SFWR 4C03: Computer Networks & Computer Security	Mar 15-18, 2004
Lecturer: Kartik Krishnan	Lecture 27-30

1 Introduction

We compare the public and symmetric key cryptographic schemes in Table 1. We also compare how these two cryptosystems fare w.r.t confidentiality, authentication and signature in Table 2.

Authenication Services:

Message authentication is a procedure to verify that received messages come from the alleged source and have not been altered in transit. They may also verify sequencing (any modification to a sequence of messages) and timeliness (delay or replay of messages). On the other hand, a digital signature also includes measures to counter repudiation by the source. There is an essential difference between authentication and signatures: authentication is to protect the two communicating parties (Alice/Bob) from a 3rd party (Oscar/Trudy) who masquerades as either Alice or Bob, or modifications of the messages in transit. Authentication cannot help if Alice and Bob do not trust each other; a digital signature is a solution to this problem. Alice's digital signature on a message reassures Bob that it indeed came from Alice, and Alice cannot deny sending this message at a later time.

Any message authentication or digital signature mechanism can be viewed as having fundamentally two levels. At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate the message. This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message. There are two classes of functions that can be used to produce an authenticator:

1. **Message Authentication Code:** A publicly available function that uses the plaintext message and a secret key to produce a fixed-length value message that serves as the authenticator.

2. **Hash function:** A publicly available function that maps a plaintext message of any length into a fixed-length hash value, which serves as the authenticator.

Table 9.1 CONVENTIONAL AND PUBLIC-KEY ENCRYPTION

Conventional Encryption	Public-Key Encryption
Needed to Work:	*Needed to Work:*
1. The same algorithm with the same key is used for encryption and decryption.	1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption.
2. The sender and receiver must share the algorithm and the key.	2. The sender and receiver must each have one of the matched pair of keys (not the same one).
Needed for Security:	*Needed for Security:*
1. The key must be kept secret.	1. One of the two keys must be kept secret.
2. It must be impossible or at least impractical to decipher a message if no other information is available.	2. It must be impossible or at least impractical to decipher a message if no other information is available.
3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Figure 1: Symmetric and Public-Key Encryption

Table 11.1 Confidentiality and Authentication Implications of Message Encryption (see Figure 11.1)

A → B: $E_K[M]$ •Provides confidentiality —Only A and B share K •Provides a degree of authentication —Could come only from A —Has not been altered in transit —Requires some formatting/redundancy •Does not provide signature —Receiver could forge message —Sender could deny message (a) Symmetric encryption
A → B: $E_{KU_b}[M]$ •Provides confidentiality —Only B has KR_b to decrypt •Provides no authentication —Any party could use KU_b to encrypt message and claim to be A (b) Public-key encryption: confidentiality
A → B: $E_{KR_a}[M]$ •Provides authentication and signature —Only A has KR_a to encrypt —Has not been altered in transit —Requires some formatting/redundancy —Any party can use KU_a to verify signature (c) Public-key encryption: authentication and signature
A → B: $E_{KU_b}[E_{KR_a}(M)]$ •Provides confidentiality because of KU_b •Provides authentication and signature because of Kr_a (d) Public-key encryption: confidentiality, authentication, and signature

Figure 2: Confidentiality and Authentication Implications in Symmetric and Public-key cryptography

Message Authentication code:

