Lecture Notes

Introduction to Cryptography

Raj Bridgelall, PhD

Table of Contents

Cryptography and Network Security

1 Introduction

Cryptography is Greek for "hidden writing." In computer based cryptography, it is the art of *ciphering* an easily understood message or "*plain text*" into one that cannot be easily *deciphered* (Mel and Baker 2001). The basic components for a cryptographic system are a *ciphering engine*, a *key exchange* mechanism, and a random number generator. A reversible ciphering engine will encrypt or decrypt the plain text message using the same key, which is a secret known only to the parties involved. This is called symmetric key cryptography. It is different from public key cryptography whereby one key is used for encryption while another mathematically related key is used to decipher the message. Public key cryptography is also called asymmetric key cryptography and often involves a *hashing* function. Ciphering engines are either *block ciphers* which encrypt blocks of text at a time, or *stream ciphers*, which produce an output bit stream in response to an input bit stream.

Cryptography is essential for maintaining the *confidentiality*, *authenticity*, and *integrity* of messages that are communicated over untrustworthy channels. *Confidentiality* is the assurance that only the owners of the keys can access the data. *Authenticity* is the assurance that the originator of the message is not an imposter. *Integrity* is the assurance that data has not been altered while in transit.

All ciphering methods are based on the principles of *diffusion* and *confusion*, which are terms coined by Claude Shannon. *Diffusion* is the technique of transposing and substituting characters or bits. The intent is to disperse the statistical nature of the encrypted message or *cipher text*, and thereby hide its relationship with the plain text. Alternatively, *confusion* is the cryptographic principle of hiding the relationship between the cipher text and the secret key. Given a key length in bits, a s*trong* cryptographic method has many possible secret keys such that a brute force search for the secret key will be infeasible.

2 Symmetric Key Cryptography

In symmetric key cryptography the same key, which is a secret, both encrypts and subsequently decrypts a message. This type of cryptography is an attractive means for secure communications among low cost computing devices such as sensors because it involves less computationally expensive ciphering methods than public key cryptography. However, its strength ultimately depends on the robustness of a system for distributing secret keys to the network participants.

Data Encryption Standard (DES) was one of the most popular symmetric encryption algorithms, which was a published standard since 1977. However in 1999, a team from the Electronic Frontier Foundation managed to break the DES encryption in less than 24 hours. Around the time of this event, the National Institute of Standards and Technology (NIST) requested algorithm submissions for a new federal Advanced Encryption Standard (AES.) Five contenders made it to the last round of selection. In October 2000 NIST selected the *Rijndael* algorithm, which was developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen. While this standard was being ratified, Walter Tuchman of IBM proposed Triple DES, which was an effort to improve the security of the DES algorithm. With triple DES, the DES algorithm is applied three times with two different secret keys.

2.1 Symmetric Key Distribution

A trusted third party or *key distribution center* (KDC) is often used to distribute copies of the keys. Key distribution can become a significant problem when the number of keys to distribute grows exponentially with the addition of as new network clients. Therefore, the KDC can become the bottleneck of information exchange and is also a likely target for hackers. One way around this is to use public key encryption as a secure means of exchanging secret keys so that subsequent communications will use the less computationally expensive symmetric key methods.

2.2 Key Agreement

The Diffie-Hellman (DH) algorithm developed by Whitfield Diffie and Martin Hellman of Stanford in 1976 is a popular method of key agreement. That is, the common secret key is derived, rather than distributed by another system, via plain text communications across an untrustworthy network. Correspondents agree on a large prime generator ***g*** and a large prime number modulus ***n***, each of which can be made public. The generator, ***g*** must have the property that its exponentiation with a random number from the selected number space will result in a large number of unique outcomes. The following table summarizes the DH algorithm for secret key agreement:

Correspondent-A	Correspondent-B
Publicly known: $\{g, n\}$	
Generate random number A (secret)	Generate random number B (secret)
Calculate $X = g^A$ MOD n	Calculate $Y = g^B$ MOD n
Transmit X	Transmit Y
Calculate Key = Y^A MOD n = $(g^B)^A$ MOD n = g^{BA} MOD n	Calculate Key = X^B MOD n = $(g^A)^B = g^{AB}$ MOD n
Shared Key = g^{AB} MOD n	

For a key that is 1024 bits long, an eavesdropper will need to compute the discrete log of both X and Y many times in order to discover the random numbers A and B. The security of this method relies on the fact that the computational intensity of the discrete log problem is expensive because it will take existing computers many years to compute. However, as computing technologies evolve, the computational time will continuously decrease.

