UNIT - I
PART A

1. Give the types of Attack.

Passive Attacks

The main goal of a passive attack is to obtain unauthorized access to the information. For example, actions such as intercepting and eavesdropping on the communication channel can be regarded as passive attack. These actions are passive in nature, as they neither affect information nor disrupt the communication channel. A passive attack is often seen as *stealing* information. The only difference in stealing physical goods and stealing information is that theft of data still leaves the owner in possession of that data. Passive information attack is thus more dangerous than stealing of goods, as information theft may go unnoticed by the owner.

Active Attacks

An active attack involves changing the information in some way by conducting some process on the information. For example,

- Modifying the information in an unauthorized manner.
- Initiating unintended or unauthorized transmission of information.
- Alteration of authentication data such as originator name or timestamp associated with information
- Unauthorized deletion of data.
- Denial of access to information for legitimate users (denial of service).

2. List out the problems of one time pad.

The one-time pad has serious drawbacks in practice because it requires:

- Truly random (as opposed to pseudorandom) one-time pad values, which is a non-trivial requirement.
- Secure generation and exchange of the one-time pad values, which must be at least as long as the message. (The security of the one-time pad is only as secure as the security of the one-time pad exchange).
- Careful treatment to make sure that it continues to remain secret, and is disposed of correctly preventing any reuse in whole or part—hence "one time".

3. What are the two basic functions used in encryption algorithms?

All the encryption algorithms are based on two general principles:

Substitution: In which each element in the plaintext(bit, letter, group of bits or letters) is mapped into another element.

Transposition: In which elements in the plaintext are rearranged.

The fundamental requirement is that no information be lost(that is ,that all operations are reversible). Most systems, referred to as product systems, involve multiple stages of substitutions and transpositions.

4. How many keys are required for two people to communicate via a cipher?

If both sender and receiver use the same key, the system is referred as symmetric, single-key, secret-key or conventional encryption. If both sender and receiver uses a different key, the system is referred as asymmetric, two-key or public key encryption.

5. What is a transposition cipher?

Transposition cipher, simple data encryption scheme in which plaintext characters are shifted in some regular pattern to form ciphertext.

In manual systems transpositions are generally carried out with the aid of an easily remembered mnemonic. For example, a popular schoolboy cipher is the “rail fence,” in which letters of the plaintext are written alternating between rows and the rows are then read sequentially to give the cipher. In a depth-two rail fence (two rows) the message WE ARE DISCOVERED SAVE YOURSELF would be written

```
W A E I C V R D A E O R E F
E R D S O E E S V Y U S L
```

or

```
W A E I C V R D A E O R E F E R D S O E E S V Y U S L .
```

If both sender and receiver use the same key, the system is referred as symmetric, single-key, secret-key or conventional encryption. If both sender and receiver uses a different key, the system is referred as asymmetric, two-key or public key encryption.

6. Define cryptography

Cryptography is a method of storing and transmitting data in a particular form so that only those for whom it is intended can read and process it. Cryptography is closely related to the disciplines of cryptology and cryptanalysis. Cryptography includes techniques such as microdots, merging words with

images, and other ways to hide information in storage or transit. However, in today's computer-centric world, cryptography is most often associated with scrambling plaintext (ordinary text, sometimes referred to as cleartext) into cipher text (a process called encryption), then back again (known as decryption).Modern cryptography concerns itself with the following four objectives:

1) **Confidentiality** (the information cannot be understood by anyone for whom it was unintended)

2) **Integrity** (the information cannot be altered in storage or transit between sender and intended receiver without the alteration being detected)

3) **Non-repudiation** (the creator/sender of the information cannot deny at a later stage his or her intentions in the creation or transmission of the information)

4) **Authentication** (the sender and receiver can confirm each other's identity and the origin/destination of the information)

7. What are the types of security attacks?

Passive. Network. Wiretapping. Port scan. Idle scan.

Active. Denial-of-service attack. Spoofing. Network. Man in the middle. ARP poisoning. Ping flood. Ping of death. Smurf attack. Host. Buffer overflow. Heap overflow. Stack overflow. Format string attack.

8. Define Fermat & Euler's theorem

In number theory, Fermat's Last Theorem (sometimes called Fermat's conjecture, especially in older texts) states that no three positive integers a,b, and c satisfy the equation $a^n + b^n = c^n$ for any integer value of n strictly greater than two. The cases $n = 1$ and $n = 2$ have been known to have infinitely many solutions since antiquity

In number theory, Euler's theorem (also known as the Fermat–Euler theorem or Euler's totient theorem) states that if n and a are coprime positive integers, then. where φ(n) is Euler's totient function. (The notation is explained in the article Modular arithmetic.)

9. Define the monoalphabetic cipher.

The monoalphabetic substitution takes a letter of an alphabet and substitutes it with another letter, this way a ciphertext is generated. The way of converting is fixed. A character of the plaintext will be replaced by the same ciphertext character, during the entire ciphertext. There is no additional key. The only way of security is to keep the substitution-table secret. A popular example for the monoalphabetic substitution is the caesar cipher.

Principle

The first step is to write down the plaintext alphabet. It includes all characters needed for the message. In this example only upper case letters will be used.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

The second step is to build an additional random alphabet.

QWERTZ*LKJHGFDSAÄÖÜMNBVCX

An **A** becomes **/**. **B** is replaced by **Q**. **C** replaced by **W**, etc. The word „EXAMPLE" would be encoded by: „RV/GSHR".

10. What is the difference between a block cipher and a stream cipher?

A **block cipher** is a deterministic and computable function of kk-bit keys and nn-bit (plaintext) blocks to nn-bit (ciphertext) blocks. (More generally, the blocks don't have to be bit-sized, nn-character-blocks would fit here, too). This means, when you encrypt the same plaintext block with the same key, you'll get the same result.

A **stream cipher** is a function which directly maps kk-bit keys and arbitrary length plaintexts to (same arbitrary length) ciphertext, in such a way that prefixes of the plaintext map to prefixes of the ciphertext, i.e. we can compute the starting part of the ciphertext before the trailing part of the plaintext is known. (Often the message sizes might be limited to multiples of some "block size", too, but usually with smaller blocks like whole bytes or such.)

11. What is integrity?

Data integrity refers to maintaining and assuring the accuracy and consistency of data over its entire life-cycle, and is a critical aspect to the design, implementation and usage of any system which stores, processes, or retrieves data.

12. What is non repudiation?

Nonrepudiation is the assurance that someone cannot deny something. Typically, nonrepudiation refers to the ability to ensure that a party to a contract or a communication cannot deny the authenticity of their signature on a document or the sending of a message that they originated.

13. Write about CIA triad.

Confidentiality and privacy are one and the same. This facet of the security triangle focuses on keeping sensitive data away from unauthorized parties. A commitment to confidentiality means organizations in

possession of vast amounts of information must adopt practices specifically built around safeguarding that information. It may very well call for a special training program that educate access-carrying employees across all departments on the best practices in password protection, social engineering, and other topics that are imperative to cyber security.

Integrity is all about ensuring the quality and consistency of data.Ensuring integrity can be as simple as creating policies that dictate which users have access to certain information and who has what level of file privileges.

Availability is the CIA facet that arguably demands the most from an organization. In a nutshell, it entails the ongoing process of doing whatever is necessary to keep all hardware and software components up and running.

14. **What are the classical encryption techniques?**

1. SUBSTITUTION TECHNIQUES
    - Caesar Cipher
    - Monoalphabetic Ciphers
    - Playfair Cipher
    - Hill Cipher
    - Polyalphabetic Ciphers
2. One-Time Pad
3. TRANSPOSITION TECHNIQUES
4. ROTOR MACHINES
5. STEGANOGRAPHY

15. What are the types of cryptographics attacks?

1. Fault analysis Attacks
2. Power Analysis Attacks
3. Timing Attacks
4. Side Channel Attack (SCA)
5. Man in Middle Attack (MIM)
6. Birthday Attack
7. Brute Force Attack (BFA)
8. Dictionary Attack
9. Chosen Plaintext Attack (CPA)
10. Known Plaintext Attack (KPA)
11. Ciphertext Only Attacks (COA)

16. Write about polynomial arithmetic.

Polynomial arithmetic is a branch of algebra dealing with some properties of polynomials which share strong analogies with properties of number theory relative to integers. It includes basic mathematical operations such as addition, subtraction, and multiplication, as well as more elaborate operations like Euclidean division, and properties related to roots of polynomials. The latter are essentially connected to the fact that the set $K[X]$ of univariate polynomials with coefficients in a field K is a commutative ring, such as the ring of integers $\mathbb{Z}$.

17. Write about the Chinese remainder problem.

The Chinese remainder theorem is a result about congruences in number theory and its generalizations in abstract algebra. It was first published some time between the 3rd and 5th centuries by the Chinese mathematician Sun Tzu. In its basic form, the Chinese remainder theorem will determine a number n that, when divided by some given divisors, leaves given remainders. In Sun Tzu's example (stated in modern terminology),[2] what is the smallest number n that when divided by 3 leaves a remainder of 2, when divided by 5 leaves a remainder of 3, and when divided by 7 leaves a remainder of 2

18. What are groups, Rings, and Fields?

A **group** is defined as: a set of elements, together with an operation performed on pairs of these elements such that:

1. The operation, when given two elements of the set as arguments, always returns an element of the set as its result. It is thus fully defined, and closed over the set.
2. One element of the set is an identity element. Thus, if we call our operation op, there is some element of the set e such that for any other element of the set x, e op x = x op e = x.
3. Every element of the set has an inverse element. If we take any element of the set p, there is another element q such that p op q = q op p = e.
4. The operation is associative. For any three elements of the set, (a op b) op c always equals a op (b op c).

A **ring** is a set of elements with two operations, one of which is like addition, the other of which is like multiplication, which we will call add and mul. It has the following properties:

1. The elements of the ring, together with the addition operation, form a group.
2. Addition is commutative. That is, for any two elements of the set p and q, p add q = q add p. (The word *Abelian* is also used for "commutative", in honor of the mathematician Niels Henrik Abel.)
3. The multiplication operation is associative.
4. Multiplication distributes over addition: that is, for any three elements of the group a, b, and c, a mul (b add c) = (a mul b) add (a mul c).
5. Addition and multiplication modulo 5 and modulo 6 both yield rings. Matrix multiplication also leads to rings as well.

A **field** is a ring in which the elements, other than the identity element for addition, and the multiplication operator, also form a group. There are only two kinds of finite fields. One kind is the field formed by addition and multiplication modulo a prime number. The other kind of finite field has a number of elements that is a power of a prime number. The addition operator consists of multiple independent additions modulo that prime. The elements of the field can be thought of as polynomials whose coefficients are numbers modulo that prime. In that case, multiplication is polynomial multiplication, where not only the coefficients modulo that prime, but the polynomials are are modulo a special kind of polynomial, known as a *primitive* polynomial. All finite fields, but particularly those of this second kind, are known as *Galois fields*.

19. Explain the substitution techniques.

A substitution cipher is a method of encoding by which units of plaintext are replaced with ciphertext, according to a fixed system; the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing the inverse substitution.

There are a number of different types of substitution cipher. If the cipher operates on single letters, it is termed a simple substitution cipher; a cipher that operates on larger groups of letters is termed polygraphic. A monoalphabetic cipher uses fixed substitution over the entire message, whereas a polyalphabetic cipher uses a number of substitutions at different positions in the message, where a unit from the plaintext is mapped to one of several possibilities in the ciphertext and vice versa.

20. Write about asymmetric encryption.

Asymmetric cryptography or public-key cryptography is cryptography in which a pair of keys is used to encrypt and decrypt a message so that it arrives securely. Initially, a network user receives a public and private key pair from a certificate authority.

21. Write about Symmetric Encryption.

Symmetric encryption is the oldest and best-known technique. A secret key, which can be a number, a word, or just a string of random letters, is applied to the text of a message to change the content in a particular way. This might be as simple as shifting each letter by a number of places in the alphabet.

22. Write about Steganography.

Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video. The word steganography combines the Greek words steganos meaning "covered, concealed, or protected", and graphein meaning "writing". The first recorded use of the term was in 1499 by Johannes Trithemius in his Steganographia, a treatise on cryptography and steganography, disguised as a book on magic. Generally, the hidden messages appear to be (or be part of) something else: images, articles, shopping lists, or some other cover text. For example, the hidden message may be in invisible ink between the visible lines of a private letter.

PART B

1. Using Playfair cipher algorithm encrypt the message using the key MONARCHY and explain.

PLAYFAIR CIPHER

The best known multiple letter encryption cipher is the playfair, which treats digrams in the plaintext as single units and translates these units into cipher text digrams.

The playfair algorithm is based on the use of **5x5 matrix of letters** constructed using a **keyword**.

Let the keyword be ‘monarchy’ The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetical order.The letter ‘I’ and ‘J’ count as one letter. Plaintext is encrypted two letters at a time according to the following rules:

- Repeating plaintext letters that would fall in the same pair are separated with a filler letter such as ‘x’.

- Plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row following the last. For example **ar** is encrypted as **RM**
- Plaintext letters that fall in the same column are replaced by the letter beneath, with the top element of the column following the last. For example **mu** is encrypted as **CM**
- Otherwise, each plaintext letter is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. For example **hs** is encrypted as **BP**

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

Example:

Plaintext = meet me at the school house

Splitting two letters as a unit => me et me at th es ch ox ol ho us ex

Corresponding cipher text => CL KL CL RS PD IL HY AV MP HF XL IU

me- 'm' and 'e' occurs in same column. Hence each letter is replaced by the letter beneath it-CL

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

et- '**e**' is replaced by '**K**'.**'t'** is replaced with '**L**'

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	**K**
L	P	Q	S	**T**
U	V	W	X	Z

ch -'c' is replaced with **'H'. 'h'** is replaced with **'Y'**

M	O	N	A	R
C	**H**	**Y**	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

Strength of playfair cipher

- Playfair cipher is a great advance over simple mono alphabetic ciphers.
- Since there are 26 letters, 26x26 = 676 diagrams are possible, so identification of individual digram is more difficult.

