RFC 1321 and used as authentication information.

However, it has been shown that such one-key envelope methods are vulnerable to key recovery attacks [PV96]. Therefore, alternative techniques for the IP Authentication Header have been developed.

HMAC

RFC 2104 is an informational document that specifies an alternative keyed MAC transform called HMAC using a generic cryptographic hash-function. Candidate hash-functions for HMAC include MD5, SHA-1 [FIPS180-1], RIPEMD-128, and RIPEMD-160 [DBP96], [KP98].

HMAC is based on a hash-function which operates on blocks of B bytes of data. Let L denote the byte-length of the hash-value (e.g., L=16 for MD5 and RIPEMD-128, L=20 for SHA-1 and RIPEMD-160). The authentication key K can be of any length, the minimal recommended length for K is L bytes. Applications that use keys longer than B first need to hash the key and then use the result as the actual key to HMAC. K is appended with zeros to create a B byte string. Furthermore, two fixed strings *ipad* (0x36 repeated B times) and *opad* (0x5C repeated B times) are defined. HMAC then is

$$\text{hash}(\ K \oplus \text{opad} \,\|\, \text{hash}(\ K \oplus \text{ipad} \,\|\, \text{data}\ ))$$

Despite two hash-function calls, the HMAC construction is quite efficient, since the outer hash-function only processes a two-block input, independent of the length of the data.

[MG98a] specifies a keyed MD5 transform based on HMAC for use with either ESP or AH. To compute HMAC-MD5, the authentication key K is appended with zeros to create a 64 byte string. The mandatory key-length is 128 bits. HMAC-MD5 then is

$$\text{MD5}(\ K \oplus \text{opad} \ || \ \text{MD5}(\ K \oplus \text{ipad, data })).$$

MD5 generates a message digest of 128 bits. As described in RFC 2104, this 128-bit value can be truncated. For use with either ESP or AH, [MG98a] specifies to use the truncated value consisting of the first 96 bits.

[MG98b] specifies a similar transform using SHA-1:

$$\text{SHA}(\ K \oplus \text{opad} \ || \ \text{SHA}(\ K \oplus \text{ipad, data })).$$

SHA-1 generates a message digest of 160 bits which makes it more resistant to brute force attacks than MD5. Again, for application with either ESP or AH, [MG98b] specifies to only use the first 96 bits. The mandatory key-length for this HMAC transform is 160 bits.

4.2 The IP Encapsulating Security Payload

The IP *Encapsulating Security Payload* (ESP) [RFC1827], [KA98c] is designed to provide data confidentiality, data origin authentication, connectionless integrity, an anti-replay service, and limited traffic flow confidentiality for IP datagrams. Confidentiality may be selected independent of all other services and is always provided if ESP is present. Data origin authentication and connectionless integrity are joint services offered as an option in conjunction with confidentiality. As with AH, these services are provided by adding authentication information. The anti-replay service may only be selected if data origin authentication is selected. Alternatively, the IP Authentication Header may be used in conjunction with ESP to support data origin authentication, connectionless integrity, and replay detection.

Unlike AH, ESP offers security only for the protocols encapsulated by it, not for the protocol that carries it. The most obvious use of ESP is to provide security services for upper layer protocols such as TCP or UDP, without protection to the IP header that carries these protocols (this ESP use is referred to as Transport mode). Alternatively, ESP may encapsulate an entire IP datagram (Tunnel mode). Because ESP is an encapsulation protocol, it may be employed recursively, to create nested security associations.

Tunnel mode typically is used by security gateways for packets not originating on the gateway. It can be used to create virtual private networks across untrusted backbones via security gateways implementing ESP (gateway-to-gateway encryption). When two hosts directly implement ESP and there is no intervening security gateway (host-to-host encryption), they probably will use the Transport mode. This mode reduces both the bandwidth consumed and the protocol processing costs for users that don't need to keep the entire IP datagrams confidential.

The ESP format is designed to support a variety of encryption and authentication algorithms. If an algorithm is employed that requires the payload to be a multiple of some number of bytes (e.g., the block size of a block cipher), a padding field of up to 255 bytes is used. Padding may also be required for other alignment reasons or may be used to conceal the actual length of the payload. If padding is needed and not specified by the algorithm, a default process is used where the padding bytes are filled with a sequence of monotonically increasing 1-byte integer values, starting with 1.

For interoperability, all conforming implementations of ESP must implement a number of mandatory ESP algorithms. As of this writing, these are DES in CBC mode [MD98] (see below), HMAC-MD5 [MG98a] and HMAC-SHA-1 [MG98b]. Other algorithms may additionally be supported.