An overview of the entire process is given in Figure 3. message that serves as

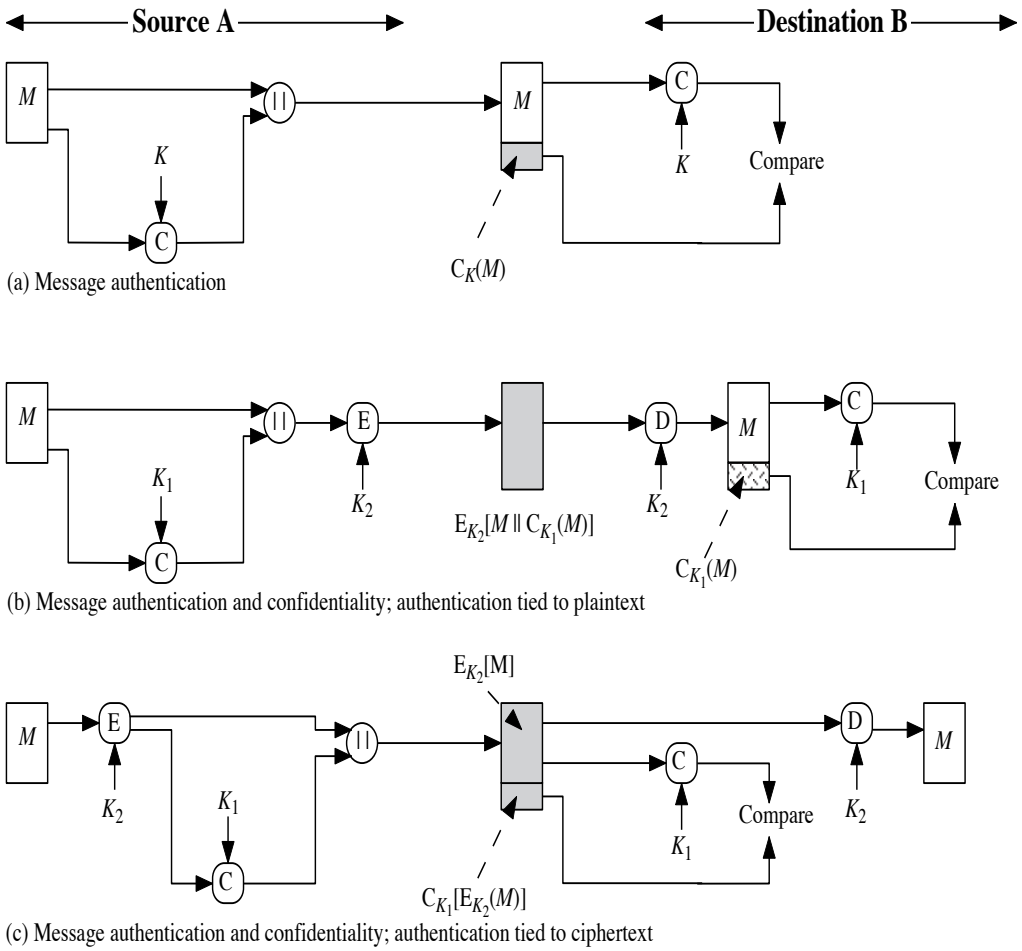


Figure 11.4 Basic Uses of Message Authentication Code (MAC)

Figure 3: Basic use of a message authentication code

the authenticator. The MAC authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as cryptographic checksum or MAC that is appended to the message. This technique assumes that the communicating parties Alice and Bob share a common secret key K. When Alice has a message to send to Bob, she calculates the MAC as a function of the message M and the key K, i.e., $MAC = C_K(M)$. The message M plus MAC are transmitted to the intended recipient. Bob performs the same calculation on the received message, using the same key, to generate a new MAC. The received MAC is compared to the calculated MAC (see Figure 3). If we assume that

only Alice and Bob know the identity of the secret key, then Bob is assured that the message has not been altered in transit. The idea is that since the attacker does not know the secret key he/she cannot alter the MAC to correspond to alterations in the message. Also, Bob knows that the message came from Alice, since she alone knew the secret key.

A MAC function is similar to encryption. One difference is that since the MAC maps an arbitrary length message into a shorter fixed-length message, the MAC function is general a many-to-one function.

Finally, computing the MAC function is much faster than conventional encryption.

Hash function:

An overview of the entire process is shown in Figures 4 and 5. This is a variation of the message authentication code. As with the MAC, a hash function accepts a variable-size message M as input and produces a fixed-size output, referred to as a hash code $H(M)$. Unlike a MAC, a hash code does not use a key but is a function only of the input message. The hash code is also referred to as a message digest or hash value. The hash code is a function of all the bits of the message and provides an error-detection capability. A change to any bit or bits results in a change to the hash code.

A hash function is much faster to compute from a MAC since no key is involved in the computation, but a MAC provides better authentication in an unsecure channel.