2.3 Key Exchange

Even though the DH algorithm allows each party to create a shared secret key with publicly exchanged parameters it does not provide authentication. That is, neither correspondent can be assured that the other is genuine. Therefore, public key techniques are used to first exchange keys that correspondents can subsequently utilize.

3 Public Key Cryptography

Unlike secret key cryptography, public key cryptography provides a better way to publicly distribute keys while keeping the secret or private key safely guarded. Public key cryptography is also known as asymmetric key cryptography because one key is used for encryption while another mathematically related key is used for decryption. Asymmetric or public key encryption methods typically use the concept of modular inverses to create public/private key pairs.

Modular inverses are a pair of numbers in modular arithmetic that when multiplied together yield unity. For example, the numbers 3 and 7 are modular inverses in modulo 10 arithmetic because

$$(3 \times 7) \text{ MOD } 10 = 1.$$

To encrypt a message, we essentially multiply numeric equivalents of it with the first number, transmit the result as the encrypted message, and then multiply the received encryption with the second number so as to recover the original message. We can view the overall process as multiplying the message by unity whereby the first number is the public key, and it's inverse as the private key or vice versa.

The success of public key cryptography for worldwide electronic commerce depends on the establishment of a trusted third party that will authenticate and distribute public keys. This will prevent imposters from claiming any public key as their own without first presenting proof of authenticity. Even though public key cryptography can provide all of the needed security functions while secret key cryptography cannot, it is not as practical to implement across all platforms primarily because of its high computational complexity. Therefore, secret key techniques are used for many networking applications, while public key techniques are used when necessary to facilitate authentication and secret key exchanges.

RSA and Pretty Good Privacy (PGP), which was created in 1991 by Philip Zimmerman are probably the two most well-known public key cryptographic systems. Figure 1 illustrates the basic operation of a public key based message exchange.