Frequency analysis is much more difficult .The relative frequencies of individual letters exhibit a much greater range than that of digrams making frequency analysis difficult. For this reason, the playfair cipher was for long time considered unbreakable

2. Explain Ceaser cipher and monoalphabetic cipher.

One of the simplest examples of a substitution cipher is the *Caesar cipher*, which is said to have been used by Julius Caesar to communicate with his army. Caesar is considered to be one of the first persons to have ever employed encryption for the sake of securing messages. Caesar decided that shifting each letter in the message would be his standard algorithm, and so he informed all of his generals of his decision, and was then able to send them secured messages. Using the Caesar Shift (3 to the right), the message,

"**RETURN TO ROME**"

would be encrypted as,

"**UHWXUA WR URPH**"

In this example, 'R' is shifted to 'U', 'E' is shifted to 'H', and so on. Now, even if the enemy did intercept the message, it would be useless, since only Caesar's generals could read it.

Thus, the Caesar cipher is a *shift cipher* since the ciphertext alphabet is derived from the plaintext alphabet by shifting each letter a certain number of spaces. For example, if we use a shift of 19, then we get the following pair of ciphertext and plaintext alphabets:

Plaintext: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Ciphertext: T U V W X Y Z A B C D E F G H I J K L M N O P Q R S

To encipher a message, we perform a simple substitution by looking up each of the message's letters in the top row and writing down the corresponding letter from the bottom row. For example, the message

THE FAULT, DEAR BRUTUS, LIES NOT IN OUR STARS BUT IN OURSELVES.

would be enciphered as

MAX YTNEM, WXTK UKNMNL, EBXL GHM BG HNK LMTKL UNM BG HNKLXEOXL.

Essentially, each letter of the alphabet has been shifted nineteen places ahead in the alphabet, wrapping around the end if necessary. Notice that punctuation and blanks are not enciphered but are copied over as themselves.

Mono alphabetic Cipher:

In cryptography, a **substitution cipher** is a method of encoding by which units of plaintext are replaced with ciphertext, according to a fixed system the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing the inverse substitution.

Substitution ciphers can be compared with transposition ciphers. In a transposition cipher, the units of the plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged. By contrast, in a substitution cipher, the units of the plaintext are retained in the same sequence in the ciphertext, but the units themselves are altered.

There are a number of different types of substitution cipher. If the cipher operates on single letters, it is termed a **simple substitution cipher**; a cipher that operates on larger groups of letters is termed **polygraphic cipher**. A **monoalphabetic cipher** uses fixed substitution over the entire message, whereas a **polyalphabetic cipher** uses a number of substitutions at different positions in the message, where a unit from the plaintext is mapped to one of several possibilities in the cipher text and vice versa.

Example : Start by creating a key that maps each letter of the alphabet to a (possibly the same) letter of the alphabet. A sample key might be:

Plaintext letter	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext letter	y	n	l	k	x	b	s	h	m	i	w	d	p	j	r	o	q	v	f	e	a	u	g	t	z	c

To encrypt the message "meet me at nine", start by taking the first letter of the message, 'm', and look up the corresponding ciphertext letter 'p'. Repeat by looking up the next plaintext letter 'e', and noting it becomes 'x'. Continue this process for the rest of the message. Typically spaces, numbers, and punctuation are left alone. In this case "meet me at nine." would become "pxxe px ye jmjx."

Decryption is similar. Start with the first ciphertext letter 'p', and look at the table to find the corresponding plaintext letter 'm'. Continue with the next letter 'x', and find it maps to 'e.

Monoalphabetic encryption is very easy to break, for two main reasons. First, commonly used letters like 'e' show up very quickly as the 'x' in the example. Second, words with repeated letters like "meet" in the example show that repetition in the ciphertext. And indeed this is so weak that the daily cryptogram run by some newspapers is typically an monoalphabetic substitution.

3. Explain Key generation, Encryption and Decryption in detail.

Key Generation:

Modern cryptographic systems include **symmetric-key algorithms** (such as DES and AES) and **public-key algorithms** (such as RSA). Symmetric-key algorithms use a single shared key, keeping data secret requires keeping this key secret. Public-key algorithms use a public key and a private key. The public key is made available to anyone (often by means of a digital certificate). A sender encrypts data with the public key, only the holder of the private key can decrypt this data.

Since public-key algorithms tend to be much slower than symmetric-key algorithms, modern systems such as TLS and SSH use a combination of the two: one party receives the other's public key, and encrypts a small piece of data (either a symmetric key or some data used to generate it). The remainder of the conversation uses a (typically faster) symmetric-key algorithm for encryption.

The simplest method to read encrypted data without actually decrypting it is a brute force attack- simply attempting every number, up to the maximum length of the key. Therefore, it is important to use a sufficiently long key length. Longer keys take exponentially longer to attack, rendering a brute force attack impractical. Currently, key lengths of 128 bits (for symmetric key algorithms) and 1024 bits (for public-key algorithms) are common.

Encryption and Decryption:

Encryption is the process of transforming information so it is unintelligible to anyone but the intended recipient. Decryption is the process of transforming encrypted information so that it is intelligible again. A cryptographic algorithm, also called a cipher, is a mathematical function used for encryption or decryption. In most cases, two related functions are employed, one for encryption and the other for decryption.

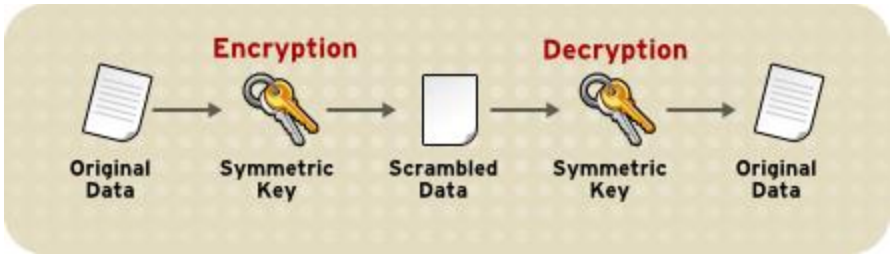
With most modern cryptography, the ability to keep encrypted information secret is based not on the cryptographic algorithm, which is widely known, but on a number called a key that must be used with the algorithm to produce an encrypted result or to decrypt previously encrypted information. Decryption with the correct key is simple. Decryption without the correct key is very difficult, and in some cases impossible for all practical purposes.

The sections that follow introduce the use of keys for encryption and decryption.

- Symmetric-Key Encryption
- Public-Key Encryption
- Key Length and Encryption Strength

Symmetric-Key Encryption

With symmetric-key encryption, the encryption key can be calculated from the decryption key and vice versa. With most symmetric algorithms, the same key is used for both encryption and decryption, as shown in Figure 1.



Implementations of symmetric-key encryption can be highly efficient, so that users do not experience any significant time delay as a result of the encryption and decryption. Symmetric-key encryption also provides a degree of authentication, since information encrypted with one symmetric key cannot be decrypted with any other symmetric key. Thus, as long as the symmetric key is kept secret by the two parties using it to encrypt communications, each party can be sure that it is communicating with the other as long as the decrypted messages continue to make sense.

Symmetric-key encryption is effective only if the symmetric key is kept secret by the two parties involved. If anyone else discovers the key, it affects both confidentiality and authentication. A person with an unauthorized symmetric key not only can decrypt messages sent with that key, but can encrypt new messages and send them as if they came from one of the two parties who were originally using the key.

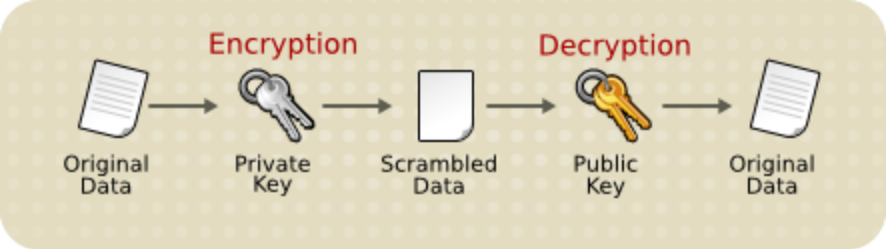
Symmetric-key encryption plays an important role in the SSL protocol, which is widely used for authentication, tamper detection, and encryption over TCP/IP networks. SSL also uses techniques of public-key encryption, which is described in the next section.

Public-Key Encryption

The most commonly used implementations of public-key encryption are based on algorithms patented by RSA Data Security. Therefore, this section describes the RSA approach to public-key encryption.

Public-key encryption (also called asymmetric encryption) involves a pair of keys-a public key and a private key-associated with an entity that needs to authenticate its identity electronically or to sign or encrypt data. Each public key is published, and the corresponding private key is kept secret. Data encrypted

with your public key can be decrypted only with your private key. Figure 2 shows a simplified view of the way public-key encryption works.



The scheme shown in Figure 2 lets you freely distribute a public key, and only you will be able to read data encrypted using this key. In general, to send encrypted data to someone, you encrypt the data with that person's public key, and the person receiving the encrypted data decrypts it with the corresponding private key.

Compared with symmetric-key encryption, public-key encryption requires more computation and is therefore not always appropriate for large amounts of data. However, it's possible to use public-key encryption to send a symmetric key, which can then be used to encrypt additional data. This is the approach used by the SSL protocol.

As it happens, the reverse of the scheme shown in Figure 2 also works: data encrypted with your private key can be decrypted only with your public key. This would not be a desirable way to encrypt sensitive data, however, because it means that anyone with your public key, which is by definition published, could decrypt the data. Nevertheless, private-key encryption is useful, because it means you can use your private key to sign data with your digital signature-an important requirement for electronic commerce and other commercial applications of cryptography. Client software such as Firefox can then use your public key to confirm that the message was signed with your private key and that it hasn't been tampered with since being signed. "Digital Signatures" describes how this confirmation process works.

Key Length and Encryption Strength

Breaking an encryption algorithm is basically finding the key to the access the encrypted data in plain text. For symmetric algorithms, breaking the algorithm usually means trying to determine the key used to encrypt the text. For a public key algorithm, breaking the algorithm usually means acquiring the shared secret information between two recipients.

One method of breaking a symmetric algorithm is to simply try every key within the full algorithm until the right key is found. For public key algorithms, since half of the key pair is publicly known, the other half (private key) can be derived using published, though complex, mathematical calculations. Manually finding the key to break an algorithm is called a brute force attack.

Breaking an algorithm introduces the risk of intercepting, or even impersonating and fraudulently verifying, private information.

The key strength of an algorithm is determined by finding the fastest method to break the algorithm and comparing it to a brute force attack.

For symmetric keys, encryption strength is often described in terms of the size or length of the keys used to perform the encryption: in general, longer keys provide stronger encryption. Key length is measured in bits. For example, 128-bit keys for use with the RC4 symmetric-key cipher supported by SSL provide significantly better cryptographic protection than 40-bit keys for use with the same cipher. Roughly speaking, 128-bit RC4 encryption is 3 x 10^{26} times stronger than 40-bit RC4 encryption. (For more information about RC4 and other ciphers used with SSL, see "Introduction to SSL.") An encryption key is considered full strength if the best known attack to break the key is no faster than a brute force attempt to test every key possibility.

Different ciphers may require different key lengths to achieve the same level of encryption strength. The RSA cipher used for public-key encryption, for example, can use only a subset of all possible values for a key of a given length, due to the nature of the mathematical problem on which it is based. Other ciphers, such as those used for symmetric key encryption, can use all possible values for a key of a given length, rather than a subset of those values. Because it is relatively trivial to break an RSA key, an RSA public-key encryption cipher must have a very long key, at least 1024 bits, to be considered cryptographically strong. On the other hand, symmetric-key ciphers can achieve approximately the same level of strength with an 80-bit key for most algorithms.

4. Explain about the different encryption mechanisms in detail with examples.

In cryptography, **encryption** is the process of encoding messages or information in such a way that only authorized parties can read it. Encryption does not of itself prevent interception, but denies the message content to the interceptor. In an encryption scheme, the intended communication information or message, referred to as plaintext, is encrypted using an encryption algorithm, generating ciphertext that can only be read if decrypted. For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm. It is in principle possible to decrypt the message without possessing the key, but, for a well-designed encryption scheme, large computational resources and skill are required. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients, but not to unauthorized interceptors.

The purpose of encryption is to ensure that only somebody who is authorized to access data (e.g. a text message or a file), will be able to read it, using the decryption key. Somebody who is not authorized can be excluded, because he or she does not have the required key, without which it is impossible to read the encrypted information.

Types of Encryption:

Symmetric key encryption

In symmetric-key schemes, the encryption and decryption keys are the same. Communicating parties must have the same key before they can achieve secure communication.

Public key encryption

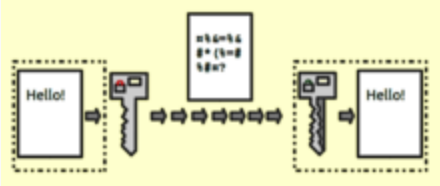


Illustration of how encryption is used within servers Public key encryption.

Public key encryption

In public-key encryption schemes, the encryption key is published for anyone to use and encrypt messages. However, only the receiving party has access to the decryption key that enables messages to be read. Public-key encryption was first described in a secret document in 1973, before then, all encryption schemes were symmetric-key (also called private-key).