The requirements for a hash function H are the following:

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. $H(x)$ is relatively easy to compute for any given x.
4. For any given value h, it is computationally infeasible to find an x such that $H(x) = h$. Thus, the hash function should be a *one-way* function.
5. For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$. This is sometimes referred to as the *weak collision resistance* property.
6. It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$. This is referred to as the *strong collision resistance* property.

The first three properties are requirements for the practical application of a hash function to message authentication. Sometimes as shown in Figure 5 (e) a secret code S is added to the message M before their concatenation is hashed (note that there is no encryption of the hashed component of the message here); it is important here that fourth property hold so that the opponent cannot decipher the secret code S. The fifth property prevents a forgery of a given message: an attacker intercepts a message plus its encrypted hash function, generates an unencrypted hash code from the message, and finally generates an alternate message with the same hash code. The sixth property makes the hash function resistant to an attack known as the birthday attack.

All the hash functions operate using the following general principles. The input message is viewed as a sequence of n-bit blocks. The input is processed

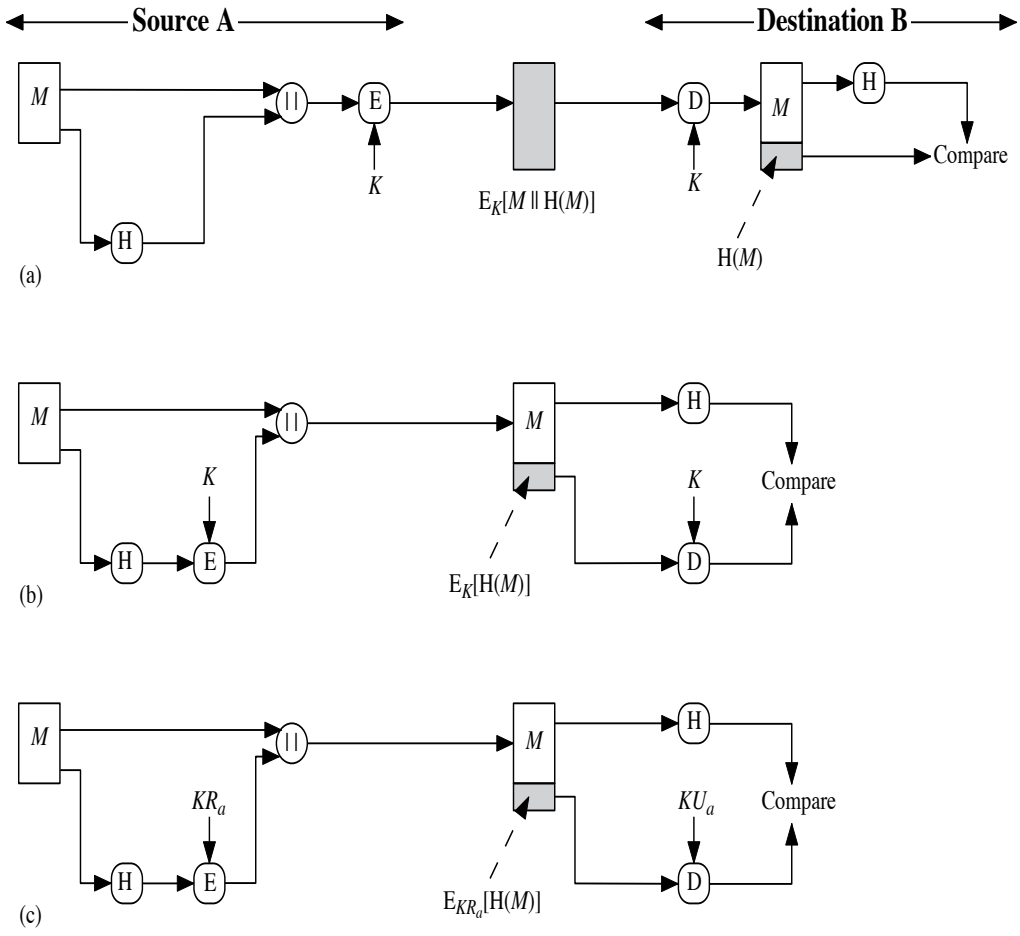


Figure 11.5 Basic Uses of Hash Function (page 1 of 2)