The revised specification of ESP [KA98c] differs from RFC 1827 in several significant ways. RFC 1827 provides a framework that is completed through the definition of transforms that may be offered in the context of ESP. The combinatorial growth of proposed transforms motivated the reformulation of the ESP as a more complete specification. Fields previously defined in transform documents are now part of the base ESP specification. E.g., the fields necessary to support data origin authentication and anti-replay are defined in [KA98c], even though the provision of these services is an option.

DES-CBC Transforms

RFC 1829 describes the ESP use of the *Cipher Block Chaining* (CBC) mode [ISO10116] of the *Data Encryption Standard* (DES) [FIPS46-1]. The transform provides data confidentiality only.

The keying material necessary for this transform consists of the secret 56-bit DES key shared between the communicating parties and a 64-bit Initialization Vector (IV) required for the CBC mode. The IV value should not repeat during the lifetime of the DES key. Therefore, the session key should be changed at least after 2^{32} enciphered datagrams.

DES is a block cipher that operates on 64-bit blocks. This typically requires padding after the end of the unencrypted payload data. For encryption, the plaintext is padded with zeroes to make its byte-length equal to 6 modulo 8. Then, a byte containing the number of padding octets and a byte identifying the protocol header is appended. The payload is encrypted with DES in CBC mode, producing a ciphertext of the same length.

RFC 1851 is an experimental RFC that specifies a variant of the DES-CBC transform which uses Triple-DES (3DES-EDE) instead of DES. In this case, the shared key is 168 bits long and consists of three independent 56-bit quantities.

[MD98] is a revised version of RFC 1829 that requires an explicit 64-bit Initialization Vector. This IV immediately precedes the encrypted payload and must be a random value. Including the IV in each datagram ensures that decryption of each received datagram can be performed, even when some datagrams are dropped, or datagrams are re-ordered in transit. Unlike RFC 1829, [MD98] does not specify a padding technique, i.e., for padding the default process specified in [KA98c] is used.

4.3 Combining AH and ESP

A node normally will not apply both AH and ESP to the same IP datagram. The datagrams transmitted over an individual *Security Association* (SA) are protected by exactly one security protocol, either AH or ESP. If confidentiality is a requirement, then the ESP header needs to be used, otherwise the Authentication Header should be used.

Sometimes a security policy may call for a combination of security services for a particular traffic flow that is not achievable with a single SA. In such instances it will be necessary to employ multiple SAs to implement the required security policy. SAs may be combined in two ways - transport adjacency and iterated tunneling. These two approaches can also be combined.

Transport adjacency refers to applying more than one security protocol to the same IP datagram without invoking tunneling. This approach to combining AH and ESP allows for only one level of combination, since further nesting yields no added benefits. The Authentication Header is placed before the ESP header and is calculated across the entire IP datagram, i.e., AH is applied to the ciphertext output of ESP.

Iterated tunneling refers to applying multiple layers of security protocols effected through IP tunneling. This approach allows for multiple levels of flexible nesting, since each tunnel can originate or terminate at different sites along the path.

5 Transport Layer Security Protocols

A variety of specific protocols has been proposed for adding security services to TCP. An important advantage of transport layer security protocols is that they are application protocol independent and that higher level protocols can transparently layer on top of them. Prominent examples for transport layer security are Netscape's *Secure Sockets Layer* (SSL) protocol, Microsoft's *Private Communication Technology* (PCT) and the *Transport Layer Security* (TLS) protocol currently being standardized by the IETF.

5.1 SSL

The *Secure Sockets Layer* (SSL) protocol was developed by Netscape Communications. SSL operates above the transport layer and is application independent. The protocol provides authentication, data compression, data integrity, and data encipherment. It supports various authentication and encryption techniques and can protect any application-layer protocol such as the *HyperText Transfer Protocol* (HTTP), ftp, telnet, or the *Simple Mail Transport Protocol* (SMTP).

Netscape’s Navigator and Microsoft's Internet Explorer both support SSL at the browser level, and Netscape's Commerce Server and several other Web servers support SSL at the server level. When both a client (typically in the form of a Web browser) and a server support SSL, a secure link is established. E.g., when using the Netscape Navigator, a user can tell by the key icon on the screen that a session is encrypted. The key, which is usually broken, then becomes whole. In addition, the first part of the URL changes from ‘http://’ to ‘https://’.