5. Explain classical encryption techniques

- cryptography is the study of **secret** (crypto-) **writing** (-graphy)
- concerned with developing algorithms which may be used to:
  - conceal the context of some message from all except the sender and recipient (privacy or secrecy), and/or
  - verify the correctness of a message to the recipient (**authentication**)

cryptography

the art or science encompassing the principles and methods of transforming an intelligible message into one that is unintelligible, and then retransforming that message back to its original form

plaintext

the original intelligible message

ciphertext

the transformed message

cipher

an algorithm for transforming an intelligible message into one that is unintelligible by transposition and/or substitution methods

key

some critical information used by the cipher, known only to the sender & receiver

encipher (encode)

the process of converting plaintext to ciphertext using a cipher and a key

decipher (decode)

the process of converting ciphertext back into plaintext using a cipher and a key

cryptanalysis

the study of principles and methods of transforming an unintelligible message back into an intelligible message *without* knowledge of the key. Also called **codebreaking**

cryptology

both cryptography and cryptanalysis

code

an algorithm for transforming an intelligible message into an unintelligible one using a code-book

Concepts

Encryption C = E_(K)(P)

Decryption P = E_(K)^(-1)(C)

E_(K) is chosen from a family of **transformations** known as a **cryptographic system.**

The parameter that selects the individual transformation is called the **key** K, selected from a **keyspace *K***

More formally a **cryptographic system** is a single parameter family of invertible transformations

E_(K) ; K in ***K*** : P -> C

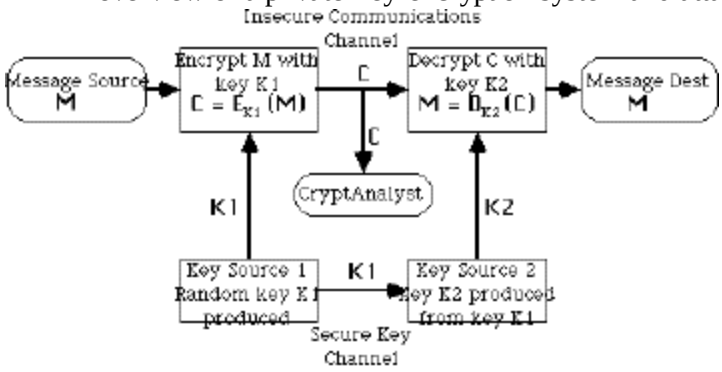
with the inverse algorithm E_(K)^(-1) ; K in ***K*** : C -> P

such that the inverse is unique

Usually assume the cryptographic system is public, and only the key is secret information

Private-Key Encryption Algorithms

- a private-key (or secret-key, or single-key) encryption algorithm is one where the sender and the recipient share a common, or closely related, key
- all traditional encryption algorithms are private-key
- overview of a private-key encryption system and attacker



Symmetric (Private-Key) Encryption System

Cryptanalytic Attacks

- have several different types of attacks

ciphertext only

- only have access to some enciphered messages
- use statistical attacks only

known plaintext

- know (or strongly suspect) some plaintext-ciphertext pairs
- use this knowledge in attacking cipher

chosen plaintext

- can select plaintext and obtain corresponding ciphertext
- use knowledge of algorithm structure in attack

chosen plaintext-ciphertext

- can select plaintext and obtain corresponding ciphertext, or select ciphertext and obtain plaintext
- allows further knowledge of algorithm structure to be used

Unconditional and Computational Security

- two fundamentally different ways ciphers may be secure

  unconditional security

  - no matter how much computer power is available, the cipher cannot be broken

  computational security

  - given limited computing resources (eg time needed for calculations is greater than age of universe), the cipher cannot be broken

Classical Cryptographic Techniques

- have two basic components of classical ciphers: **substitution** and **transposition**
- in substitution ciphers letters are replaced by other letters
- in transposition ciphers the letters are arranged in a different order
- these ciphers may be:
- **monoalphabetic** - only one substitution/ transposition is used, or
- **polyalphabetic** - where several substitutions/ transpositions are used
- several such ciphers may be concatentated together to form a **product cipher**

Caesar Cipher - a monoalphabetic cipher

- replace each letter of message by a letter a fixed distance away eg use the 3rd letter on
- reputedly used by Julius Caesar

eg.

L FDPH L VDZ L FRQTXHUHG
I CAME I SAW I CONQUERED

General Monoalphabetic

- special form of mixed alphabet
- use key as follows:
  - write key (with repeated letters deleted)
  - then write all remaining letters in columns underneath
  - then read off by columns to get ciphertext equivalents

Example Seberry p66

STARW
BCDEF
GHIJK
LMNOP
QUVXY
Z

Plain: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher: SBGLQZTCHMUADINVREJOXWFKPY

Plaintext: I KNOW ONLY THAT I KNOW NOTHING
Ciphertext: H UINF NIAP OCSO H UINF INOCHIT

Polyalphabetic Substitution

- in general use more than one substitution alphabet
- makes cryptanalysis harder since have more alphabets to guess
- and because flattens frequency distribution
- (since same plaintext letter gets replaced by several ciphertext letter, depending on which alphabet is used)

Vigenère Cipher

- basically multiple caesar ciphers
- key is multiple letters long K = k_(1) k_(2) ... k_(d)
- ith letter specifies ith alphabet to use
- use each alphabet in turn, repeating from start after d letters in message

Plaintext THISPROCESSCANALSOBEEXPRESSED
Keyword CIPHERCIPHERCIPHERCIPHERCIPHE
Plaintext VPXZTIQKTZWTCVPSWFDMTETIGAHLH

- can use a **Saint-Cyr Slide** for easier encryption

6. Explain security attacks and services

Security Attacks

- Attacks against the encryption scheme (cryptoanalytic attacks).
- Chosen-plaintext - Same as Known-plaintext attack, but tha analyst gets to choose the known plaintext.

- Cyphertext-only - Attempt to recover plaintext from encrypted text sent in the message.
- Known-plaintext - Attempt to discover the key used when the analyst has access to the plaintext of the encrypted message.
- Brute force
- differential cryptoanalysis
- linear cryptoanalysis
- Passive - An attack such as listening to communications then attacking the encryption scheme off line may be done.
- Active - A common attack of this type is the man in the middle attack. During this attack the attacker may try to convince the victim that they are communicating with another party when they are really communicating with the attacker. The attacker may use the attack to gain passwords or other vital information.
- Dictionary attack - A means attacking a system to determine passwords from hashed or encrypted passwords.

Security Attacks

DoS- Denial of Service

Trojan Horse - Comes with other software.

Virus - Reproduces itself by attaching to other executable files.

Worm - Self-reproducing program. Creates copies of itself. Worms that spread using e-mail address books are often called viruses.

Logic Bomb - Dormant until an event triggers it (Date, user action, random trigger, etc.).

Hacker Attacks

I use the term "hacker attacks" to indicate hacker attacks that are not automated by programs such as viruses, worms, or trojan horse programs. There are various forms that exploit weakneses in security. Many of these may cause loss of service or system crashes.

IP spoofing - An attacker may fake their IP address so the receiver thinks it is sent from a location that it is not actually from. There are various forms and results to this attack.

The attack may be directed to a specific computer addressed as though it is from that same computer. This may make the computer think that it is talking to itself. This may cause some operating systems such as Windows to crash or lock up.

Gaining access through source routing. Hackers may be able to break through other friendly but less secure networks and get access to your network using this method.

Man in the middle attack -

Session hijacking - An attacker may watch a session open on a network. Once authentication is complete, they may attack the client computer to disable it, and use IP spoofing to claim to be the client who was just authenticated and steal the session. This attack can be prevented if the two legitimate systems share a secret which is checked periodically during the session.

Server spoofing - A C2MYAZZ utility can be run on Windows 95 stations to request LANMAN (in the clear) authentication from the client. The attacker will run this utility while acting like the server while the user attempts to login. If the client is tricked into sending LANMAN authentication, the attacker can read their username and password from the network packets sent.

DNS poisoning - This is an attack where DNS information is falsified. This attack can succeed under the right conditions, but may not be real practical as an attack form. The attacker will send incorrect DNS information which can cause traffic to be diverted. The DNS information can be falsified since name servers do not verify the source of a DNS reply. When a DNS request is sent, an attacker can send a false DNS reply with additional bogus information which the requesting DNS server may cache. This attack can be used to divert users from a correct webserver such as a bank and capture information from customers when they attempt to logon.

Password cracking - Used to get the password of a user or administrator on a network and gain unauthorized access.

Some DoS Attacks

Ping broadcast - A ping request packet is sent to a broadcast network address where there are many hosts. The source address is shown in the packet to be the IP address of the computer to be attacked. If the router to the network passes the ping broadcast, all computers on the network will respond with a ping reply to the sttacked system. The attacked system will be flooded with ping responses which will

cause it to be unable to operate on the network for some time, and may even cause it to lock up. The attacked computer may be on someone else's network. One countermeasure to this attack is to block incoming traffic that is sent to a broadcast address.

Ping of death - An oversized ICMP datagram can crash IP devices that were made before 1996.

Smurf - An attack where a ping request is sent to a broadcast network address with the sending address spoofed so many ping replies will come back to the victim and overload the ability of the victim to process the replies.

Teardrop - a normal packet is sent. A second packet is sent which has a fragmentation offset claiming to be inside the first fragment. This second fragment is too small to even extend outside the first fragment. This may cause an unexpected error condition to occur on the victim host which can cause a buffer overflow and possible system crash on many operating systems.

Services and Mechanisms

Having identified the relevant security threats to a system, the system operator can apply various security services and mechanisms to confront these threats and implement a desired security policy. In this section we provide a general description of such services and techniques. The science behind these methods is researched and developed as part of the broad discipline of Cryptography. Cryptography embodies the mathematical principles, means, and methods for the transformation of data in order to hide its information content, prevent its undetected modification, and/or prevent its unauthorized use. Cryptographic functions may be used as part of encipherment, decipherment, data integrity, authentication exchanges, password storage and checking, etc. to help achieve confidentiality, integrity, and/or authentication.

The following subsections summarize some key security services and mechanisms.

Encipherment and Data Confidentiality

Encipherment is a security mechanism that involves the transformation of data into some unreadable form. Its purpose is to ensure privacy by keeping the information hidden from anyone for whom it is not intended, even those who can see enciphered data. Decipherment is the reverse of encipherment. That is, it is the transformation of encrypted data back into some intelligible form. Encipherment which is performed on cleartext (intelligible data) to produce ciphertext (encrypted data whose semantic content is not available). The result of decipherment is either cleartext, or ciphertext under some cover.

Encipherment can provide confidentiality of either data or traffic flow information and can play a part in, or complement other security mechanisms.

Encipherment and Decipherment require the use of some secret information, usually referred to as a key, which directs specific transformations. This is one of two cryptovariables used: The other is the initialization variable, which is sometimes required to preserve the apparent randomness of ciphertext.

Encipherment techniques can be symmetric or secret key, where knowledge of the encipherment key implies knowledge of the private decipherment key and vice versa, or asymmetric. In asymmetric algorithms, generally one key is called public (because it is publicly available), while the other is called private (because it is kept secret). Once a private key has been compromised, the system (or at least the use of that private key) is no longer secure. Both encipherment techniques are used to provide the data confidentiality service.

Modern cryptographic systems also provides mechanisms for authentication, for instance through digital signatures that bind a document to the possessor of a specific key, or digital timestamps which bind a document to its creation at a given time. In general the existence of an encipherment mechanism implies the use of a key management mechanism.

Public Key Cryptography

When Alice wishes to send a secret message to Bob, she looks up Bob's public key in a directory, uses it to encrypt the message, and sends it off. Bob then uses his private key to decrypt the message and read it. No one listening in can decrypt the message. Anyone can send Bob an encrypted message but only Bob can read it. Clearly one requirement is that no one can figure out the private key from the corresponding public key.

7. Describe Chinese Reminder Theorem & Modular Exponentiation with an example

The **Chinese remainder theorem** is a result about congruences in number theory and its generalizations in abstract algebra. It was first published some time between the 3rd and 5th centuries by the Chinese mathematician Sun Tzu.

In its basic form, the Chinese remainder theorem will determine a number *n* that, when divided by some given divisors, leaves given remainders. In Sun Tzu's example (stated in modern terminology), what is the smallest number *n* that when divided by 3 leaves a remainder of 2, when divided by 5 leaves a remainder of 3, and when divided by 7 leaves a remainder of 2

The Chinese Remainder Theorem

Suppose we wish to solve

$$x=2(\mathrm{mod}5)x=2(\mathrm{mod}5)$$
$$x=3(\mathrm{mod}7)x=3(\mathrm{mod}7)$$

for xx. If we have a solution yy, then y+35y+35 is also a solution, and more generally yy plus any mulitple of 3535. So we only need to look for solutions modulo 3535. By brute force, we find the only solution is $x=17(\mathrm{mod}35)x=17(\mathrm{mod}35)$.

For any system of equations like this, the Chinese Remainder Theorem tells us there is always a unique solution up to a certain modulus, and describes how to find the solution efficiently.

Theorem: Let p,qp,q be coprime. Then the system of equations

$$x=a(\mathrm{mod}p)x=a(\mathrm{mod}p)$$
$$x=b(\mathrm{mod}q)x=b(\mathrm{mod}q)$$

has a unique solution for xx modulo pqpq.

The reverse direction is trivial: given $x\in Zpqx\in Zpq$, we can reduce xx modulo pp and xx modulo qq to obtain two equations of the above form.

Proof: Let $p1=p-1(\mathrm{mod}q)p1=p-1(\mathrm{mod}q)$ and $q1=q-1(\mathrm{mod}p)q1=q-1(\mathrm{mod}p)$. These must exist since p,qp,q are coprime. Then we claim that if yy is an integer such that

$$y=aqq1+bpp1(\mathrm{mod}pq)y=aqq1+bpp1(\mathrm{mod}pq)$$

then yy satisfies both equations:

Modulo pp, we have $y=aqq1=a(\mathrm{mod}p)y=aqq1=a(\mathrm{mod}p)$ since $qq1=1(\mathrm{mod}p)qq1=1(\mathrm{mod}p)$.

Similarly $y=b(\mathrm{mod}q)y=b(\mathrm{mod}q)$. Thus yy is a solution for xx.

It remains to show no other solutions exist modulo pqpq. If $z=a(\mathrm{mod}p)z=a(\mathrm{mod}p)$ then $z-yz-y$ is a multiple of pp. If $z=b(\mathrm{mod}q)z=b(\mathrm{mod}q)$ as well, then $z-yz-y$is also a multiple of qq. Since pp and qq are coprime, this implies $z-yz-y$ is a multiple of pqpq, hence $z=y(\mathrm{mod}pq)z=y(\mathrm{mod}pq)$. ■

This theorem implies we can represent an element of ZpqZpq by one element of ZpZp and one element of ZqZq, and vice versa. In other words, we have a bijection between ZpqZpq and $Zp\times ZqZp\times Zq$.