Figure 4: Basic use of a hash function

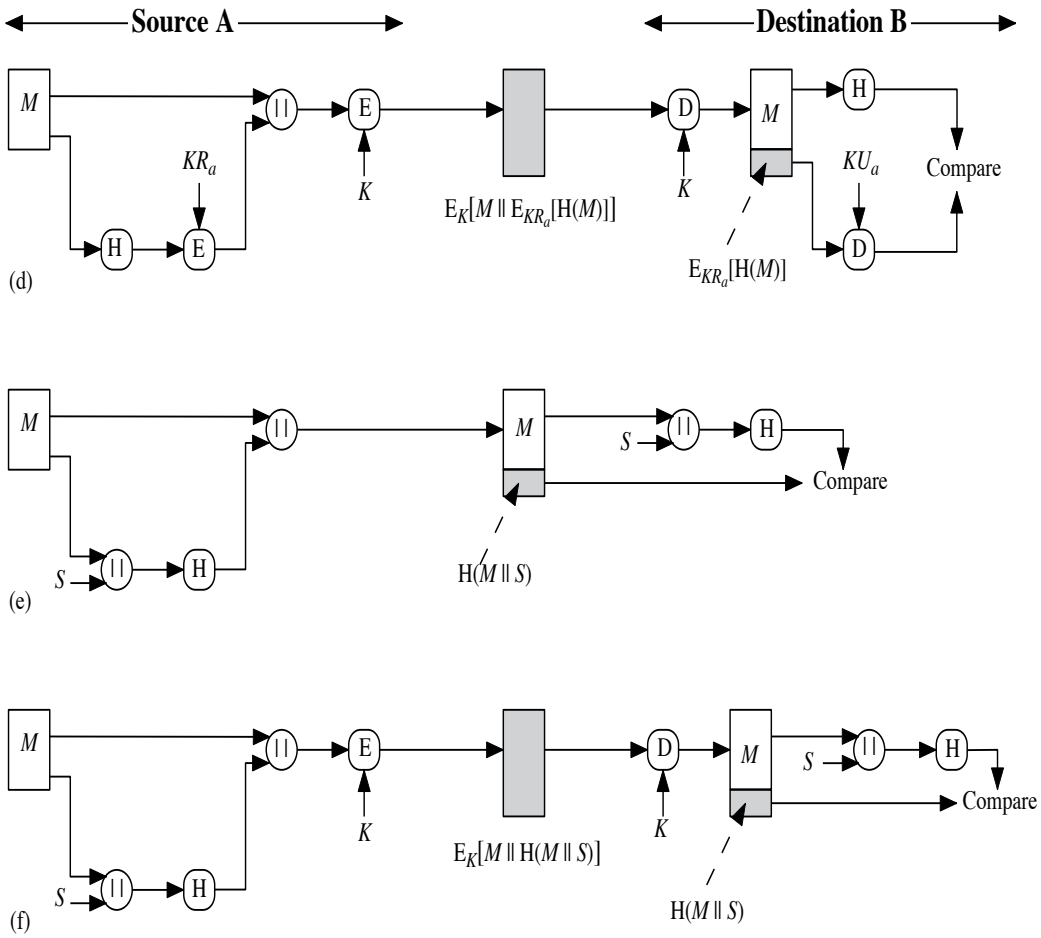


Figure 11.5 Basic Uses of Hash Function (page 2 of 2)

Figure 5: Basic use of a hash function (continued)

one block at a time in an iterative fashion to produce an n-bit hash function. One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block. This can be expressed as follows:

$$C_i \quad = \quad b_{i1} \oplus b_{i2} \oplus \ldots b_{im}$$

where

$$\begin{gathered} C_i = i\text{th bit of the hash code, } 1 \leq i \leq n \\ m = \text{number of } n \text{ blocks in the input} \\ b_{ij} = i\text{th bit in } j\text{th block} \\ \oplus = \text{XOR operation} \end{gathered}$$

To improve matters, one typically performs a one-bit circular shift, or rotation, on the hash value after each block is processed.