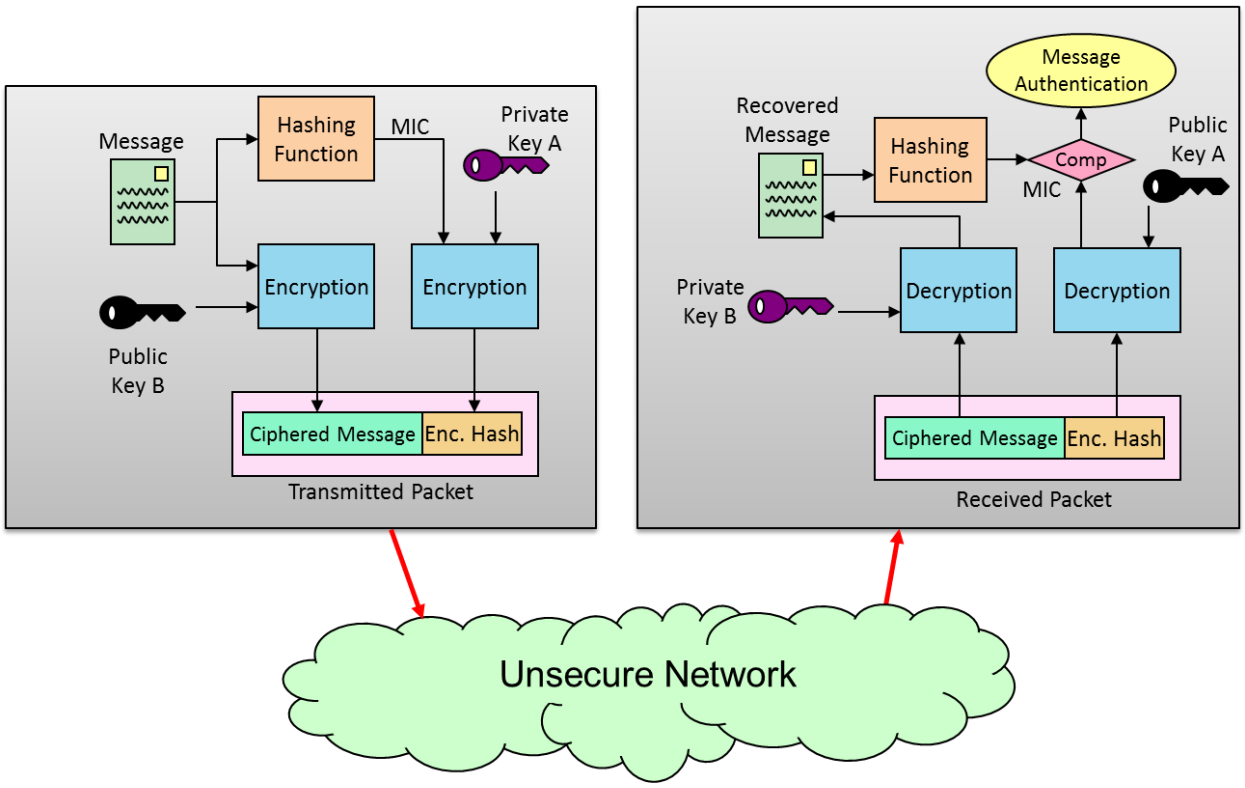

Figure 1: Secure communications over an untrusted network using public key methods.

The sender first obtains the recipient's public key from a trusted third party that vies for its authenticity. Section 3.4 describes public key distribution and digital certificates. With the assurance that the public key is indeed assigned to the intended recipient, the sender encrypts the

plain text message with the recipient's public key B and transmits the encrypted message. The recipient decrypts the ciphered message with its securely held private key B. However, this process only provides confidentiality, which is the assurance that no one else could have deciphered the message in transit, other than the recipient. However, the message itself could have been intercepted, substituted, re-encrypted with the recipient's public key, and delivered instead of the original message. Therefore, in order to authenticate the message, the sender adds a digital signature to the transmitted packet by encrypting a highly compressed form (a hash) of the original message with the sender's private key. Upon receiving the message, the recipient also decrypts the hash with the sender's public key and compares the result with its own hash of the received message, thereby validating the authenticity of the message. Section 3.3 describes hashing and digital signatures in greater detail.

3.1 RSA Public Key Cryptography

RSA is named after its inventors, Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman of MIT. It is a popular public key encryption algorithm that many cryptographers scrutinize. Its strength is based on the difficulty of factoring a very large number into two large prime numbers. The algorithm, developed from a basic theory by Pierre de Fermat in the 1600's, specifies

$$m^{(p-1)} \text{ MOD } p = 1$$

for any prime number $\boldsymbol{p}$ and positive number $\boldsymbol{m}$ that is less than $\boldsymbol{p}$. Later, Leonard Euler extended Fermat's theorem in the 1700's by proving that the expression was also true for prime products. That is, when replacing $\boldsymbol{p}$ by the product of two primes $\boldsymbol{p_a}$ and $\boldsymbol{p_b}$, the expression

$$m^{(p_a-1)(p_b-1)} \text{ MOD } p_a p_b = 1$$

is also true for base $\boldsymbol{m}$ and modulus ($\boldsymbol{p_a p_b}$) relatively prime, which means having no common factors other than unity. Multiplying both sides by $\boldsymbol{m}$ yields

$$m^{(p_a-1)(p_b-1)+1} \text{ MOD } p_a p_b = m$$

Letting

$$(p_a-1)(p_b-1) = \phi(n)$$

where $n = p_a p_b$, any message $\boldsymbol{m}$ raised to the power $\phi(n)+1$ in modulo $\boldsymbol{n}$ arithmetic gives back the original message $\boldsymbol{m}$ or

$$m^{\phi(n)+1} \text{ MOD } n = m$$

The basic idea behind asymmetric key cryptography is to generate two numbers (keys) $\boldsymbol{K_p}$ and $\boldsymbol{K_q}$ such that their products will be equal to $\phi(n)+1$,