SSL is being considered by the IETF as a possible Internet standard. *Transport Layer Security* (TLS) is the IETF working group responsible for moving security protocols such as SSL through the standards tracks. The first official TLS working group meeting was in June 1996, before then the group was an unofficial BOF ("birds of a feather").

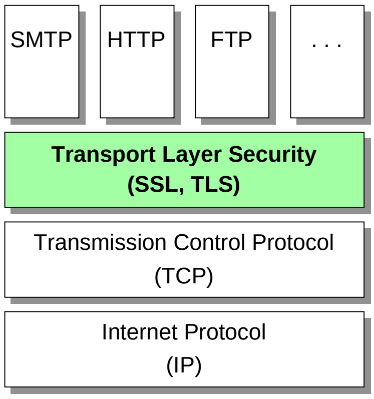

SMTP
HTTP
FTP
. . .
Transport Layer Security (SSL, TLS)
Transmission Control Protocol (TCP)
Internet Protocol (IP)

Fig. 3: Transport Layer Security Protocols

5.2 TLS

The current *Transport Layer Security* (TLS) protocol version 1.0 [DA97] is based on the *Secure Sockets Layer* (SSL) version 3.0 protocol [FKK96]. Like SSL, TLS is composed of two layers, the Handshake Protocol and the Record Protocol. The TLS Record Protocol is layered on top of some reliable transport protocol, typically TCP (see figure 3). It offers two basic services:

- *Connection confidentiality*. Symmetric cryptographic techniques (e.g., DES, IDEA, RC2, RC4, etc.) are used for message encipherment. TLS supports both stream and block ciphers. In the case of a stream cipher, the plaintext is XORed with the output generated from a keyed pseudo-random generator. Block ciphers typically are used in CBC mode [ISO10116].

- *Connection integrity.* Message transport includes a data integrity check based on a keyed MAC. Cryptographic hash-functions (e.g., SHA-1, MD5) and the HMAC construction [RFC2104] are used for the MAC computations.

The TLS Record Protocol is used for encapsulation of higher level protocols. One such encapsulated protocol is the TLS Handshake Protocol which allows server and client to authenticate each other and to negotiate an encipherment algorithm and cryptographic keys before the application protocol transmits or receives data. In particular, the TLS Handshake Protocol provides the following services:

- *Entity authentication.* The peer's identities can be authenticated using asymmetric cryptographic techniques (e.g., RSA, DSA). TLS makes use of digital signatures with appendix, i.e. one-way hash-functions are used to generate the input for the signature algorithm. In the case of RSA, the 288-bit output of two hash-functions (SHA-1 and MD5) is signed; DSA mandates the use of SHA-1. Entity authentication can be made optional, but is generally required for at least one of the peers.
- *Key establishment.* A shared secret is negotiated between the peers. The TLS Record Protocol uniquely derives the keys for encipherment and data integrity from the keying material established by the TLS Handshake Protocol.

The TLS Record Protocol

The TLS Record Protocol takes messages of arbitrary size to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, enciphers, and transmits the result. Transmissions include a sequence number so that missing, altered, or inserted messages are detectable. At the receiving end, messages are deciphered, verified, decompressed, reassembled, and then delivered to higher protocol levels.

The cryptographic techniques supported by a SSL or TLS implementation are specified in the form of *CipherSuites*. The versions of SSL and TLS which are exportable from the United States currently are restricted to 40-bit keys for encipherment and 512-bit moduli for key establishment. A special case is the CipherSuite TLS_NULL_WITH_NULL_NULL, where the message is not encrypted and the protocol operates without a MAC.

In SSL version 3.0, the MAC computations are based on an early version of HMAC [BCK96]. The MAC is computed before encipherment as

$$\text{hash}(\ K \,\|\, \text{opad} \,\|\, \text{hash}(\ K \,\|\, \text{ipad} \,\|\, \text{seq\_num} \,\|\, \text{data}\ )),$$

where *ipad* and *opad* are repetitions of the bytes 0x36 and 0x5C to fill up one block for the hash-function. This proposal uses a padding of fixed length which is combined with the authentication key using XOR rather than concatenation (see section 4.1). In TLS version 1.0 this MAC computation has been replaced by the version of HMAC as described in RFC 2104.

A stream cipher directly encrypts the entire message, including the MAC. In the case of a block cipher, padding is added to force the length of the plaintext to be a multiple of the block cipher's block length. The initialization vector (IV) needed for the CBC mode is generated uniquely for each connection.