Digital signatures:

Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other. Digital signatures work best in public-key cryptography and an approach to generate these signatures was discussed in Lecture 9 in the context of public-key cryptography. I would also like to add here that the actual digital signature algorithm (DSA) (which you used in Lab 4 in ssh with public key authentication) is based on the difficulty of computing discrete logarithms (schemes originally presented by ElGamal and Schnorr); I have a link to the DSA algorithm on the course webpage. This is a competitor to the RSA public-key cryptosystem we discussed in Lecture 9, which was based on the difficulty of factoring large numbers. It is difficult/perhaps downright impossible to have digital signatures in symmetric-key cryptography without third party intervention. So, along with the distribution of the symmetric key, this remains one of the advantages of public-key cryptography over symmetric-key cryptography.

IP Security:

There are two views about network security; the first is one known as end-to-end (application layer security), where the two end-to-end processes (communicating parties) do the necessary encryption and decryption; any tampering done in between these two processes including within either operating system can then be detected. The trouble with this approach is that it requires changing all the applications to make them security aware. In this view, the next best approach is putting encryption between the application layer and the transport layer, making it still end-to-end, but not requiring the applications to be changed. Another trouble is that the IP headers in this approach cannot be encrypted, else the intermediate routers will not be able to see the plain IP headers for routing purposes. Thus, even if the eavesdropper cannot see the actual data, he/she can perform a traffic analysis based on the values in the IP header. The opposite view is that the network layer should do the necessary encryption/decryption (link level security). in this case each vulnerable communications link is equipped on both ends with an encrypting device. Thus, traffic over all communicating links is secured. Although this recourse requires a lot of

encryption devices in a large network (one between any pair of communicating hosts), its value is clear; moreover one can encrypt the entire packet including the IP header. One of the disadvantages is that message must be decrypted at each router to enable it to route the packet; thus the actual packet is vulnerable at each router. Several other implications of link encryption should also be noted: for this strategy to be effective, all the potential links in a path from source to destination must use link encryption. Each pair of routers/hosts that share a link must also share an unique key (we are talking about a symmetric-key cryptosystem here). Thus, many keys have to be distributed too. The general consensus is that link-level encryption despite its greater overhead is the better option. The end-to-end encryption and the link-level encryption schemes are summarized in figure 6. The result is a design called *IPsec (IP Security)*.