Examples: We can write $17\in Z3517\in Z35$ as $(2,3)\in Z5\times Z7(2,3)\in Z5\times Z7$. We can write $1\in Zpq1\in Zpq$ as $(1,1)\in Zp\times Zq(1,1)\in Zp\times Zq$.

In fact, this correspondence goes further than a simple relabelling. Suppose $x,y\in Zpqx,y\in Zpq$ correspond to $(a,b),(c,d)\in Zp\times Zq(a,b),(c,d)\in Zp\times Zq$ respectively. Then a little thought shows $x+yx+y$ corresponds to $(a+c,b+d)(a+c,b+d)$, and similarly xyxy corresponds to $(ac,bd)(ac,bd)$.

A practical application: if we have many computations to perform on $x\in Zpqx\in Zpq$ (e.g. RSA signing and decryption), we can convert xx to $(a,b)\in Zp\times Zq(a,b)\in Zp\times Zq$ and do all the computations on aa and bb instead before converting back. This is often cheaper because for many algorithms, doubling the size of the input more than doubles the running time.

Example: To compute $17\times17(\mathrm{mod}35)17\times17(\mathrm{mod}35)$, we can compute $(2\times2,3\times3)=(4,2)(2\times2,3\times3)=(4,2)$ in $Z5\times Z7Z5\times Z7$, and then apply the Chinese Remainder Theorem to find that $(4,2)(4,2)$ is $9(\mathrm{mod}35)9(\mathrm{mod}35)$.

Let us restate the Chinese Remainder Theorem in the form it is usually presented.

8. Write short notes on LFSR sequences & Finite fields

A linear feedback shift register (LFSR) is a shift register whose input bit is a linear function of its previous state.

The only linear function of single bits is xor, thus it is a shift register whose input bit is driven by the exclusive-or (xor) of some bits of the overall shift register value.

The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well-chosen feedback function can produce a sequence of bits which appears random and which has a very long cycle.

Applications of LFSRs include generating pseudo-random numbers, pseudo-noise sequences, fast digital counters, and whitening sequences. Both hardware and software implementations of LFSRs are common.
Fibonacci LFSRs
The bit positions that affect the next state are called the taps. In the diagram the taps are [16,14,13,11]. The rightmost bit of the LFSR is called the output bit. The taps are XOR'd sequentially with the output bit and then fed back into the leftmost bit. The sequence of bits in the rightmost position is called the output stream.
The bits in the LFSR state which influence the input are called taps (white in the diagram).
A maximum-length LFSR produces an m-sequence (i.e. it cycles through all possible $2n − 1$ states within the shift register except the state where all bits are zero), unless it contains all zeros, in which case it will never change.
As an alternative to the XOR based feedback in an LFSR, one can also use XNOR.[1] This function is not linear, but it results in an equivalent polynomial counter whose state of this counter is the complement of the state of an LFSR. A state with all ones is illegal when using an XNOR feedback, in the same way as a state with all zeroes is illegal when using XOR. This state is considered illegal because the counter would remain "locked-up" in this state.
The sequence of numbers generated by an LFSR or its XNOR counterpart can be considered a binary numeral system just as valid as Gray code or the natural binary code.
The arrangement of taps for feedback in an LFSR can be expressed in finite field arithmetic as a polynomial mod 2. This means that the coefficients of the polynomial must be 1's or 0's. This is called the feedback polynomial or characteristic polynomial. For example, if the taps are at the 16th, 14th, 13th and 11th bits (as shown), the feedback polynomial is
x^{16} + x^{14} + x^{13} + x^{11} + 1.\,
The 'one' in the polynomial does not correspond to a tap — it corresponds to the input to the first bit (i.e. x0, which is equivalent to 1). The powers of the terms represent the tapped bits, counting from the left. The first and last bits are always connected as an input and tap respectively.
Tables of primitive polynomials from which maximum-length LFSRs can be constructed are given below and in the references.

- The LFSR will only be maximum-length if the number of taps is even; just 2 or 4 taps can suffice even for extremely long sequences.
- The set of taps must be relatively prime, and share no common divisor to all taps.
- There can be more than one maximum-length tap sequence for a given LFSR length
- Once one maximum-length tap sequence has been found, another automatically follows. If the tap sequence, in an n-bit LFSR, is [n, A, B, C, 0], where the 0 corresponds to the x0 = 1 term, then the corresponding 'mirror' sequence is [n, n − C, n − B, n − A, 0]. So the tap sequence [32, 7, 3, 2, 0] has as its counterpart [32, 30, 29, 25, 0]. Both give a maximum-length sequence.

Some example C/C++ code is below (assuming 16-bit shorts):

```
unsigned short lfsr = 0xACE1u;
unsigned bit;
unsigned period = 0;
do
{
  /* taps: 16 14 13 11; characteristic polynomial:  x^16 + x^14 + x^13 + x^11 + 1 */
  bit  = ((lfsr >> 0) ^ (lfsr >> 2) ^ (lfsr >> 3) ^ (lfsr >> 5) ) & 1;
  lfsr =  (lfsr >> 1) | (bit << 15);
  ++period;
} while(lfsr != 0xACE1u);
```

The above code assumes the most significant bit of lfsr is bit 1, and the least significant bit is bit 16.

As well as Fibonacci, this LFSR configuration is also known as standard, many-to-one or external XOR gates. LFSR has an alternative configuration.

9. Explain Legendre & Jacobi symbols

The **Jacobi symbol** is a generalization of the Legendre symbol. Introduced by Jacobi in 1837, it is of theoretical interest in modular arithmetic and other branches of number theory, but its main use is in computational number theory, especially primality testing and integer factorization; these in turn are important in cryptography.

Let $a \geq 0, n \in \mathbf{Z}^+$. Let $\mathrm{QR}(a,n)$ hold if $(a,n) = 1$ and a is a quadratic residue modulo n. Let $\mathrm{QNR}(a,n)$ hold if $(a,n) = 1$ and a is a quadratic non-residue modulo n (i.e., there is no $y \in \mathbf{Z}_n^*$ such that $a \equiv y^2 \pmod{n}$).

For a prime p, the structure of quadratic residues can be fairly easily explained. Let g be a primitive root of $\mathbf{Z}_p^*$. Then every element of $\mathbf{Z}_p^*$ is uniquely expressible as g^k for some $k \in \{0, \ldots, p-2\}$.

Theorem 1 *Let p be a prime, g a primitive root of p, $a \equiv g^k \pmod{p}$. Then a is a quadratic residue iff k is even.*

Proof: If k is even, then $g^{k/2}$ is easily seen to be a square root of a modulo p.

Conversely, suppose $a \equiv y^2 \pmod{p}$. Write $y \equiv g^\ell \pmod{p}$. Then $g^k \equiv g^{2\ell} \pmod{p}$. Multiplying both sides by g^{-k}, we have $1 \equiv g^0 \equiv g^{2\ell-k} \pmod{p}$. But then $\phi(p) \mid 2\ell - k$. Since $2 \mid \phi(p) = p - 1$, it follows that $2 \mid k$, as desired. ∎

The following theorem, due to Euler, is now easily proved:

Theorem 2 (Euler) *Let p be an odd prime, and let $a \geq 0$, $(a,p) = 1$. Then*

$$a^{(p-1)/2} \equiv \begin{cases} 1 \pmod{p} & \text{if } \mathrm{QR}(a,p) \text{ holds;} \\ -1 \pmod{p} & \text{if } \mathrm{QNR}(a,p) \text{ holds.} \end{cases}$$

Proof: Write $a \equiv g^k \pmod{p}$.

If $\mathrm{QR}(a,p)$ holds, then a is a quadratic residue modulo p, so k is even by Theorem 1. Write $k = 2r$ for some r. Then $a^{(p-1)/2} \equiv g^{2r(p-1)/2} \equiv (g^r)^{p-1} \equiv 1 \pmod{p}$ by Fermat's theorem.

If $\mathrm{QNR}(a,p)$ holds, then a is a quadratic non-residue modulo p, so k is odd by Theorem 1. Write $k = 2r + 1$ for some r. Then $a^{(p-1)/2} \equiv g^{(2r+1)(p-1)/2} \equiv g^{r(p-1)} g^{(p-1)/2} \equiv g^{(p-1)/2} \pmod{p}$. Let $b = g^{(p-1)/2}$. Clearly $b^2 \equiv 1 \pmod{p}$, so $b \equiv \pm 1 \pmod{p}$.[1] Since g is a primitive root modulo p and $(p-1)/2 < p - 1$, $b = g^{(p-1)/2} \not\equiv 1 \pmod{p}$. Hence, $b \equiv -1 \pmod{p}$.

∎

Definition: The Legendre symbol is a function of two integers a and p, written $\left(\frac{a}{p}\right)$. It is defined for $a \geq 0$ and p an odd prime as follows:

$$\left(\frac{a}{p}\right) = \begin{cases} 1 \text{ if } \mathrm{QR}(a,p) \text{ holds;} \\ -1 \text{ if } \mathrm{QNR}(a,p) \text{ holds;} \\ 0 \text{ if } (a,p) \neq 1. \end{cases}$$

A multiplicative property of the Legendre symbols follows immediately from Theorem 1.

Observation 3 *Let $a, b \geq 0$, p an odd prime. Then*

$$\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right) \cdot \left(\frac{b}{p}\right).$$

[1]This follows from the fact that $p|(b^2 - 1) = (b-1)(b+1)$, so either $p|(b-1)$, in which case $b \equiv 1 \pmod{p}$, or $p|(b+1)$, in which case $b \equiv -1 \pmod{p}$.

As an easy corollary of Theorem 2, we have:

Corollary 4 *Let $a \geq 0$ and let p be an odd prime. Then*

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p}.$$

The Jacobi symbol extends the domain of the Legendre symbol.

Definition: The Jacobi symbol is a function of two integers a and n, written $\left(\frac{a}{n}\right)$, that is defined for all $a \geq 0$ and all odd positive integers n. Let $\prod_{i=1}^{k} p_i^{e_i}$ be the prime factorization of n. Then

$$\left(\frac{a}{n}\right) = \prod_{i=1}^{k} \left(\frac{a}{p_i}\right)^{e_i}.$$

Here $\left(\frac{a}{p_i}\right)$ denotes the previously-defined Legendre symbol. (Note that by this definition, $\left(\frac{0}{1}\right) = 1$, and $\left(\frac{0}{n}\right) = 0$ for odd $n \geq 3$.)

Here $\left(\frac{a}{p_i}\right)$ denotes the previously-defined Legendre symbol. (Note that by this definition, $\left(\frac{0}{1}\right) = 1$, and $\left(\frac{0}{n}\right) = 0$ for odd $n \geq 3$.)

We have seen that if $(a, p) = 1$ and p is prime, then the Legendre symbol $\left(\frac{a}{p}\right) = 1$ iff a is a quadratic residue modulo p. It is *not* true for the Jacobi symbol that $\left(\frac{a}{n}\right) \equiv 1 \pmod{n}$ implies that a is a quadratic residue modulo n. For example, $\left(\frac{8}{15}\right) = 1$, but 8 is not a quadratic residue modulo 15. However, the converse does hold:

Observation 5 *If $\left(\frac{a}{n}\right) \not\equiv 1 \pmod{n}$, then a is not a quadratic residue modulo n.*

The usefulness of the Jacobi symbol $\left(\frac{a}{n}\right)$ stems from its ability to be computed efficiently, even without knowning the factorization of either a or n. The algorithm is based on the following theorem, which is stated without proof.

Theorem 6 *Let n be an odd positive integer, $a, b \geq 0$. Then the following identities hold:*

1. $\left(\frac{0}{n}\right) = \begin{cases} 1 & \text{if } n = 1; \\ 0 & \text{if } n > 1 \end{cases}$
2. $\left(\frac{2}{n}\right) = \begin{cases} 1 & \text{if } n \equiv \pm 1 \pmod{8}; \\ -1 & \text{if } n \equiv \pm 3 \pmod{8} \end{cases}$
3. $\left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$ *if* $a \equiv b \pmod{n}$.
4. $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right) \cdot \left(\frac{b}{n}\right)$
5. *(Quadratic reciprocity). If a is odd, then*
$\left(\frac{a}{n}\right) = \begin{cases} -\left(\frac{n}{a}\right) & \text{if } a \equiv n \equiv 3 \pmod{4}; \\ \left(\frac{n}{a}\right) & \text{otherwise.} \end{cases}$

Theorem 6 leads directly to a recursive algorithm for computing $\left(\frac{a}{n}\right)$:

10. Write in detail about the symmetric encryption mechanisms.

An encryption system in which the sender and receiver of a message share a single, common key that is used to encrypt and decrypt the message. Contrast this with public-key cryptology, which utilizes two keys - a public key to encrypt messages and a private key to decrypt them.

Symmetric-key systems are simpler and faster, but their main drawback is that the two parties must somehow exchange the key in a secure way. Public-key encryption avoids this problem because the public key can be distributed in a non-secure way, and the private key is never transmitted.

Symmetric-key cryptography is sometimes called *secret-key cryptography*. The most popular symmetric-key system is the *Data Encryption Standard (DES)*.

1. Triple DES

Triple DES was designed to replace the original Data Encryption Standard (DES) algorithm, which hackers eventually learned to defeat with relative ease. At one time, Triple DES was the recommended standard and the most widely used symmetric algorithm in the industry.

Triple DES uses three individual keys with 56 bits each. The total key length adds up to 168 bits, but experts would argue that 112-bits in key strength is more like it.

Despite slowly being phased out, Triple DES still manages to make a dependable hardware encryption solution for financial services and other industries.

2. RSA

RSA is a public-key encryption algorithm and the standard for encrypting data sent over the internet. It also happens to be one of the methods used in our PGP and GPG programs.

Unlike Triple DES, RSA is considered an asymmetric algorithm due to its use of a pair of keys. You've got your public key, which is what we use to encrypt our message, and a private key to decrypt it. The result of RSA encryption is a huge batch of mumbo jumbo that takes attackers quite a bit of time and processing power to break.