The TLS Record Protocol generates keys, IVs, and MAC secrets from the security parameters provided by the Handshake Protocol. Those are a 48 byte master_secret shared between the two peers in the connection, a 32 byte client_random r_C provided by the client and a 32 byte server_random r_S provided by the server. A pseudo-random function is used to expand the security parameters into a sequence of arbitrary length. The construction uses two hash algorithms (MD5 and SHA-1) in a way designed to guarantee its security if either algorithm remains secure.

TLS first defines a data expansion function P_h which uses a single hash function to expand a secret and a seed value into an arbitrary quantity of output:

$$\begin{aligned} &P_h(\text{secret, seed}) = \\ &\qquad \text{HMAC}_h(\text{secret}, \text{HMAC}_h(\text{secret, seed}) \mathbin{\|} \text{seed}) \mathbin{\|} \\ &\qquad \text{HMAC}_h(\text{secret}, \text{HMAC}_h(\text{secret}, \text{HMAC}_h(\text{secret, seed})) \mathbin{\|} \text{seed}) \mathbin{\|} \\ &\qquad \ldots \end{aligned}$$

P_h can be iterated as necessary to produce the required quantity of data. The pseudo-random function PRF of TLS is then created by splitting the secret into two halves S_1 and S_2 and using P_{MD5} to generate pseudo-random data from S_1 and P_{SHA-1} to generate data from S_2. In a final step the outputs of these two expansion functions are XORed together. The pseudo-random function uses an additional parameter 'label' which is an ASCII string:

$$\text{PRF(secret, label, seed)} = P_{MD5}(S_1, \text{label} \mathbin{\|} \text{seed}) \oplus P_{SHA-1}(S_2, \text{label} \mathbin{\|} \text{seed}).$$

The TLS Handshake Protocol

The cryptographic parameters for a session are provided by the Handshake Protocol, which operates on top of the Record Protocol. When a TLS client and server first start communicating, they agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and use public-key techniques to negotiate shared secrets. The Handshake Protocol can be summarized as follows:

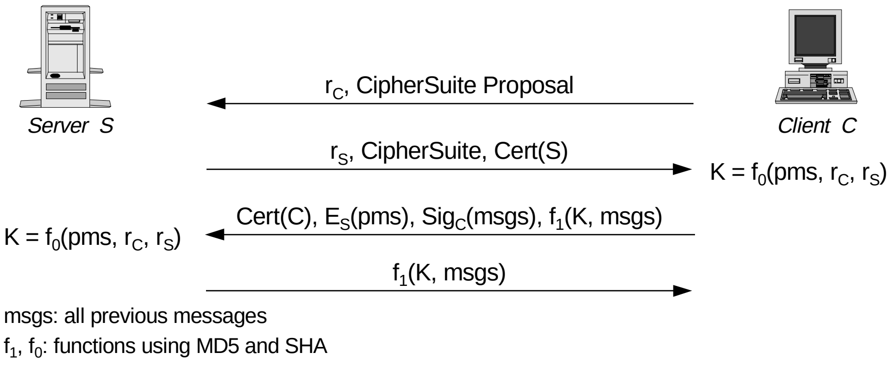
- Client and server exchange hello messages which are used to establish security parameters between client and server, i.e., Protocol Version, Session ID, CipherSuite, and Compression Method. Additionally, two random values are generated and exchanged: client_random and server_random.
- Client and server exchange the necessary cryptographic parameters to establish a pre_master_secret. Currently defined key establishment methods provide secrets which range from 48 to 128 bytes in length.
- Client and server exchange certificates and other information to allow unilateral or mutual entity authentication.

- Client and server generate a master_secret from the pre_master_secret and the exchanged random values and provide the master_secret and the exchanged random values to the Record Protocol.
- Finally, the Handshake Protocol allows the entities to verify that their peer has calculated the same security parameters, including key confirmation.

The key establishment method is specified by the negotiated CipherSuite. For all key exchange methods, the same transform is used to calculate the master_secret from the pre_master_secret:

$$\text{master\_secret} = \text{PRF}(\text{pre\_master\_secret}, \text{"master secret"}, r_C \,||\, r_S).$$

The length of the master_secret is truncated to 48 bytes while the length of the pre_master_secret varies depending on the key establishment method. When RSA is used for server authentication and key exchange, a 48-byte pre_master_secret is generated by the client, encrypted under the server's public key, and sent to the server. Figure 4 shows an example for the TLS Handshake Protocol using RSA key transport.



r_C, CipherSuite Proposal
Server S
Client C
r_S, CipherSuite, Cert(S)
$K = f_0(pms, r_C, r_S)$
Cert(C), E_S(pms), Sig_C(msgs), f_1(K, msgs)
$K = f_0(pms, r_C, r_S)$
f_1(K, msgs)
msgs: all previous messages
f_1, f_0: functions using MD5 and SHA

Fig. 4: TLS Handshake Protocol

In the case of Diffie-Hellman key agreement, a conventional Diffie-Hellman computation is performed and the negotiated key is used as the pre_master_secret. Diffie-Hellman parameters are specified by the server, and may be either ephemeral or contained within the server's certificate.