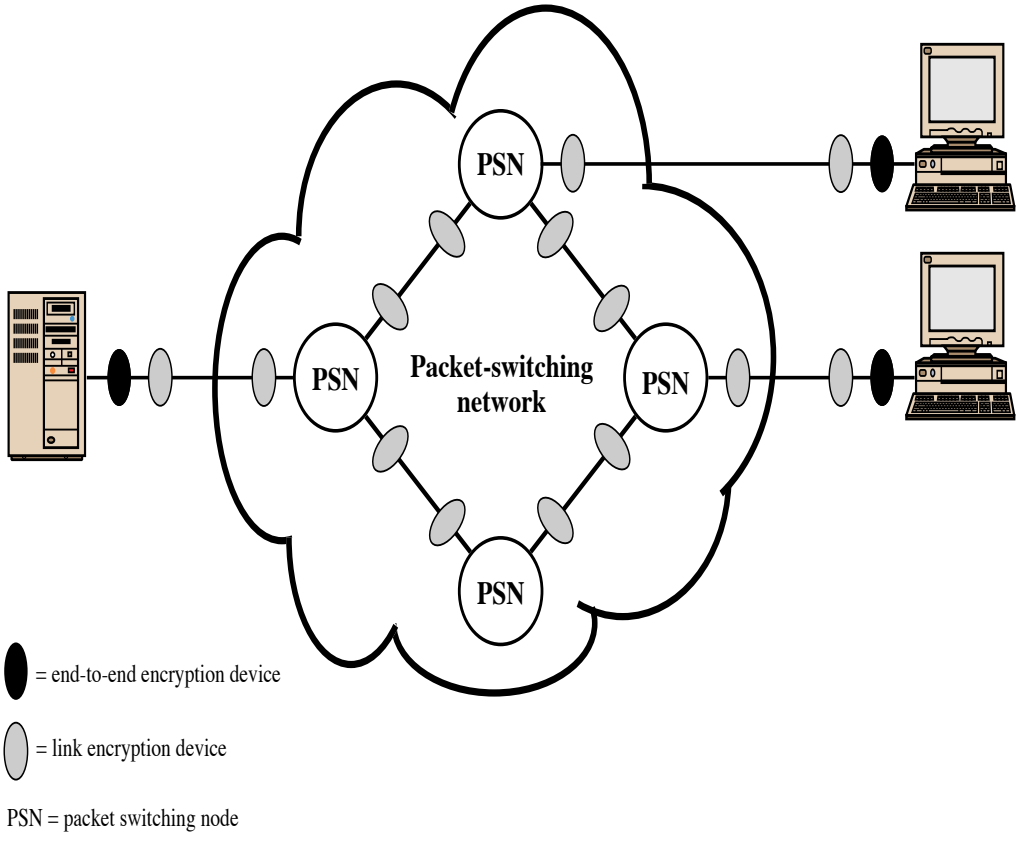


Figure 7.2 Encryption Across a Packet-Switching Network

Figure 6: Encryption across a packet switching network

The complete IPsec design is a framework for multiple services, algorithms and granularities. The reason for multiple services is that not everyone wants to pay the price for having all the services all the time. The major services are secrecy, data integrity, and protection from replay attacks (where an intruder replays an earlier conversation). All of these services are based on symmetric-key cryptography because high performance is crucial. Moreover, the reason for multiple algorithms is that the security algorithm (that is now thought be secure) may be broken in the future. By making IPsec algorithm independent, the framework can survive even if the algorithm does not. Finally, the reason for having multiple granularities is to make it possible to protect a single TCP connection, all traffic between a pair of hosts etc.

An important aspect of IPsec is that even though it is in the IP layer, it is connection oriented. However, this is not surprising, since the security protocol involves setting up a session key which is used for some period of time, so some sort of connection needs to be set up. A connection in the context of IPsec is called an SA (security association).

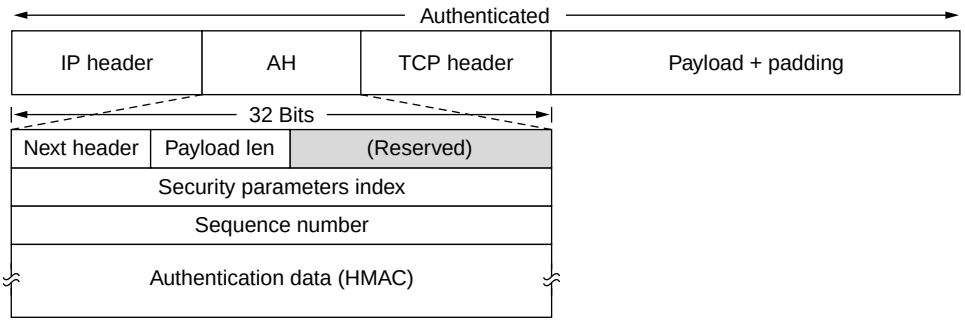
IPsec can be operated in two modes: a *transport* mode in which the IPsec header is added just after the IP header. The protocol field in the original IP header is changed to indicate that an IPsec header follows the normal IP header. The IPsec header contains security information, primarily the SA identifier, a new sequence number, and possibly an integrity check on the payload. In the *tunnel* mode, the entire IP packet, header and all, is encapsulated in the body of a new IP packet with a completely new IP header. Here is an example of how the tunnel mode IPsec operates: Host A on a network generates an IP packet with the destination address of host B on another network. This packet is routed from the originating host to a firewall (secure router) at the boundary of A's network. The packet is transmitted in the clear until it reaches this firewall. The firewall filters all outgoing packets to determine the need for IPsec processing. If the packet from A to B requires IPsec, the firewall performs this processing and encapsulates the resulting packet with an outer IP header. The source IP address of this outer IP packet is this firewall, and the destination address may be a firewall that forms the boundary to B's local network. This packet is now routed to B's firewall, with the intermediate routers examining only the outer IP header. At B's firewall, the outer IP header is stripped off, and the inner packet is delivered to B.