3. Blowfish

Blowfish is yet another algorithm designed to replace DES. This symmetric cipher splits messages into blocks of 64 bits and encrypts them individually.

Blowfish is known for both its tremendous speed and overall effectiveness as many claim that it has never been defeated. Meanwhile, vendors have taken full advantage of its free availability in the public domain.

Blowfish can be found in software categories ranging from e-commerce platforms for securing payments to password management tools, where it used to protect passwords. It's definitely one of the more flexible encryption methods available.

4. Twofish

Computer security expert Bruce Schneier is the mastermind behind Blowfish and its successor Twofish. Keys used in this algorithm may be up to 256 bits in length and as a symmetric technique, only one key is needed.

Twofish is regarded as one of the fastest of its kind, and ideal for use in both hardware and software environments. Like Blowfish, Twofish is freely available to anyone who wants to use it. As a result, you'll find it bundled in encryption programs such as PhotoEncrypt, GPG, and the popular open source software TrueCrypt.

5. AES

The Advanced Encryption Standard (AES) is the algorithm trusted as the standard by the U.S. Government and numerous organizations.

Although it is extremely efficient in 128-bit form, AES also uses keys of 192 and 256 bits for heavy duty encryption purposes.

AES is largely considered impervious to all attacks, with the exception of brute force, which attempts to decipher messages using all possible combinations in the 128, 192, or 256-bit cipher. Still, security experts believe that AES will eventually be hailed the de facto standard for encrypting data in the private sector.

UNIT - II

PART A

1. Write down the purpose of S-Boxes in DES.

The role of the S-boxes in the function F is that the substitution consists of a set of eight S-boxes ,each of which accepts 6 bits as input and produces 4 bits as follows: The first and last bits of the input to box Si form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for Si. The middle four bits select one of the sixteen columns. The decimal value in the cell selected by the row and column is then

converted to its 4-bit representation to produce the output. For example, in S1,for input 011001,the row is 01 and the column is 1100.The value in row 1,column 12 is 9,so the output is 1001.

2. Define : Diffusion

Confusion refers to making the correlation between the key and the ciphertext as complex and intricate as possible. Diffusion refers to the property that the redundancy in the statistics of the plain text is "dissipated" in the statistics of the ciphertext. Or, The non uniformity in the distribution of the individual letters in the plaintext should be redistributd into the non uniformity in the distribution of much larger elements of the ciphertext, which is more difficult to detect.

3. Define :Replay attack.

Replay attacks are the network attacks in which an attacker spies the conversation between the sender and receiver and takes the authenticated information e.g. sharing key and then contact to the receiver with that key. In Replay attack the attacker gives the proof of his identity and authenticity.

Example:

Suppose in the communication of two parties A and B; A is sharing his key to B to prove his identity but in the meanwhile Attacker C eavesdrop the conversation between them and keeps the information which are needed to prove his identity to B. Later C contacts to B and prove its authenticity.

4. Write in short about DES (data encryption standard.).

Replay attacks are the network attacks in which an attacker spies the conversation between the sender and receiver and takes the authenticated information e.g. sharing key and then contact to the receiver with that key. In Replay attack the attacker gives the proof of his identity and authenticity.

Example:

Suppose in the communication of two parties A and B; A is sharing his key to B to prove his identity but in the meanwhile Attacker C eavesdrop the conversation between them and keeps the information which are needed to prove his identity to B. Later C contacts to B and prove its authenticity. Since DES is based on the Feistel Cipher, all that is required to specify DES is −

- Round function
- Key schedule
- Any additional processing − Initial and final permutation

5. Write about Block Ciphers.

A block cipher is a method of encrypting text (to produce ciphertext) in which a cryptographic key and algorithm are applied to a block of data (for example, 64 contiguous bits) at once as a group rather than to one bit at a time. The main alternative method, used much less frequently, is called the stream cipher.

6. Write in short about (advanced encryption standard)AES.

- The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES.
- A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.
- AES is an iterative rather than Feistel cipher. It is based on 'substitution–permutation network'. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).
- Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix
- Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

7. Write in short about triple DES.

Triple Data Encryption Standard (DES) is a type of computerized cryptography where block cipher algorithms are applied three times to each data block. The key size is increased in Triple DES to ensure additional security through encryption capabilities. Each block contains 64 bits of data. Three keys are referred to as bundle keys with 56 bits per key. There are three keying options in data encryption standards:

1. All keys being independent
2. Key 1 and key 2 being independent keys
3. All three keys being identical

Key option #3 is known as triple DES. The triple DES key length contains 168 bits but the key security falls to 112 bits.

8. What for the miller Rabin algorithm is used?

The Miller–Rabin primality test or Rabin–Miller primality test is a primality test. An algorithm which determines whether a given number is prime, similar to the Fermat primality test.

9. What are the strength of DES algorithm ?

- Avalanche effect: a slight(a char or bit) change in the plaintext will drastically change the cipher text.
- Completeness: each bit of ciphertext depends upon multiple bits of plaintext.
- It's not a group cipher, hence DES instances can be applied many times to a plaintext. (2DES 3DES).
- Trying (2^56) combinations is not that easy.

10. Write down the difference between Conventional encryption & Public key encryption.

For symmetric encryption, the same key is used to encrypt the message and to decrypt it. This key must be random, or cryptographically generated in a way that makes it look random.
For public-key encryption, instead the recipient generates two keys together, a public encryption key and a private decryption key. The message is encrypted with the public key, and can only be decrypted with the private key.

11. Explain the avalanche effect.

In cryptography, the **avalanche effect** refers to an attractive property of block ciphers and cryptographic hash functions algorithms. The **avalanche effect** is satisfied if: The output changes significantly (e.g., half the output bits flip) cause of a slight change in input (e.g., flipping a single bit). In the case of high-quality block ciphers, such a small change in either the key or the plaintext should cause a drastic change in the ciphertext.

12. What is differential cryptanalysis ?

Differential cryptanalysis is a general form of **cryptanalysis** applicable primarily to block ciphers, but also to stream ciphers and cryptographic hash functions. In the broadest sense, it is the study of how differences in information input can affect the resultant difference at the output.

13. What is difference between Rijindeal & AES ?

AES is a specification defined by the National Institute of Standards & Technology of the United States (NIST). AES is the successor of the Data Encryption Standard (DES).
AES has been announced in FIPS PUB 197 on November 26, 2001. Federal Information Processing Standards Publications (FIPS PUB) are issued by NIST after approval by the US Secretary of Commerce.
Rijndael is a symmetric key encryption algorithm created by Joan Daemen and Vincent Rijmen. It is a block cipher, with variable block size, variable key length & variable round number. Block length and key length can be independently specified to any multiple of 32 bits from 128 bits to 256 bits. The Rijndael cipher as been selected as the Advanced Encryption Standard (AES). In the Rijndael AES variant the block size is restricted to 128 bits and key length to 128, 192 or 256 bits only.

14. What is the purpose of the state array

A single 128-bit block is depicted as a square matrix of bytes. This block is copied into the State array, which is modified at each stage of encryption or decryption. After the final stage, State is copied to an output matrix

15. Write the difference between diffusion and confusion.

In diffusion, the statistical structure of the plain text is dissipated into long-range statistics of the cipher text. This is achieved by permutation. In confusion, the relationship between the statistics of the cipher text and the value of the encryption key is made complex. It is achieved by substitution.

16. What is Feistel cipher?

This cipher can be used to approximate the simple substitution cipher by utilizing the concept of a product cipher, which is the performing of two or more basic ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers.

17. What is a Shift rows?

- In shift row, a row shift moves an individual byte from one column to another, which is a linear distance of a multiple of 4 bytes.
- In Forward Shift Row, each row perform circular left shift. Second Row a 1-byte circular left shift is performed.
- Third Row a 2-byte circular left shift is performed. For the Fourth Row a 3-byte circular left shift is performed. In Inverse Shift Row, each row perform circular right shift.

18. How the key is expanded in AES

AES (Rijndael) uses a key schedule to expand a short key into a number of separate round keys. This is known as the Rijndael key schedule. The three AESvariants have a different number of rounds. Each variant requires a separate 128-bit round key for each round plus one more.

19. What is a meet-in-middle attack.

Meet-in-the-middle is a known attack that can exponentially reduce the number of bruteforce permutations required to decrypt text that has been encrypted by more than one key. Such an attack makes it much easier for an intruder to gain access to data.

The meet-in-the-middle attack targets block cipher cryptographic functions. The intruder applies brute force techniques to both the plaintext and cipher text of a block cipher. He then attempts to encrypt the plaintext according to various keys to achieve an intermediate cipher text (a text that has only been encrypted by one key). Simultaneously, he attempts to decrypt the cipher text according to various keys, seeking a block of intermediate ciphertext that is the same as the one achieved by encrypting the plaintext. If there is a match of intermediate ciphertext, it is highly probable that the key used to encrypt the plaintext and the key used to decrypt the ciphertext are the two encryption keys used for the block cipher.

20. What primitive operations are used in RC5

- Key expansion
- Encryption
- Decryption

21. Why do some block cipher modes of operation only use encryption while others use both encryption & decryption ?

Some modes of operation (eg CTR) work in such a way that only known values are ever encrypted, forming a stream of pseudo-random data that is then combined with the plaintext by a keyless reversible operation (often xor) to form the ciphertext. Other modes (eg CBC) directly encrypt secret (ie plaintext) values, meaning decryption is required to find out what the secret value was.

One of the biggest advantages of a scheme that does not require decryption is that it can be implemented in hardware with reduced footprint (ie it's smaller). Moreover, for block ciphers such as AES it can often be easier to implement efficient encryption than decryption because the internal coefficients have been optimized for this direction.

22. Define Quadratic sieve

The quadratic sieve algorithm (QS) is an integer factorization algorithm and, in practice, the second fastest method known (after the general number field sieve). It is still the fastest for integers under 100 decimal digits or so, and is considerably simpler than the number field sieve. It is a general-purpose factorization algorithm, meaning that its running time depends solely on the size of the integer to be factored, and not on special structure or properties.

23. Write about elliptic curve cryptography.

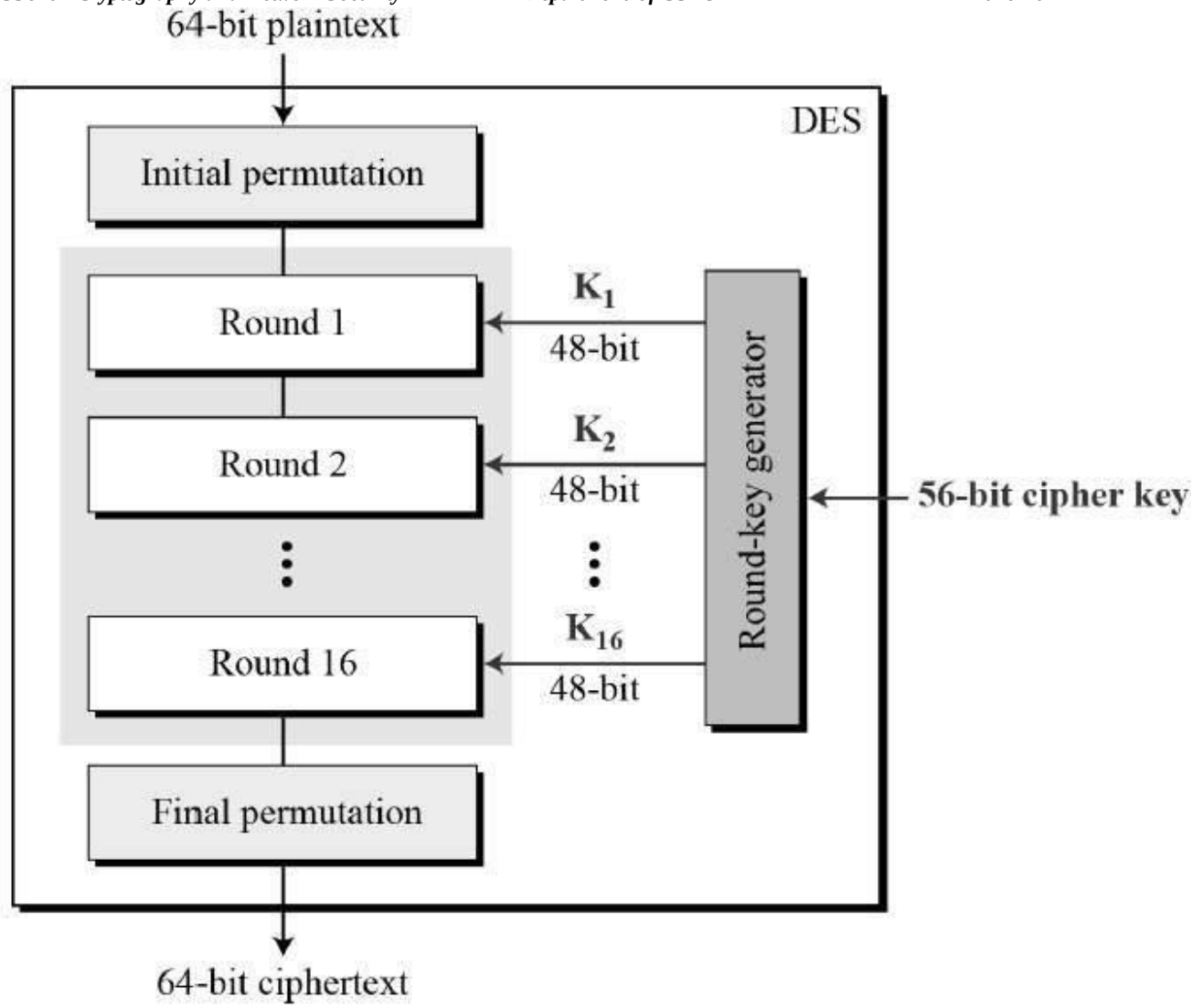
Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC requires smaller keys compared to non-ECC cryptography (based on plain Galois fields) to provide equivalent security. Elliptic curves are applicable for encryption, digital signatures, pseudo-random generators and other tasks. They are also used in several integer factorization algorithms that have applications in cryptography, such as Lenstra elliptic curve factorization.