TLS CipherSuites

The following tables list the non-trivial CipherSuite definitions of TLS Version 1.0 [DA97]. In addition, the CipherSuite TLS_NULL_WITH_NULL_NULL is specified. Unlike SSL, TLS specifies a mandatory CipherSuite, i.e. a TLS compliant application must implement TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA (see table 2).

The CipherSuite definitions of table 1 require that the server provides an RSA certificate that is used for authentication and key transport. The CipherSuite definitions of table 2 are used for server-authenticated (and optionally client-authenticated) Diffie-Hellman key agreement.

CipherSuite	Export	Key Establishment	Cipher (key size)	HMAC
RSA_WITH_NULL_MD5	yes	RSA	NULL	MD5
RSA_WITH_NULL_SHA	yes	RSA	NULL	SHA-1
RSA_EXPORT_WITH_ RC4_40_MD5	yes	RSA_EXPORT[3] (max 512 bits)	RC4_40 (40 bits)	MD5
RSA_WITH_ RC4_128_MD5	no	RSA	RC4_128 (128 bits)	MD5
RSA_WITH_ RC4_128_SHA	no	RSA	RC4_128 (128 bits)	SHA-1
RSA_EXPORT_WITH_ RC2_CBC_40_MD5	yes	RSA_EXPORT (max 512 bits)	RC2_CBC_40 (40 bits)	MD5
RSA_WITH_ IDEA_CBC_SHA	no	RSA	IDEA_CBC (128 bits)	SHA-1
RSA_EXPORT_WITH_ DES40_CBC_SHA	yes	RSA_EXPORT (max 512 bits)	DES40_CBC (40 bits)	SHA-1
RSA_WITH_ DES_CBC_SHA	no	RSA	DES_CBC (56 bits)	SHA-1
RSA_WITH_ 3DES_EDE_CBC_SHA	no	RSA	3DES_EDE_CB C (168 bits)	SHA-1

Table 1: TLS CipherSuites (1)

DH denotes cipher suites in which the server's certificate contains the Diffie-Hellman parameters signed by a certification authority (CA). DHE denotes ephemeral Diffie-Hellman key agreement, where the Diffie-Hellman parameters are signed by DSA or RSA and where a certificate has been issued for use with DSA or RSA. In all

[3] The U.S. National Security Agency imposes restrictive munitions-export controls on encipherment technologies. Therefore, exportable versions of each of these technologies must make use of weaker encipherment than is permitted for domestic use. Current US export restrictions limit RSA keys used for encipherment to 512 bits, but do not place any limit on lengths of RSA keys used for digital signatures.

cases, client and server must have the same type of certificate, and must use the Diffie-Hellman parameters chosen by the server.

CipherSuite	**Ex-port**	**Key Establishment**	**Cipher (key size)**	**HMAC**
DH_DSS_EXPORT_WITH_ DES40_CBC_SHA	yes	DH_DSS_EXPORT (max 512 bits)	DES40_CBC (40 bits)	SHA-1
DH_DSS_WITH_ DES_CBC_SHA DH_DSS	no	DH_DSS	DES_CBC (56 bits)	SHA-1
DH_DSS_WITH_ 3DES_EDE_CBC_SHA	no	DH_DSS	3DES_EDE_CBC (168 bits)	SHA-1
DH_RSA_EXPORT_WITH_ DES40_CBC_SHA	yes	DH_RSA_EXPORT (max 512 bits)	DES40_CBC (40 bits)	SHA-1
DH_RSA_WITH_ DES_CBC_SHA	no	DH_RSA	DES_CBC (56 bits)	SHA-1
DH_RSA_WITH_ 3DES_EDE_CBC_SHA	no	DH_RSA	3DES_EDE_CBC (168 bits)	SHA-1
DHE_DSS_EXPORT_WITH _DES40_CBC_SHA	yes	DHE_DSS_EXPORT	DES40_CBC (40 bits)	SHA-1
DHE_DSS_WITH_ DES_CBC_SHA	no	DHE_DSS	DES_CBC (56 bits)	SHA-1
DHE_DSS_WITH_ 3DES_EDE_CBC_SHA	**no**	**DHE_DSS**	**3DES_EDE_CBC (168 bits)**	**SHA-1**
DHE_RSA_EXPORT_WITH _DES40_CBC_SHA	yes	DHE_RSA_EXPORT (max 512 bits)	DES40_CBC (40 bits)	SHA-1
DHE_RSA_WITH_ DES_CBC_SHA	no	DHE_RSA	DES_CBC (56 bits)	SHA-1
DHE_RSA_WITH_ 3DES_EDE_CBC_SHA	no	DHE_RSA	3DES_EDE_CBC (168 bits)	SHA-1