There are two protocols within IPsec: AH (Authentication Header) and ESP (Encapsulating Security Payload). AH only provides authentication, whereas ESP provides both authentication and encryption. The header employed in the AH protocol (in transport mode) is shown in Figure 7. As shown in this figure, the header consists of the following fields:

1. **Next header:** This stores the value the protocol field in the IP header had, before this was replaced with 51 to indicate that an AH header follows. In most cases, this value is 6 to indicate the code for TCP.

2. **Security parameters index:** This basically contains the shared session key used on the connection, besides other information.

3. **Sequence number field:** This numbers all the packets over the SA connection. Every packet gets a unique sequence number, even retransmissions. The purpose of this is to prevent replay attacks.

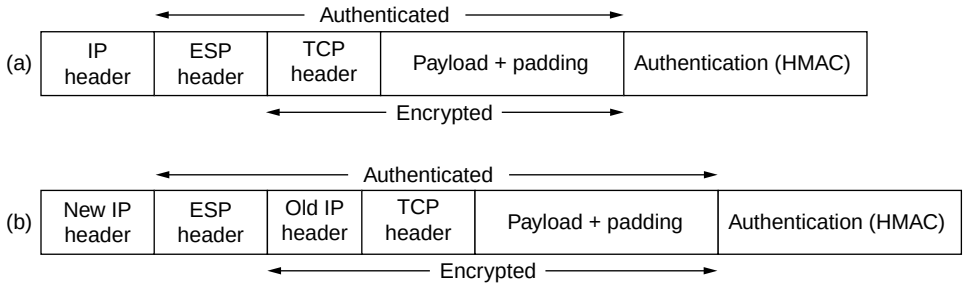

Authenticated
IP header
AH
TCP header
Payload + padding
32 Bits
Next header
Payload len
(Reserved)
Security parameters index
Sequence number
Authentication data (HMAC)

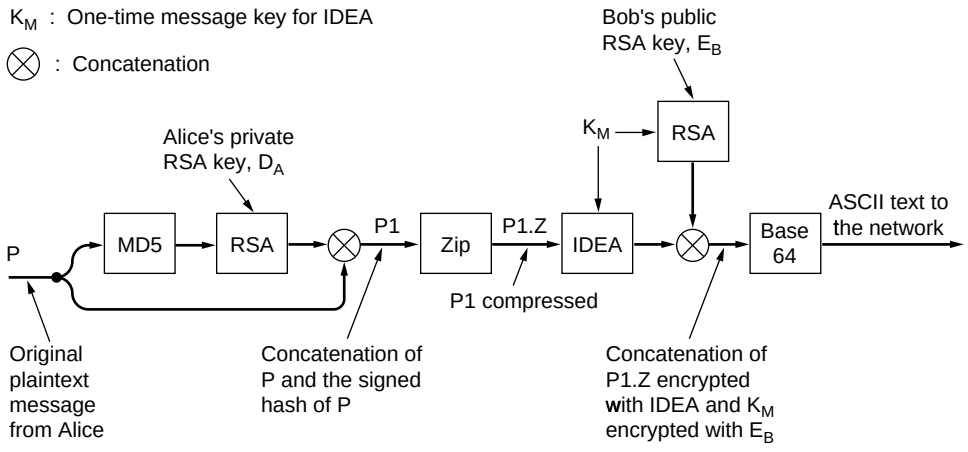
4. **Authentication data:** This contains a variable length field that verifies of the payload data. This signature is established by computing a hash over the packet plus the shared key using the HMAC (Hashed Message Authentication Code). One noteworthy feature of AH is that the integrity covers some of the fields in the IP header, namely, those that do not change as the packet moves from router to router (some entries like TTL change and so these fields cannot be encrypted). Moreover, the source IP address is included in the check making it impossible for an intruder to falsify the origin of a packet.