$$K_p K_q = \phi(n)+1$$

This problem is the same problem as finding $\boldsymbol{K_p}$ and $\boldsymbol{K_q}$ such that their product in modulo $\phi(n)$ arithmetic is unity. That is,

$$K_p K_q = 1 \text{ MOD } \phi(n)$$

Note that $\boldsymbol{K_p}$ and $\boldsymbol{K_q}$ are mathematical inverses of each other in modulo $\phi(n)$ arithmetic. Picking one of the numbers, for example $\boldsymbol{K_p}$, such that it is relatively prime with both (p_a-1) and (p_b-1) allows for the *Extended Euclidean algorithm* then generate $\boldsymbol{K_q}$ (Mel and Baker 2001). Therefore,

if the public key is {K_p,n} then the private key would be K_q.

Encrypting the message requires an exponentiation of it with the public key within the modulus n. Decrypting the message similarly requires an exponentiation of the encrypted message with the private key within the modulus n to get the original message m as follows:

Encryption	Decryption
$(m)^{Kp}$ MOD n	$(m^{Kp})^{Kq}$ MOD n = $m^{\phi(n)+1}$ MOD n = m

An observer knowing K_p, n, and the cipher text $(m)^{Kp}$ MOD n cannot solve for the message m within a few lifespan by using the fastest computer currently available. The only known practical method for computing m is to first know the secret key K_p. If p_a and p_b are known, then the secret key K_q can be derived. However, finding p_a and p_b will involve brute-force factoring of a very large (1024-bit) number, and this will also take several lifespans with the fastest computer currently available.

3.2 Elliptic Curve Cryptography

Since the RSA key lengths have become long (1024 bits), new techniques such as Elliptic Curve Cryptography (ECC) have been invented. ECC is based on sets of predefined mathematical rules for translating coordinates on an elliptic curve in modulo arithmetic. The elliptic curve is defined as

$$y^2 \text{ MOD } p = (x^3 + ax + b) \text{ MOD } p$$

The public key includes the curve parameters (p, a, b), a point on the curve defined by an (x,y) coordinate pair P, and another point on the curve Q that is a "special" translation of the point P by some secret amount d. This secret amount d is the private key. The "special" translation is such that

$$d \otimes P = Q$$

This special translation operation is a one way function. That is, one cannot solve for the secret translation scalar d simply by knowing the input and output coordinates P and Q respectively. Therefore, correspondents at each end of the untrustworthy communications channel can easily derive a shared secret key by combining their private key with the correspondent's public key as illustrated in the following table.

Correspondent-A	Correspondent-B
Public Key A = {p, a, b, P, Q_a}	Public Key B = {p, a, b, P, Q_b}
Private Key A = {d_a}	Private Key B = {d_b}
Transmit $Q_a = d_a \otimes P$	Transmit $Q_b = d_b \otimes P$
Shared Key = $d_a \otimes Q_b = d_a \otimes (d_b \otimes P)$	Shared Key = $d_b \otimes Q_a = d_b \otimes (d_a \otimes P)$

This form of key exchange works because the correspondent's public key also contains the private key in cryptic form. EC keys require about 20 bytes of storage compared with about 256 bytes for RSA keys. Researchers found that a 170 bit EC key length will give approximately the same level of security as a 1024 bit RSA key. Smaller key sizes provide more flexibility for storage in low-end computing devices such as smart cards.

3.3 Digital Signatures

Encrypting the plain text or a compressed version of it with the private key rather than the public key creates a digital signature. The recipient verifies the authenticity of the signature by applying the associated public key as we illustrated in Figure 1. The U.S. Federal Information Processing Standard (FIPS) adopted Digital Signature Algorithm (DSA) in the 1990s, even though RSA could have also been used. DSA is used only for digital signatures. Unlike RSA, DSA cannot be used with a public key to decipher the original plain text message. In fact, it requires the original plain text in order to verify the signature. RSA can verify signatures much faster than DSA. However, DSA can create signatures faster than RSA by using pre-computed values. Therefore, it is likely to be used for low complexity computing devices such as smart cards where the verification can be performed on servers which generally has computing capability.