Table 2: TLS CipherSuites (2)

Finally, the CipherSuites shown in table 3 are used for anonymous Diffie-Hellman key agreement where neither party is authenticated. Note that this key establishement method is vulnerable to man-in-the-middle attacks and therefore can not be recommended for general use.

CipherSuite	Ex-port	Key Establishment	Cipher (key size)	HMAC
DH_anon_EXPORT_WITH _RC4_40_MD5	yes	DH_anon_EXPORT (max 512 bits)	RC4_40 (40 bits)	MD5
DH_anon_WITH_ RC4_128_MD5	no	DH_anon	RC4_128 (128 bits)	MD5
DH_anon_EXPORT_WITH _DES40_CBC_SHA	yes	DH_anon_EXPORT (max 512 bits)	DES40_CBC (40 bits)	SHA-1
DH_anon_WITH_ DES_CBC_SHA	no	DH_anon	DES_CBC (56 bits)	SHA-1
DH_anon_WITH_ 3DES_EDE_CBC_SHA	no	DH_anon	3DES_EDE_CBC (168 bits)	SHA-1

Table 3: TLS CipherSuites (3)

6 The Secure HyperText Transfer Protocol

The *HyperText Transfer Protocol* (HTTP) [RFC2068] is the primary protocol used between WWW clients and servers. The original HTTP specification has only modest support for security. The *Web Transaction Security* (WTS) working group of the IETF develops requirements [RFC2084] and specifications for the provision of security services to transactions using HTTP.

Secure HTTP (S-HTTP) is a message-oriented communications protocol for securing messages that are transmitted using HTTP [RS96]. The protocol provides security services for transaction confidentiality, data integrity and non-repudiation of origin. S-HTTP offers flexibility in choice of key management, cryptographic algorithms and security policies by supporting option negotiation between parties for each transaction. Several cryptographic message format standards may be incorporated into S-HTTP clients and servers, particularly, PKCS-7 [PKCS7] and MOSS.

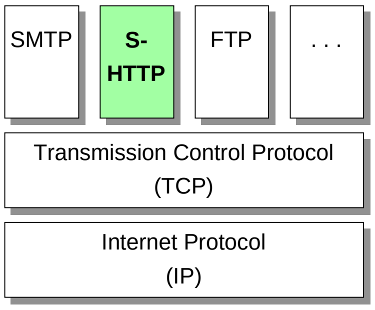


Fig. 5: Secure HyperText Transfer Protocol

While SSL and TLS operate at the transport layer, S-HTTP supports secure end-to-end transactions by adding cryptographic techniques to messages at the application layer. Therefore, while SSL and TLS are application independent, S-HTTP is tied to the HTTP protocol and cannot protect applications that use protocols other than HTTP. On the other hand, while SSL encrypts the entire communications link between a client and a server, S-HTTP is able to encrypt messages on an individual basis.

S-HTTP message protection may be provided in any combination of the following three respects: digital signatures, data integrity, and data encipherment. In addition, the protocol provides a simple challenge-response mechanism based on nonces, allowing both parties to insure the freshness of transmissions. S-HTTP does not necessarily require client-side public-key certificates (or public keys), as it supports symmetric key-only operation. Multiple key management mechanisms are supported, including manually shared secrets, public-key certificates and Kerberos [RFC1510] tickets. In particular, provision has been made for prearranged symmetric session keys.

- *Digital Signatures:* If digital signatures are applied, an appropriate certificate (in X.509 or X.509v3 format) may either be attached to the message or the sender may expect the recipient to obtain the required certificate independently. An explicitly permitteded instance of this kind is a certificate signed with the private key corresponding to the public key being attested to (self-signed certificate). S-HTTP version 1.2 allows for the signature algorithms RSA [PKCS1] and DSA [FIPS186].