PART B

1. Explain the key generation, encryption and decryption in DES algorithm in detail.

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following illustration −

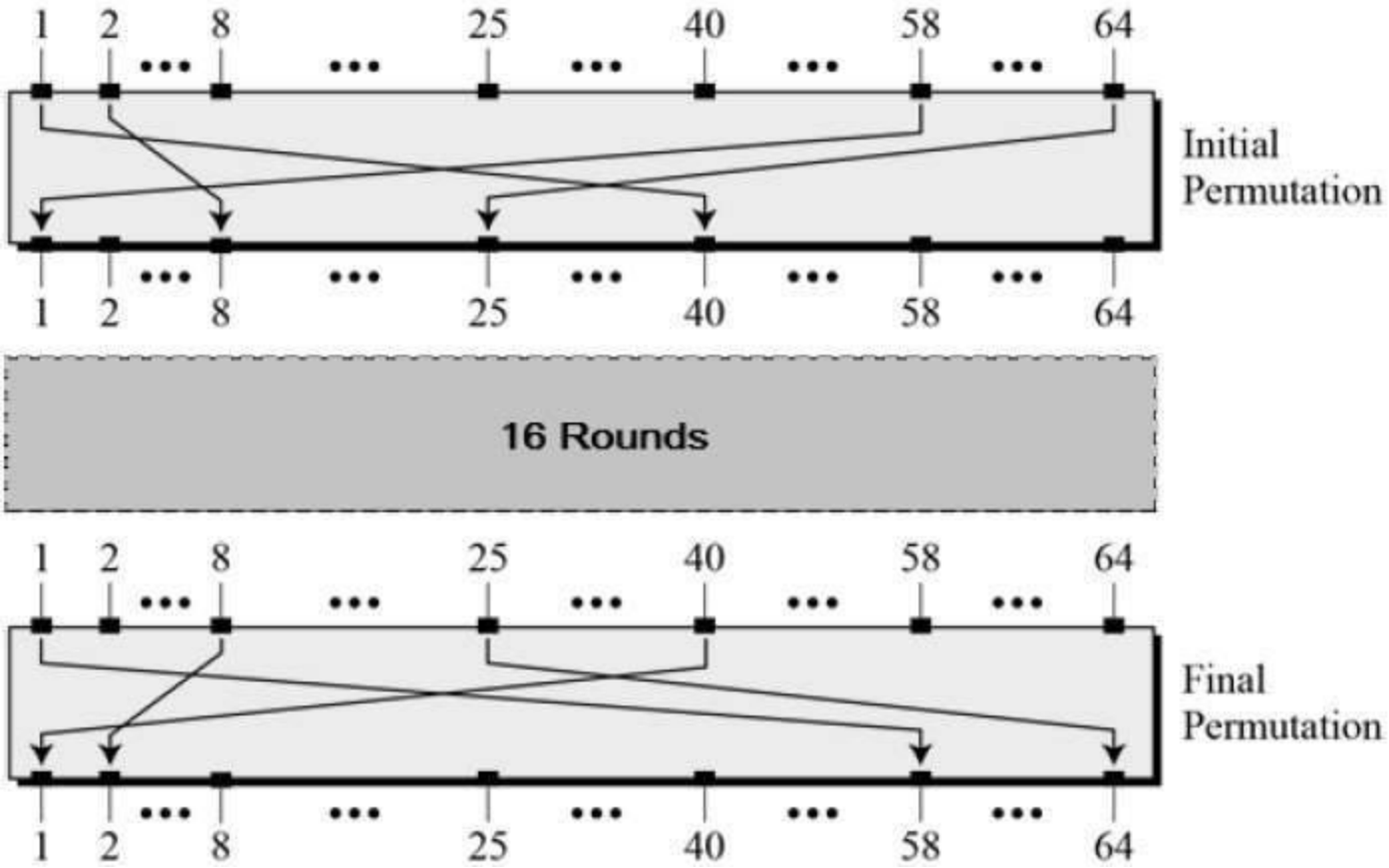


Since DES is based on the Feistel Cipher, all that is required to specify DES is −

- Round function
- Key schedule
- Any additional processing − Initial and final permutation

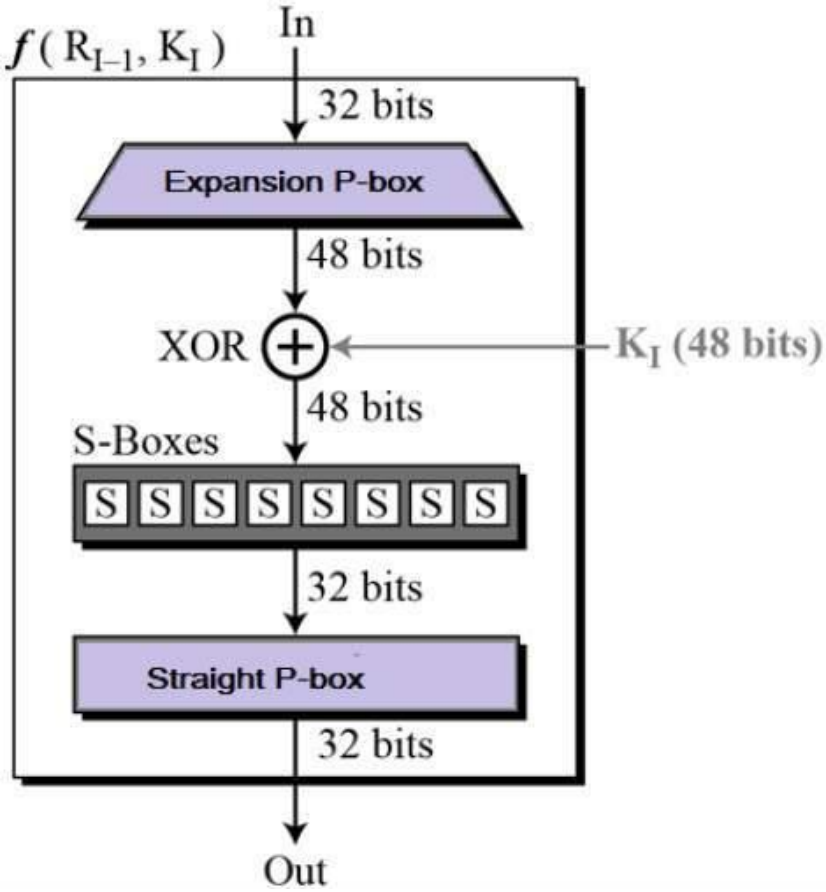
Initial and Final Permutation

The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are shown as follows −

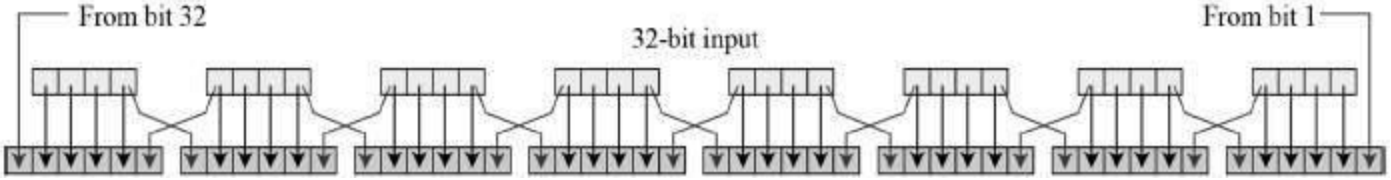


Round Function

The heart of this cipher is the DES function, *f*. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.



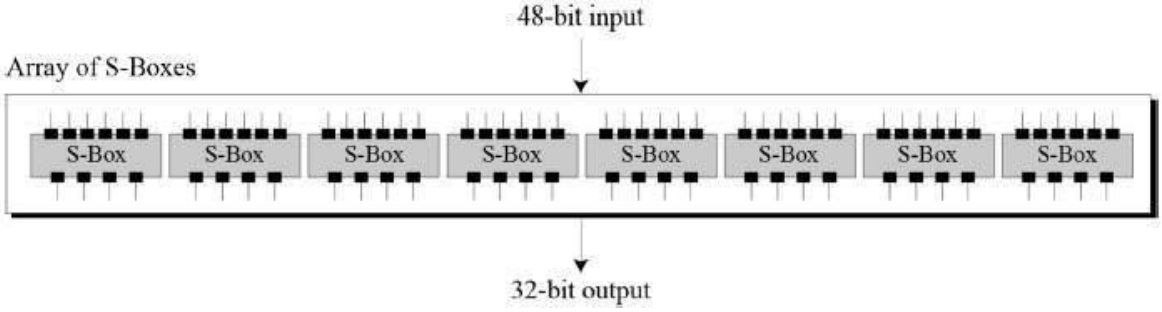
- **Expansion Permutation Box** − Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration −



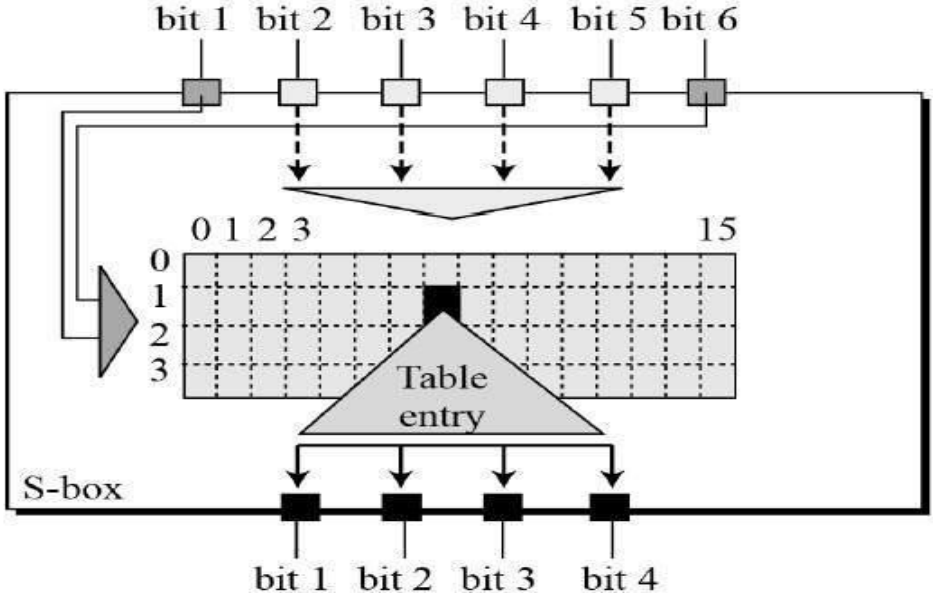
- The graphically depicted permutation logic is generally described as table in DES specification illustrated as shown −

32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

- **XOR (Whitener).** − After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.
- **Substitution Boxes.** − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration −



- The S-box rule is illustrated below −

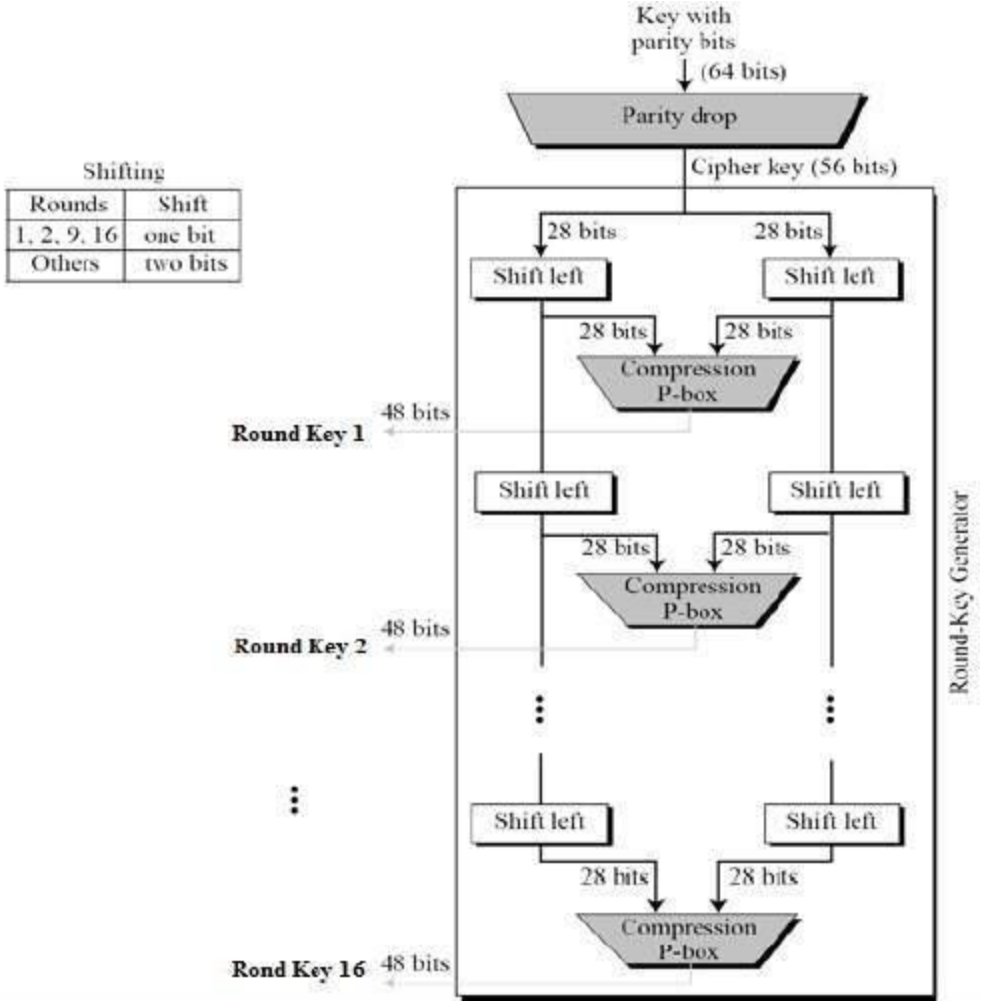


- There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32 bit section.
- **Straight Permutation** − The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration −



Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits

The logic for Parity drop, shifting, and Compression P-box is given in the DES description.