3.3.1 Hashing

When creating a digital signature with the private key, the original plain text message itself is not used directly. This is because it is not sensible to encrypt the plain text itself with a private key since it can be decrypted with a public key, as in RSA. Therefore, practical systems sign an irreversible and highly compressed version of the message so that the recipient can verify the signature as shown in Figure 1. Hashing is such a one way function that can produce an irreversible and highly compressed version of the message. When applied to a message, the hashing function produces a sequence of numbers known as the "digital fingerprint" since it is unique and repeatable. The hash value can also be viewed as a form of cryptographic checksum that must change if the input text changes. The hashing function output is also called a Message Integrity Code (MIC) or a Modification Detection Code (MDC.) MICs have a one-to-one mapping with the message. That is, no two messages will produce the same code. This is called *weak collision resistance*. MICs also have the reciprocal mapping property whereby each code uniquely represents only one message. This is called *strong collision resistance*. The subtle difference is that weak collision resistance prevents finding ***any*** two different messages that will result in the same code while strong collision resistance prevents finding two different source messages in the message space when given a specific code. Message Digest 5 (MD5), created by Ron Rivest, is a popular MIC hashing function. NIST enhanced the previous version of it (MD4) to produce Secure Hash Algorithm (SHA-1). NIST then released the SHA-2 family in 2001, which includes an update to SHA-256 in March 2012 used by Bitcoin.

A Message Authentication Code (MAC) is another form of message compression and coding that utilizes a secret key. Unlike a MIC, the correspondents using a MAC must share a secret key in order to create and then to later verify the message authenticity. Most MACs are made with secret key ciphers that are repeatedly applied to intermediate compressions of the message. For example, Data Authentication Algorithm (DAA), the FIPS standard since 1985, is really a combination of DES and a compression method. MACs execute much more slowly than MICs. Therefore in 1996, cryptographers proposed the combination of MIC hashing with secret keys and called it HMAC.

3.3.2 Non-repudiation

Non-repudiation is the assurance that the sender cannot deny having originated the message because only one who has possession of the secret key could have constructed the digital signature in question. In addition to confidentiality, authentication, and integrity, public key cryptography also provides non-repudiation. Secret key cryptography cannot provide non-repudiation.

3.4 Digital Certificates

A correspondent cannot be assured that a public key does indeed belong to the claimed party. Therefore, the public key must be distributed by a trusted source that is willing to certify its authenticity. A digital certificate is the method of choice for public key distribution. The third party authenticates a public key by attaching a digitally signed hash of the plain text message. The recipient verifies the message authenticity by decrypting the hash (verifying the signature) with the third party's public key and comparing the revealed hash with that resulting from hashing the received plain text. The plain text contains the public key as an attachment and it also describes attributes of the public key owner.

Public Key Infrastructure (PKI) is a digital certificate administrative framework for public key delivery. Well-known PKI standards are X.509 and Pretty Good Privacy (PGP.)

3.4.1 PKI Based on X.509

A root Certificate Authority (CA) in X.509 PKI is the first trusted source for authentic digital certificates. The CA can also subcontract its responsibilities to trusted Registration Authorities (RA) and hence form a trusted tree-structure for certificate distribution. Amongst other information, the digital certificate consists of plain text identifying the issuer, the subject unique identity (also called a distinguished name), the subject's public key, signature method used, and a certificate expiration date. Software vendors typically pay a CA to distribute public keys and so already include CA public keys in their configuration parameters. These are called root certificates because they are self-signed by the CA and contains the CA's trusted public key. Subjects need the CA's public key to verify the CA's digital signature on a certificate that it issues.

The CA also issues a challenge message in order to be sure that the subject does indeed possess the associated private key. In doing so, the CA encrypts a random message with the subject's public key and sends it. The subject decrypts the message with the private key and responds with the plain text message. The CA verifies that the response is the same as the challenge plain text. In addition to managing public keys, the CA also informs users when certificates have been pre-maturely revoked and puts the certificates on a certificate revocation list (CRL.)