The alternative IPsec header is the ESP. The ESP headers in transport and tunnelling modes are shown in Figure 8. One difference between this and the AH header is that HMAC integrity check comes after the payload. Moreover this integrity check does not involve any of the fields in the IP header (in the transport mode), and none of the fields in the new IP header (in the tunnelling mode).

Email security:

We will just mention PGP (Pretty Good Privacy) out here. This was essentially the brainchild of one person, Phil Zimmermann. PGP is a complete email security package that provides privacy, authentication, digital signatures, and compression. Furthermore, the complete package, including all source code, is distributed free of charge via the internet. PGP encrypts data by using a block cipher called IDEA (International Data Encryption Algorithm), which uses 128-bit keys. Figure 9 describes PGP in operation for sending a message. Here, Alice wants to send a plaintext message P to Bob in a secure way. Both Alice and Bob have private (D_A, D_B) and public (E_A, E_B) RSA keys respectively. Alice starts out by invoking the PGP program on her computer. PGP first hashes her message P, using MD5 and then encrypts the resulting hash using her private key R_A. When Bob eventually gets the message, he can decrypt the hash using Alice's public key and verify that the hash is correct. The encrypted hash and the original message are now concatenated into a single message, $P1$, and compressed using the ZIP program. Let us call the output of this message $P1.Z$. Next, PGP prompts Alice for some random imput. Both the content and the typing speed are used to generate a 128-bit IDEA message key, K_M. K_M is now used to encrypt $P1.Z$ with IDEA in cipher feedback mode. In addition, K_M is encrypted using Bob's public key, E_B. These two components are then concatenated and converted to base64. The resulting message then contains only letters, digits and other ASCII characters, which means that it can be put in the RFC 822 (email) body and be expected to arrive unmodified. When Bob gets the message, he reverses the base64 encoding and decrypts the IDEA key using the private RSA key. Using this key, he decrypts the message to get $P1.Z$. After decompressing it, Bob separates the plaintext from the encrypted hash, and decrypts the hash using Alice's public key. If the plaintext hash agress with his own MD5 computation, he knows that P is the correct message and that it came from Alice (the hash ensures authentication while Alice's digital signature confirms that it indeed came from her). It is worth noting here that PGP is a mixture of symmetric and public-key (RSA) cryptography. However, RSA is used only in two places here: to encrypt the 128 bit MD5 hash and to encrypt the 128-bit IDEA key. Since RSA has only to encrypt 256 bits in all


Authenticated
(a)
IP header
ESP header
TCP header
Payload + padding
Authentication (HMAC)
Encrypted
Authenticated
(b)
New IP header
ESP header
Old IP header
TCP header
Payload + padding
Authentication (HMAC)
Encrypted


K_M : One-time message key for IDEA
: Concatenation
Bob's public RSA key, E_B
K_M
RSA
Alice's private RSA key, D_A
P
MD5
RSA
P1
Zip
P1.Z
IDEA
Base 64
ASCII text to the network
P1 compressed
Original plaintext message from Alice
Concatenation of P and the signed hash of P
Concatenation of P1.Z encrypted with IDEA and K_M encrypted with E_B

here, the encryption and the subsequent decryption can be performed quickly. The length of the RSA keys depends on the amount of security required: a commercial application uses an 512 bit RSA key while a military application might typically use a 1024 bit key. Finally, the encryption is done over the header and the body of the email message. Other encryption schemes for email include PEM (Privacy Enhanced Mail) and S/MIME (Secure/MIME) (see Lecture 7 for a discussion of the MIME).

Supplementary Reading:

1. Chapters 11 and 13 of Stallings [1] for a discussion on message authentication and digital signatures.
2. Chapter 16 of Stallings and Section 8.6.1 of Tanenbaum [2] for a discussion of IP security.
3. Chapter 15 of Stallings and Section 8.8 of Tanenbaum for a discussion of email security.

References

[1] W. Stallings, *Cryptography and Network Security*, 3rd edition, Prentice Hall, 2003.

[2] A.S. Tanenbaum, *Computer Networks*, 4th edition, Prentice Hall, 2003.