DES Analysis

The DES satisfies both the desired properties of block cipher. These two properties make cipher very strong.

- **Avalanche effect** − A small change in plaintext results in the very grate change in the ciphertext.
- **Completeness** − Each bit of ciphertext depends on many bits of plaintext.

During the last few years, cryptanalysis have found some weaknesses in DES when key selected are weak keys. These keys shall be avoided.

DES has proved to be a very well designed block cipher. There have been no significant cryptanalytic attacks on DES other than exhaustive key search.

2. Write the algorithm of RSA and explain with an example.

- by Rivest, Shamir & Adleman of MIT in 1977
- best known & widely used public-key scheme
- based on exponentiation in a finite (Galois) field over integers modulo a prime
  - nb. exponentiation takes $O((\log n)^3)$ operations (easy)
- uses large integers (eg. 1024 bits)
- security due to cost of factoring large numbers
  - nb. factorization takes $O(e^{\log n \log \log n})$ operations (hard)
- each user generates a public/private key pair by:
- selecting two large primes at random - p, q
- computing their system modulus n=p.q
  - note ø(n)=(p-1)(q-1)
- selecting at random the encryption key e
  - where $1<e<ø(n)$, $gcd(e,ø(n))=1$
- solve following equation to find decryption key d
  - $e.d=1 \bmod ø(n)$ and $0 \le d \le n$
- publish their public encryption key: PU={e,n}
- keep secret private decryption key: PR={d,n}
- to encrypt a message M the sender:
  - obtains **public key** of recipient PU={e,n}
  - computes: $C = M^e \bmod n$, where $0 \le M < n$
- to decrypt the ciphertext C the owner:
  - uses their private key PR={d,n}
  - computes: $M = C^d \bmod n$
- note that the message M must be smaller than the modulus n (block if needed)
- because of Euler's Theorem:
  - $a^{ø(n)} \bmod n = 1$ where $gcd(a,n)=1$
- in RSA have:
  - n=p.q
  - ø(n)=(p-1)(q-1)
  - carefully chose e & d to be inverses mod ø(n)
  - hence $e.d=1+k.ø(n)$ for some k
- hence :

$$C^d = M^{e.d} = M^{1+k.ø(n)} = M^1.(M^{ø(n)})^k$$
$$= M^1.(1)^k = M^1 = M \bmod n$$

The security of the RSA cryptosystem relies on the fact that, given n, it is believed to be infeasible to factor n into its two prime factors p and q. Although, the factoring problem appears to be very difficult (there is no present algorithm that can factor numbers efficiently), no one has shown this as yet.

Thus, the security of the RSA algorithm lies on the unproven claim that factoring is indeed a difficult problem. If one could factor n into its factors p and q, then one knows the value of $\varphi(n) = (p-1)(q-1)$ and one can determine $d = e^{-1} \bmod \varphi(n)$.

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\mathbf{f}(n) = (p-1)(q-1)$	
Select integer e	$gcd(\mathbf{f}(n), e) = 1;\ 1 < e < \mathbf{f}(n)$
Calculate d	$d \equiv e^{-1} \bmod \mathbf{f}(n)$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Encryption

Plaintext: $M < n$

Ciphertext: $C = M^e \pmod{n}$

Decryption

Plaintext: $M = C^d \pmod{n}$

Example:

- Choose p = 3 and q = 11
- Compute n = p * q = 3 * 11 = 33
- Compute φ(n) = (p - 1) * (q - 1) = 2 * 10 = 20
- Choose e such that 1 < e < φ(n) and e and n are coprime. Let e = 7
- Compute a value for d such that (d * e) % φ(n) = 1. One solution is d = 3 [(3 * 7) % 20 = 1]
- Public key is (e, n) => (7, 33)
- Private key is (d, n) => (3, 33)
- The encryption c= m^emod n
- *m = 2* is *c = 2^7 % 33 = 29*
- The decryption m=c^dmod n
- *c = 29* is *m = 29^3 % 33 = 2*

- by Rivest, Shamir & Adleman of MIT in 1977
- best known & widely used public-key scheme
- based on exponentiation in a finite (Galois) field over integers modulo a prime
  - nb. exponentiation takes $O((\log n)^3)$ operations (easy)
- uses large integers (eg. 1024 bits)
- security due to cost of factoring large numbers
  - nb. factorization takes $O(e^{\log n \log \log n})$ operations (hard)
- each user generates a public/private key pair by:
- selecting two large primes at random - p, q
- computing their system modulus n=p.q
  - note ø(n)=(p-1)(q-1)
- selecting at random the encryption key e
  - where 1<e<ø(n), gcd(e,ø(n))=1
- solve following equation to find decryption key d
  - e.d=1 mod ø(n) and 0≤d≤n
- publish their public encryption key: PU={e,n}
- keep secret private decryption key: PR={d,n}
- to encrypt a message M the sender:
  - obtains **public key** of recipient PU={e,n}
  - computes: C = M^e mod n, where 0≤M<n
- to decrypt the ciphertext C the owner:
  - uses their private key PR={d,n}
  - computes: M = C^d mod n
- note that the message M must be smaller than the modulus n (block if needed)
- because of Euler's Theorem:
  - $a^{ø(n)}$mod n = 1 where gcd(a,n)=1
- in RSA have:
  - n=p.q
  - ø(n)=(p-1)(q-1)
  - carefully chose e & d to be inverses mod ø(n)
  - hence e.d=1+k.ø(n) for some k
- hence :

$C^d = M^{e.d} = M^{1+k.ø(n)} = M^1.(M^{ø(n)})^k$

$= M^1.(1)^k = M^1 = M \bmod n$

The security of the RSA cryptosystem relies on the fact that, given n, it is believed to be infeasible to factor n into its two prime factors p and q. Although, the factoring problem appears to be very difficult (there is no present algorithm that can factor numbers efficiently), no one has shown this as yet.

Thus, the security of the RSA algorithm lies on the unproven claim that factoring is indeed a difficult problem. If one could factor n into its factors p and q, then one knows the value of $\varphi(n) = (p-1)(q-1)$ and one can determine $d = e^{-1} \bmod \varphi(n)$.

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \bmod \phi(n)$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod n$

Decryption

Plaintext:	$M = C^d \pmod n$

Example:

- Choose p = 3 and q = 11
- Compute n = p * q = 3 * 11 = 33
- Compute φ(n) = (p - 1) * (q - 1) = 2 * 10 = 20
- Choose e such that 1 < e < φ(n) and e and n are coprime. Let e = 7
- Compute a value for d such that (d * e) % φ(n) = 1. One solution is d = 3 [(3 * 7) % 20 = 1]
- Public key is (e, n) => (7, 33)
- Private key is (d, n) => (3, 33)
- The encryption $c = m^e \bmod n$
- *m = 2* is $c = 2^7 \% 33 = 29$
- The decryption $m = c^d \bmod n$
- *c = 29* is $m = 29^3 \% 33 = 2$

3. Write about the various mechanisms in Public key cryptography techniques.

Public-key refers to a cryptographic mechanism. It has been named public-key to differentiate it from the traditional and more intuitive cryptographic mechanism known as: symmetric-key, shared secret, secret-key and also called private-key. Symmetric-key cryptography is a mechanism by which the same key is used for both encrypting and decrypting; it is more intuitive because of its similarity with what you expect to use for locking and unlocking a door: the same key. This characteristic requires sophisticated mechanisms to securely distribute the secret-key to both parties2. Public-key on the other hand, introduces another concept involving key pairs: one for encrypting, the other for decrypting. This concept, as you will see below, is very clever and attractive, and provides a great deal of advantages over symmetric-key: • Simplified key distribution • Digital Signature • Long-term encryption However, it is important to note that symmetric-key still plays a major role in the implementation of a Public-key Infrastructure or PKI.

Public-key is commonly used to identify a cryptographic method that uses an asymmetric-key pair: a public-key and a private-key . Public-key encryption uses that key pair for encryption and decryption. The public-key is made public and is distributed widely and freely. The private-key is never distributed and must be kept secret.

Encryption is a mechanism by which a message is transformed so that only the sender and recipient can see. For instance, suppose that Alice wants to send a private message to Bob. To do so, she first needs Bob's public-key; since everybody can see his public-key, Bob can send it over the network in the clear without any concerns. Once Alice has Bob's public-key, she encrypts the message using Bob's public-key and sends it to Bob. Bob receives Alice's message and, using his private-key, decrypts it.

Digital signature is a mechanism by which a message is authenticated i.e. proving that a message is effectively coming from a given sender, much like a signature on a paper document. For instance, suppose that Alice wants to digitally sign a message to Bob. To do so, she uses her private-key to encrypt the

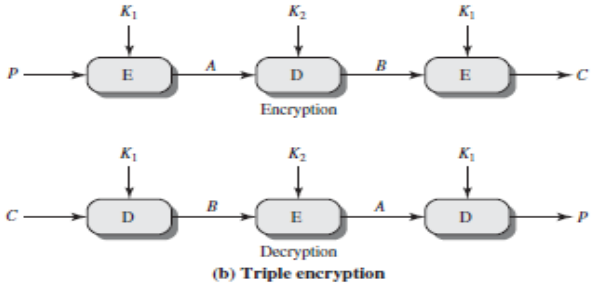
message; she then sends the message along with her public-key (typically, the public key is attached to the signed message). Since Alice's public-key is the only key that can decrypt that message, a successful decryption constitutes a Digital Signature Verification, meaning that there is no doubt that it is Alice's private key that encrypted the message.

The two previous paragraphs illustrate the encryption/decryption and signature/verification principles. Both encryption and digital signature can be combined, hence providing privacy and authentication.

4. Explain Triple DES in detail

Triple DES

- Applying DES three times is Triple DES
- Two types:
  - Triple des with two keys
  - Triple des with three keys



(b) Triple encryption

Triple-DES with Two-Keys

- In triple DES with 3 keys use 3 encryptions
  - would seem to need 3 distinct keys
- but can use 2 keys with E-D-E sequence

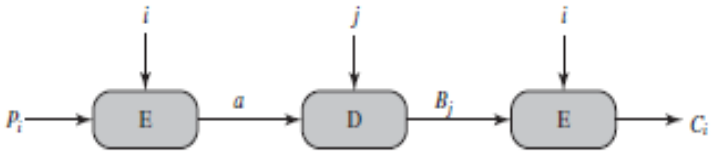
  C=E(K1,D(K2,E(K1,P)))

  P=D(K1,E(K2,D(K1,C)))

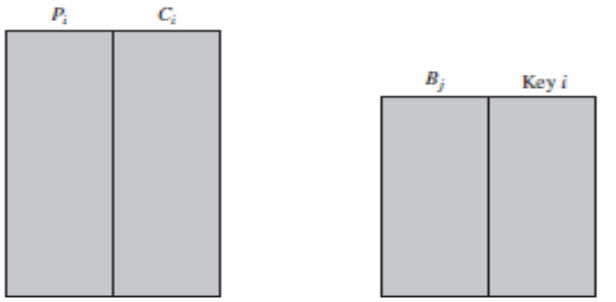
Triple-DES with Three-Keys

- although are no practical attacks on two-key Triple-DES have some indications
- can use Triple-DES with Three-Keys to avoid even these
  - $C = E_{K3}(D_{K2}(E_{K1}(P)))$
- has been adopted by some Internet applications, eg PGP, S/MIME

Triple-DES with Two-Keys known plain text attack



(a) Two-key triple encryption with candidate pair of keys



(b) Table of *n* known plaintext-ciphertext pairs, sorted on *P*

(c) Table of intermediate values and candidate keys

Figure 6.2 Known-Plaintext Attack on Triple DES

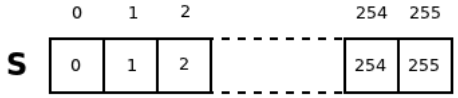
5. Explain RC5 algorithm.

RC4 is a stream cipher.Here encryption or decryption happens bit by bit.

- The RC4 algorithm generates a pseudo-random **keystream** .
- The keystream is XORed with plaintext to produce the ciphertext .
- It is called pseudo- random because it generates a sequence of numbers that only approximates the properties of random numbers.
- There is a variable length key consisting of 1 to 256 bytes.
- The key is used to initialize 256-byte state vector with elements identified as s[0],s[1]…s[256].
- To perform encryption and decryption one of the 256 bytes of S is selected and processed.
- The resulting ouput of K bits are permuted once again.
- 2 processes are involved (a) Initialisation of S (b) Stream generation

Initialisation of S:

- Choose a key (K) of length between 1 and 256 bytes.
- Set the values in the state vector S equal to the values from 0 to 255 in ascending order



Once the S array is initialized, the next step consist in shuffling the array using the key to make it a permutation array.

Create temporary array T.if the length of the key K is 256 bytes copy K into T.After copying the remaining positions are filled with values of K again.

```
for i from 0 to 255
    S[i] := i
  T[i]=K[I MOD KEYLEN];
endfor
```

- After this T array is used to produce initial permutation of S.
- For this purpose a loop executes iterating from 0 to 255.
- The byte at the position s[i] is swapped with another byte in S array as per arrangement decided by T[i].
- To do so, we simply iterate 256 times the following actions after initializing i and j to
- compute j = j + S[i] + key[i mod keylength]
- swap S[i] and S[j]
- increment i
- Once it has reached 256 (the 256 iterations were completed), the S array has been properly initialized.

```
for i from 0 to 255
    j := (j + S[i] + key[i mod keylength]) mod 256 swap
    values of S[i] and S[j]
endfor
```

- Now that the S array is generated, it is used in the next step of the RC4 algorithm to generate the keystream.

Stream generation algorithm

This step of the algorithm consists in generating a keystream of the size of the message to encrypt. This algorithm enables us to generate a keystream of any size.