3.4.2 PKI Based on PGP

Philip Zimmermann developed PGP because he was not comfortable with the bureaucratic style architecture of X.509. Rather than using a CA, users in a PKI based on PGP issue and manage their own digital certificates as well as sign and forward those of other trusted correspondents. This also means that unlike X.509, each certificate can be signed by more than one subject thereby adding higher levels of thrust. Therefore, rather than a centralized thrust model, PGP utilizes a distributed thrust model. PGP is based on RSA public key cryptography but it also supports Diffie-Hellman.

4 Cryptographic Protocols

4.1 SSL

Secure Socket Layer (SSL) is a data communications protocol primarily utilized by Internet browsers to facilitate secret key exchanges and provide authentication, confidentiality, and message integrity. It operates above TCP/IP. The IETF standardize it as the Transport Layer

Security (TLS) in 1998. Microsoft was the first to incorporate TLS into its browser.

SSL/TSL completes a secure transaction by first negotiating which cryptographic algorithms. It then exchanges public keys via digital certificates, generates shared secret keys, authenticates its clients, and finally bulk encrypts the transmitted data for confidentiality and integrity. SSL/TSL uses six secret keys. Both corresponding parties independently generate these from a *pre-master secret*, which is a random number that the initiating party generates and sends to the correspondent in encrypted form. Three keys are used in each communication direction for each bulk encryption, message integrity check, and cipher engine initialization.

4.2 IPSec

In simplest terms, IPSec can authenticate data entering and encrypt data leaving a communications device. It is, however, a complex protocol and several books have been dedicated to the subject (Doraswamy and Harkins 2003). Functionally, IPSec consists or two main parts, secret key exchanges using Internet Key Exchange (IKE) by default, and bulk encryption. The IKE manages authentication and key exchanges while the bulk encryption process provides confidentiality and message integrity. The IKE is a two-phase protocol whereby the first phase openly negotiates parameters to protect the second phase, which in turn negotiates parameters in secrecy for the bulk encryption part. Once the parties agree on cryptographic parameters, the bulk encryption part uses either of two protocols and either of two modes to provide data packet assurances. Devices that use IPSec must also comply with policies that the network manager defines in a Security Policy Database (SPD.) Figure 2 illustrates the overall organization of the IPSec protocol.

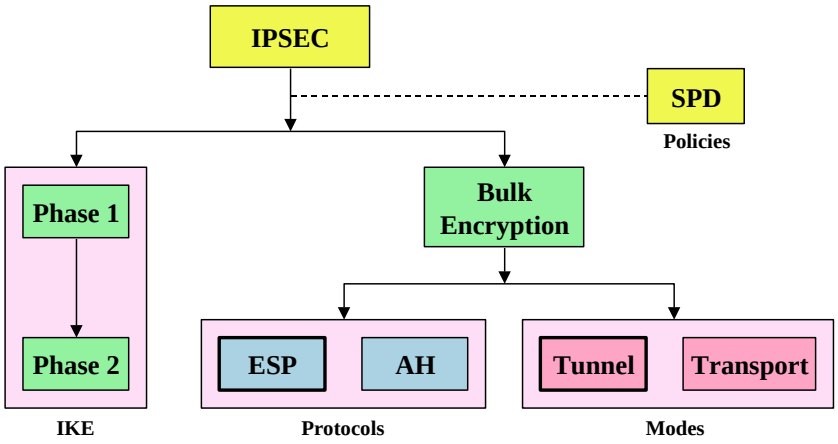

Figure 2: IPSec protocol architecture.

Unlike SSL/TSL, IKE involves two secret parameter exchanges in two phases so as to add greater security and speed. These separate secret parameter exchanges or phases are also called Security Associations (SA) because each result in the generation of many secret keys. Phase one (also known as IKE-SA) key agreement uses Diffie-Hellman to initially establish an authenticated and secure channel between the IPSec parties. Each party then derives three separate keys for symmetric cryptographic exchanges. Phase two (also known as IPSec-SA) cryptographic parameters and secret keys are then exchanged confidentially using phase one secret keys. The phase two parameters are used for the bulk encryption process and different secret keys are generated for different applications that IPSec services. These are derived much faster than phase

one parameters because a secure and authenticated channel has already been established in phase one via more lengthy public key operations. For added security, IPSec automatically updates phase two secret keys once they have aged beyond a pre-determined *lifetime*. Phase one keys are updated less often because they involve more computationally intensive public key methods. IPSec also requires a different SA for inbound and outbound messages of the bulk encryption part of the protocol.