- *Symmetric Encipherment:* For bulk encryption, block ciphers are used in CBC mode, for message header encipherment they are used in Electronic Codebook (ECB) mode [ISO10116]. Encipherment is performed as enveloping under PKCS-7 [PKCS7]. Currently defined encipherment algorithms are: DES, DES-EDE (2-

key Triple-DES), DES-EDE3 (3-key Triple-DES), DESX[4], IDEA, RC2, and CDMF[5].

- *Key Establishment:* In support of encipherment, S-HTTP defines two key establishment mechanisms, one using public-key key transport and one with externally arranged keys. In the former case, the symmetric key is transported enciphered under the receiver's public key. In the latter mode, the message content is encrypted using a prearranged session key. Alternatively, keys may be extracted from Kerberos tickets.
- *Message Integrity and Sender Authentication:* S-HTTP provides a means to verify message integrity and sender authenticity via the computation of a keyed hash-value over the encapsulated content of S-HTTP messages. S-HTTP version 1.2 uses the HMAC construction [RFC2104] based on MD2 [RFC1319], MD5, or SHA-1.

S-HTTP defines a new URL protocol designator, 'shttp'. Use of this designator as part of an anchor URL implies that the target server is S-HTTP capable, and that a dereference of this URL involves S-HTTP processing. S-HTTP oblivious agents should not dereference a URL with an unknown protocol specifier, and hence sensitive data will not be accidentally sent in the clear by users of nonsecure clients.

For interoperability, all S-HTTP agents must support the hash-function MD5 and MD5 based message authentication. As of S-HTTP/1.2, agents must also support HMAC-MD5. In addition, all S-HTTP agents must support outband key exchange. Support for encipherment is not required but only recommended; agents which implement encipherment must support one of the following three algorithms in ECB and CBC modes: DES, 40-bit RC2 and CDMF [JMLW93]. Agents are also recommended to support signature verification; server support of signature generation is additionally recommended. The preferred signature algorithm is RSA.

References

[BCK96] Bellare, M.; Canetti, R.; Krawczyk, H.: "Keying Hash Functions for Message Authentication", Proceedings of Crypto'96, Springer LNCS 1109 (1996), 1-15.

[Bel89] Bellovin, S.M.; "Security Problems in the TCP/IP Protocol Suite", ACM Computer Communications Review, Vol. 19, No. 2, March 1989.

[DA97] Dierks, T.; Allen, C.: "The TLS Protocol - Version 1.0", Internet-Draft, November 1997.

---

[4] DESX is an extension of DES that uses two additional 64-bit keys. Those keys are simply XORed with the plaintext block and the ciphertext block, respectively [KR96].

[5] CDMF (*Commercial Data Masking Facility*) is a variant of DES designed to meet export regulations. CDMF uses a 40-bit key [JMLW93].

[DBP96] Dobbertin, H.; Bosselaers, A.; Preneel, B.: "RIPEMD-160: A Strengthened Version of RIPEMD", Fast Software Encryption, Springer LNCS 1039 (1996), 71-82.

[FIPS180-1] NIST FIPS PUB 180-1: *Secure Hash Standard*, April 1995.

[FIPS186] NIST FIPS PUB 186: *Digital Signature Standard*, May 1994.

[FIPS46-1] NIST FIPS PUB 46-1: *Data Encryption Standard*, reaffirmed January 1988 (supersedes FIPS PUB 46, 1977).

[FKK96] Freier, A.O.; Karlton, P.; Kocher, P.C.: "The SSL 3.0 Protocol", Internet-Draft, November 1996.

[Fum98] Fumy, W.: "Key Management Techniques", this volume, 143-164.

[ISO10116] ISO/IEC 10116: *Modes of operation for an n-bit block cipher algorithm*, 2nd ed. 1997.

[ISO11577] ISO/IEC 11577: *Network Layer Security Protocol*, 1995.

[JMLW93] Johnson, D.B.; Matyas, S.M.; Le, A.; Wilkins, J.: "Design of the Commercial Data Masking Facility Data Privacy Algorithm", Proceedings 1st ACM Conference on Computer & Communications Security, November 1993, Fairfax, VA., 93-96.

[KA98a] Kent, S.; Atkinson, R.: "Security Architecture for the Internet Protocol", Internet-Draft, February 1998.

[KA98b] Kent, S.; Atkinson, R.: "IP Authentication Header", Internet-Draft, February 1998.