Once the key exchange phases are complete, IPSec is ready to encrypt packets via one of two *protocols* and one of two *modes*. The protocol attribute controls the level of assurance, depending on whether, or not one or both message confidentiality and integrity is required. The mode attribute controls how much of the data packet will be protected by the protocol chosen.

The two protocol choices are Encapsulating Security Protection (ESP) and Authentication Header (AH.) The mode choices are *tunnel* and *transport*. Therefore, there are four possible combinations of protocols and modes. ESP provides both message integrity and confidentiality whereas AH provides only message integrity. The ESP protocol encrypts both the upper layer payload data and the source/destination IP addresses in tunnel mode but only the payload data in transport mode. ESP also includes a signed hash (HMAC) of its own header and encrypted portion of the packet. The encrypted data is also often padded in order to fix the length of the packets. Fixed length data packets make it more difficult for eavesdropper to analyze the statistical nature of the traffic flow between hosts. The AH protocol does not encrypt any of the packet. It provides message integrity by including a signed hash of the payload data in transport mode and a signed hash of both the payload data and IP addresses in tunnel mode.

The ESP and AH headers contain a Security Parameters Index (SPI), sequence numbers, and anti-replay attributes. The SPI helps the receiver to quickly locate the governing SA in the database. The sequence numbers and anti-replay attributes are used to detect packets that hackers may copy and replay in an attempt to overload the IPSec protocol and possibly cause the host application to 'hang-up.' The sender numbers each packet and the receiver looks within a pre-determined window size for indication of duplicate sequence numbers. The receiver can ignore packets with duplicate sequence numbers and, thereby, prevent hackers from flooding the system with old packets.

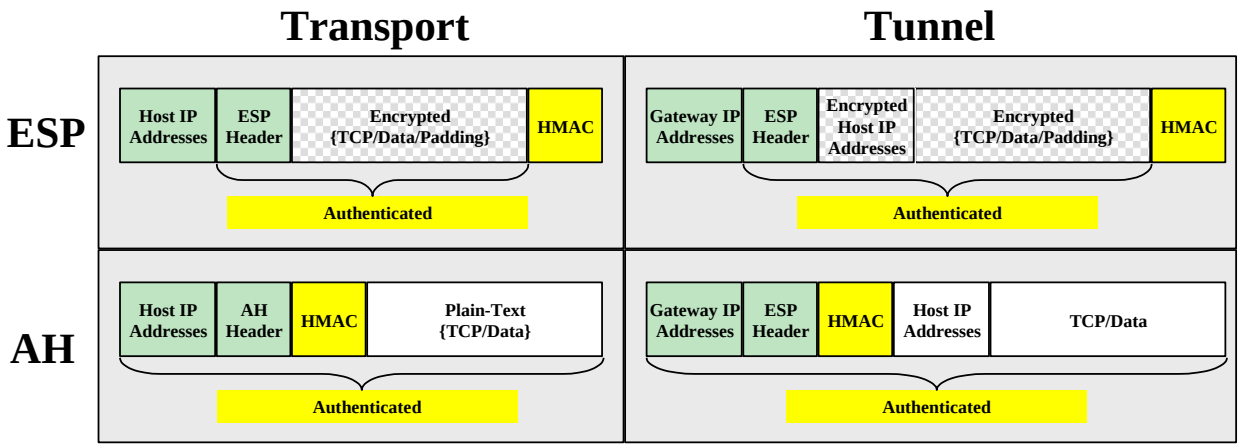

Figure 3: Assurances from the four possible protocol/mode combinations of IPSec.

A Security Policy Database (SPD) stores a set of rules that every IPSec enabled host must follow.

These set of rules limit the degree with which its host can correspond with other computers on the Internet.

4.3 Virtual Private Networks

A Virtual Private Network (VPN) is a secure communications channel that is established between two networking devices (e.g. routers) on a transit or public network such that the two entities can communicate in privacy. Before VPN technology became widespread, corporations leased dedicated channels on private networks in order to communicate in confidence. VPN's are also used for establishing a secure point-to-point communications channel between the server of a remote client (e.g. a mobile professional) and a corporate server.

Point to Point Tunneling Protocol (PPTP), Layer 2 Tunneling Protocol (L2TP), and IPSec are examples of technologies used for establishing Virtual Private Networks (VPN's.) Most IPSec based VPN products utilize the ESP protocol with tunnel mode because it is the most robust combination. Tunnel mode must be used when proxy gateways or servers (e.g. firewalls) are established between the correspondent hosts. Tunnel mode conceals both the packet and the host address headers. The proxy servers decrypt the packets before forwarding them to the addressed hosts.

4.4 Point-to-Point Protocol

PPP was designed to send data across dedicated point-to-point connections between a dial-up client and a Network Access Server (NAS.) It is a four-phase protocol that encapsulates IP (also IPX or NetBEUI) packets. PPP is often used for MODEM dial up connections over a serial communications link. It allows two machines on a point-to-point communications channel to negotiate the network layer protocols that will be used during the session. The phase one portion of the protocol establishes, maintains, and eventually tears-down the physical connection. This is also the phase where authentication protocols are selected for the next three phases of communications. The decision to use encryption and compression are also determined in this phase but the specific selections are made in the last phase.

Phase two involves client and server authentication and utilizes Password Authentication Protocol (PAP), or Challenge Handshake Authentication Protocol (CHAP), or Microsoft Challenge Handshake Authentication Protocol (MSCHAP). PAP communicates user name and password in plain text and does not provide any security. CHAP utilizes a challenge and response mechanism whereby the client sends an MD5 hash made with the password, and containing the password, a random challenge string, and the session identification. The server verifies the MD5 hash with the password linked to the user name and thereby completes the authentication phase.

The third phase of PPP is an *optional* callback control mechanism whereby the NAS will disconnect then call back the client at the specified phone number. The fourth and last phase invokes the various control protocols selected in the previous phases. In addition, both data compression and encryption protocols will be selected. Finally, data transfer begins with an encapsulation of the packets with PPP headers.

5 References

Doraswamy, Naganand, and Dan Harkins. 2003. *IPSec: The New Security Standard for the Internet, Intranets, and Virtual Private Networks.* 2nd. NJ: Prentice Hall.

Mel, H. X., and Doris Baker. 2001. *Cryptography Decrypted.* Addison-Wesley.

6 List of Acronyms

Acronym	Meaning
802.11a	A section of the IEEE 802.11 standard that specifies WLAN networks for speeds up to 54 MBPS using OFDM for channel access and QPSK or QAM for carrier modulation. In general, the 802.11 standard specifies channel sharing via CSMA/CA mechanisms that are managed by the MAC.
802.11b	A section of the IEEE 802.11 standard that specifies WLAN networks for speeds up to 11 MBPS using DSSS for channel access and QPSK for carrier modulation.
API	Applications programmer interface.
CSMA/CA	Carrier sense multiple access/collision avoidance.
HTTP	Hypertext transfer protocol.
IP	Internetworking protocol.
IPsec	IP Security – a set of protocols for secure exchange of Internet packets.
ITU	International telecommunications union.
LLC	Logical link control – upper sublayer of the data link layer as defined by IEEE 802.2.
MAC	Media access control.
MSC	Mobile switching center.
NAS	Network Authentication Server
PHY	Physical (layer.)
PPP	Point to point protocol.
RADIUS	Remote Authentication Dial-In User Service.
RTP	Real-time protocol.
TCP	Transmission control protocol.
UDP	User datagram protocol.
VoIP	Voice-over-Internetworking protocol.
VPN	Virtual private network.
WLAN	Wireless local area network.
WWAN	Wireless wide area network.