[KA98c] Kent, S.; Atkinson, R.: "IP Encapsulating Security Payload (ESP)", Internet-Draft, February 1998.

[KP98] Keromytis, A.D.; Provos, N.: "The Use of HMAC-RIPEMD-160-96 within ESP and AH", Internet-Draft, February 1998.

[KR95] Kaliski, B.; Robshaw, M.: "Message authentication with MD5", CryptoBytes, vol.1 no.1, Spring 1995.

[KR96] Kilian, J.; Rogaway, P.: "How to Protect DES Against Exhaustive Key Search", Proceedings of Crypto'96, Springer LNCS 1109 (1996), 252-267.

[MD98] Madson, C.; Doraswamy, N.: "The ESP DES-CBC Cipher Algorithm With Explicit IV", Internet-Draft, February 1998.

[MG98a] Madson, C.; Glenn, R.: "The Use of HMAC-MD5-96 within ESP and AH", Internet-Draft, February 1998.

[MG98b] Madson, C.; Glenn, R.: "The Use of HMAC-SHA-1-96 within ESP and AH", Internet-Draft, February 1998.

[MSST97] Maughan, D.; Schertler, M.; Schneider, M.; Turner, J.: "Internet Security Association and Key Management Protocol (ISAKMP)", Internet-Draft, July 1997.

[Orm96] Orman, H.K.: "The Oakley Key Determination Protocol", Internet-Draft, May 1996.

[PKCS1] RSA Laboratories: "PKCS #1: RSA Encryption Standard", version 1.5, November 1993.

[PKCS7] RSA Laboratories: "PKCS #7: RSA Cryptographic Message Syntax Standard", version 1.5, November 1993.

[PV96] Preneel, B.; van Oorschot, P.: "On the Security of two MAC Algorithms", Proceedings of Eurocrypt'96, Springer LNCS 1070 (1996), 19-32.

[RFC1108] Kent, S., "US DoD Security Options for the Internet Protocol", RFC 1108 (H), November 1991.

[RFC1319] Kaliski, B.: "The MD2 Message-Digest Algorithm", RFC1319 (I), April 1992.

[RFC1321] Rivest, R.: "The MD5 Message-Digest Algorithm", RFC1321 (I), April 1992.

[RFC1446] Galvin, J.; McCloghrie, K.: "Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1446 (PS), April 1993.

[RFC1510] Kohl, J.; Neuman, B.: "The Kerberos Network Authentication Service (V5)", RFC 1510 (PS), September 1993.

[RFC1825] Atkinson, R.: "Security Architecture for the Internet Protocol", RFC 1825 (PS), August 1995.

[RFC1826] Atkinson, R.: "IP Authentication Header", RFC 1826 (PS), August 1995.

[RFC1827] Atkinson, R.: "IP Encapsulating Security Payload (ESP)", RFC 1827 (PS), August 1995.

[RFC1828] Metzger, P.; Simpson, W.: "IP Authentication using Keyed MD5", RFC 1828 (PS), August 1995.

[RFC1829] Karn, P., Metzger, P., Simpson, W.: "The ESP DES-CBC Transform", RFC 1829 (PS), August 1995.

[RFC1851] Karn, P.; Metzger, P.; Simpson, W.: "The ESP Triple DES Transform", RFC 1851 (E), September 1995.

[RFC1883] Deering, S.; Hinden, R.: "Internet Protocol version 6 (IPv6) Specification", RFC 1883 (PS), December 1995.

[RFC2068] Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Berners-Lee, T.: "Hypertext Transfer Protocol - HTTP/1.1", RFC 2068 (PS), January 1997.

[RFC2084] Bossert, G.; Cooper, S.; Drummond, W.: "Considerations for Web Transaction Security", RFC 2084 (I), January 1997.

[RFC2104] Krawczyk, H., Bellare, M., Canetti, R.: "HMAC: Keyed-Hashing for Message Authentication", RFC 2104 (I), February 1997.

[RFC2144] Adams, C.: "The CAST-128 Encryption Algorithm", RFC 2144 (I), May 1977.

[RFC2202] Cheng, P.; Glenn, R.: "Test Cases for HMAC-MD5 and HMAC-SHA-1", RFC 2202 (I), March 1997.

[RS96] Rescorla, E.; Schiffman, A.: "The Secure HyperText Transfer Protocol", Internet-Draft, July 1996.

[SDNS] Secure Data Network System: "Security Protocol 3 - SP3", Document SDN.301, Revision 1.5, 15 May 1989.