To do so, we first initialize the two indexes to 0 and we then start the generation of the keystream one byte at a time until we reached the size of the message to encrypt. For each new byte to compute we do the following actions:

- Compute new value of i and j:

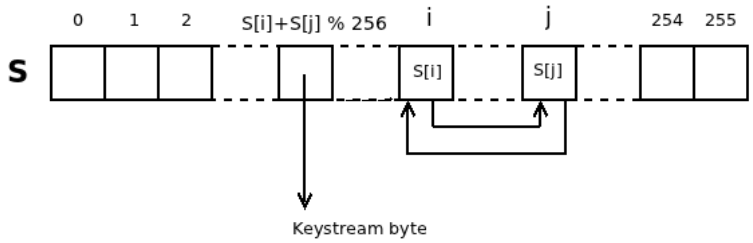
  i := (i + 1) % 256

  j := (j + S[i]) % 256

Swap S[i] and S[j] to have a dynamic state (it makes it obviously harder to crack than if the state was

computed only once and use for the generation of the whole keystream)

Retrieve the next byte of the keystream from the S array at the index S[i]+S[j]% 256

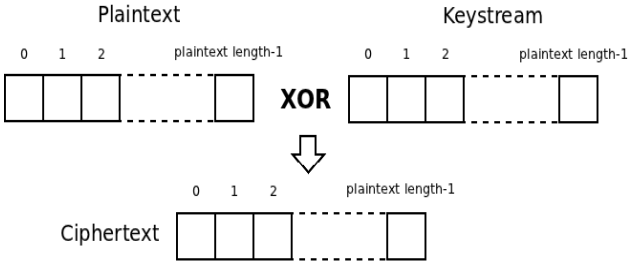


Here is some pseudo-code corresponding to the pseudo-random generation algorithm:

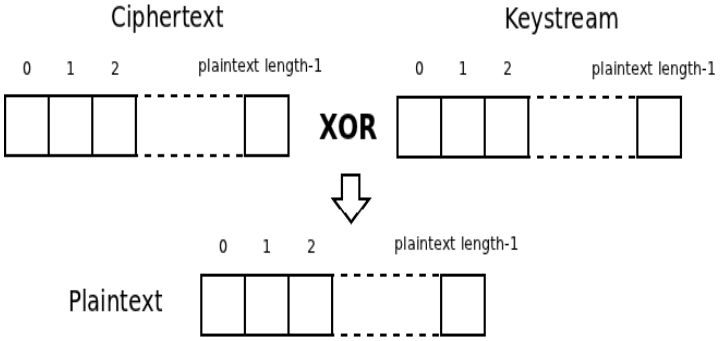
```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap values of S[i] and S[j]
    K := S[(S[i] + S[j]) mod 256]
    output K
endwhile
```

Encryption and decryption

Once the keystream has been generated, the encryption of the plaintext is really simple: it simply consists of a XOR between the plaintext and the keystream. See below an illustration of the encryption:



As for the decryption, it is as simple as the encryption, we only have to do the opposite: XOR the ciphertext with the keystream.



6. Write in detail about the Diffie - hellman key exchange algorithm

A simple public-key algorithm is Diffie-Hellman key exchange.This protocol enables two users to establish a secret key using a public-key scheme based on discrete logarithms. The protocol is secure only if the authenticity of the two participants can be established. The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms

ALGORITHM

For this scheme, there are two publicly known numbers: a prime number q and an integer α, that is a primitive root of q. Suppose the users **A** and **B** wish to exchange a key.

- User **A** selects a random integer $x_A < q$ and computes $y_A = (\alpha)^{xA} \bmod q$
- Similarly, user **B** independently selects a random integer $x_B < q$ and computes $y_B = (\alpha)^{xB} \bmod q$.
- Each side keeps the **X** value private and makes the **Y** value available publicly to the other side. User A computes the key as $\mathbf{K = (y_B)^{xA} \bmod q}$.
- user B computes the key as $\mathbf{K = y_A^{xB} \bmod q}$.

These two calculations produce identical results:

$\mathbf{K = (y_B)^{xA} \bmod q}$

$=(\alpha^{xB} \bmod q)^{xA} \bmod q$.

$=(\alpha^{xB})^{xA} \bmod q$.

$=(\alpha^{xA})^{xB} \bmod q$.

$=(\alpha^{xA} \bmod q)^{xB} \bmod q$.

$= \mathbf{y_A^{xB} \bmod q}$.

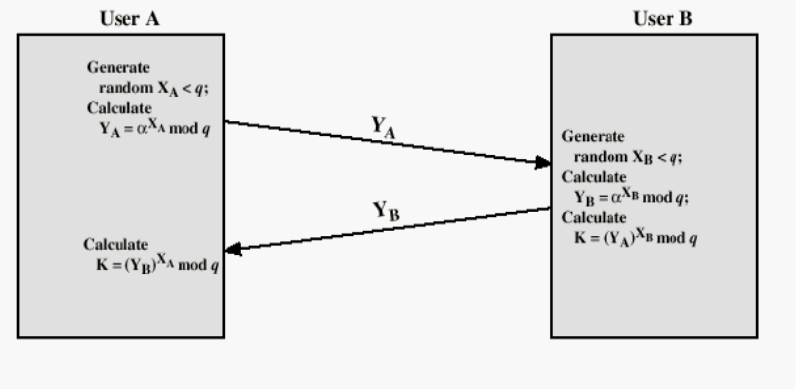


Fig: Diffie Hellman Key exchange

man-in-the-middle attack

1.Darth prepares for the attack by generating two random private keys X_{D1} and $X_{D2}$and then computing the corresponding public keys Y_{D1} and Y_{D2}

2. Alice transmits Y_A to Bob.

3. Darth intercepts Y_A and transmits Y_{D1} to Bob. Darth also calculates K2= $(Y_A)^{XD2}$ MOD Q.

4. Bob receives Y_{D1} and calculates K1= $(Y_{D1})^{X_E}$ MOD Q.

5. Bob transmits X_A to Alice.

6. Darth intercepts X_A and transmits Y_{D2} to Alice. Darth calculates K1= $(Y_B)^{X_{D1}}$MOD Q.

7. Alice receives Y_{D2} and calculates K2= $(Y_{D2})^{X_A}$ MOD Q..At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key and Alice and Darth share secret key .

All future communication between Bob and Alice is compromised in the following way.

1. Alice sends an encrypted message M:E(K2,M)

2. Darth intercepts the encrypted message and decrypts it to recover M .

3. Darth sends Bob E(K1,M) OR E(K1,M') whereM' is any message. In the first case, Darth simply wants to eavesdrop on the communication without altering it. In the second case, Darth wants to modify the message going to Bob.

7. Write about elliptic curve cryptography.

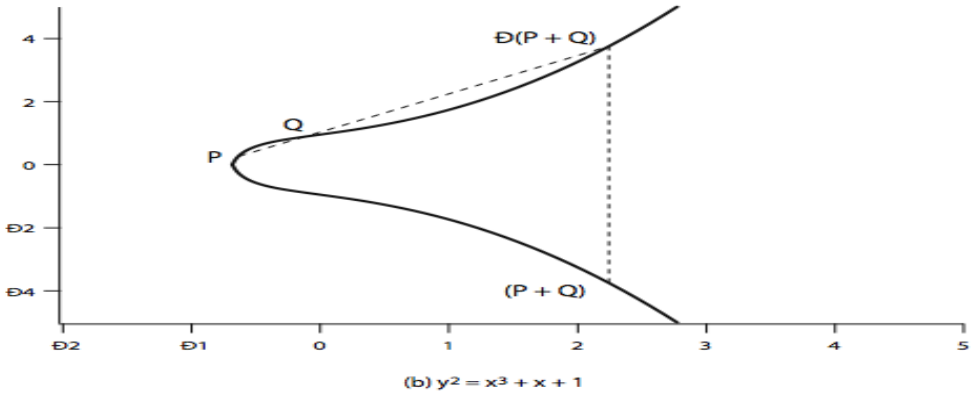
Elliptic Curve Cryptography

A major issue with the use of Public-Key Cryptography, is the size of numbers used, and hence keys being stored. Recently, an alternate approach has emerged, elliptic curve cryptography (ECC), which performs the computations using elliptic curve arithmetic instead of integer or polynomial arithmetic.

Majority of public-key crypto (RSA, D-H) use either integer or polynomial arithmetic with very large numbers/polynomials. It imposes a significant load in storing and processing keys and messages. An alternative is to use elliptic curves; it offers same security with smaller bit sizes.

Real Elliptic Curves

- an elliptic curve is defined by an equation in two variables x & y, with coefficients
- consider a cubic elliptic curve of form
  - $y^2 = x^3 + ax + b$
  - where x,y,a,b are all real numbers
  - also define zero point O
- have addition operation for elliptic curve
  - geometrically sum of Q+R is reflection of intersection R



(b) $y^2 = x^3 + x + 1$

Finite Elliptic Curves

Elliptic curve cryptography makes use of elliptic curves in which the variables and coefficients are all restricted to elements of a finite field. Two families of elliptic curves are used in cryptographic applications: prime curves over Z_p (best for software use), and binary curves over $GF(2^m)$ (best for hardware use).

- prime curves $E_p(a,b)$ defined over Z_p
  - use integers modulo a prime
  - best in software
- binary curves $E_{2m}(a,b)$ defined over $GF(2^n)$
  - use polynomials with binary coefficients
  - best in hardware

Elliptic Curve Cryptography uses addition as an analog of modulo multiply, and repeated addition as an analog of modulo exponentiation. The "hard" problem is the elliptic curve logarithm problem.

- Q=kP, where Q,P belong to a prime curve
- is "easy" to compute Q given k,P
- but "hard" to find k given Q,P
- known as the elliptic curve logarithm problem

Elliptic Curve over GF(p)

Let GF(p) be a finite field, p > 3, and let a, b ∈ GF(p) are constant such that

$$4a^3 + 27b^2 \equiv 0 \pmod p.$$

An elliptic curve, $E_{(a,b)}(GF(p))$, is defined as the set of points (x,y) ∈ GF(p) * GF(p) which satisfy the equation

$$y^2 \equiv x^3 + ax + b \pmod p$$

together with a special point, O, called the point at infinity.

P and Q be two points on $E_{(a,b)}(GF(p))$ and O is the point at infinity.

- P+O = O+P = P
- If $P = (x_1,y_1)$ then $-P = (x_1,-y_1)$ and P + (-P) = O.
- If $P = (x_1,y_1)$ and $Q = (x_2,y_2)$, and P and Q are not O. then $P + Q = (x_3,y_3)$ where

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$
$$\text{and} \quad \lambda = (y_2-y_1)/(x_2-x_1) \quad \text{if } P \neq Q$$
$$\lambda = (3x_1^2+a)/2y_1 \quad \text{if } P = Q$$

Elliptic Curve over $GF(2^m)$ for some $m \geq 1$

- Elliptic curve $E_{(a,b)}(GF(2^m))$ is defined to be the set of points (x,y) ∈ $GF(2^m)$ * $GF(2^m)$ which satisfy the equation

$$y^2 + xy = x^3 + ax^2 + b;$$

where a, b ∈ $GF(2^m)$ and b≠0, together with the point on the curve at infinity, O.

- The points on an elliptic curve form an abelian group under a well defined group operation. The identity of the group operation is the point O.

P and Q be two points on $E_{(a,b)}(GF(2^m))$ and O is the point at infinity.

- P+O = O+P = P
- If $P = (x_1,y_1)$ then $-P = (x_1,-y_1)$ and P + (-P) = O.
- If $P = (x_1,y_1)$ and $Q = (x_2,y_2)$, and P and Q are not O.

then $P + Q = (x_3,y_3)$,
where $y_3 = \lambda(x_1 + x_3) + x_3 + y_1$
and $\lambda = (y_1+y_2)/(x_1+x_2)$ if $P \neq Q$
$\lambda = (x_1 y_1 + x_1)/ y_1$ if P = Q

ECC Diffie-Hellman

- can do key exchange analogous to D-H
- users select a suitable curve $E_p(a,b)$
- select base point $G=(x_1,y_1)$
  - with large order n such that . nG=0
- A & B select private keys $n_A<n$, $n_B<n$
- compute public keys: $P_A=n_AG$, $P_B=n_BG$
- compute shared key: $K=n_AP_B$, $K=n_BP_A$
  - same since $K=n_An_BG$

ECC Encryption/Decryption

Several approaches to encryption/decryption using elliptic curves have been analyzed . This one is an analog of the ElGamal public-key encryption algorithm. The sender must first encode any message M as a point on the elliptic curve P_m . Note that the ciphertext is a pair of points on the elliptic curve. The sender masks the message using random k, but also sends along a "clue" allowing the receiver who know the private-key to recover k and hence the message. For an attacker to recover the message, the attacker would have to compute k given G and kG, which is assumed hard.

- must first encode any message M as a point on the elliptic curve P_m
- select suitable curve & point G as in D-H
- each user chooses private key $n_A<n$
- and computes public key $P_A=n_AG$
- to encrypt P_m : $C_m=\{kG, P_m+kP_b\}$, k random
- decrypt C_m compute:
  $P_m+kP_b-n_B(kG) = P_m+k(n_BG)-n_B(kG) = P_m$

ECC Security

The security of ECC depends on how difficult it is to determine k given kP and P. This is referred to as the elliptic curve logarithm problem. The fastest known technique for taking the elliptic curve logarithm is known as the Pollard rho method. Compared to factoring integers or polynomials, can use much smaller numbers for equivalent levels of security.

8. Write in detail about the principles of Public key cryptosystems.

KEY MANAGEMENT

• Public-key encryption helps address key distribution problems

Have two aspects:

o Distribution of public keys

o Use of public-key encryption to distribute secret keys

Distribution of Public Keys

Distribution of Public Keys can be done in one of the four ways:

- Public announcement
- Publicly available directory
- Public-key authority
- Public-key certificates

Public Announcement

• Users distribute public keys to recipients or broadcast to community at large

o eg. Append PGP keys to email messages or post to news groups or email list

• Major weakness is forgery
o Anyone can create a key claiming to be someone else and broadcast it
o Until forgery is discovered can masquerade as claimed user

Publicly Available Directory

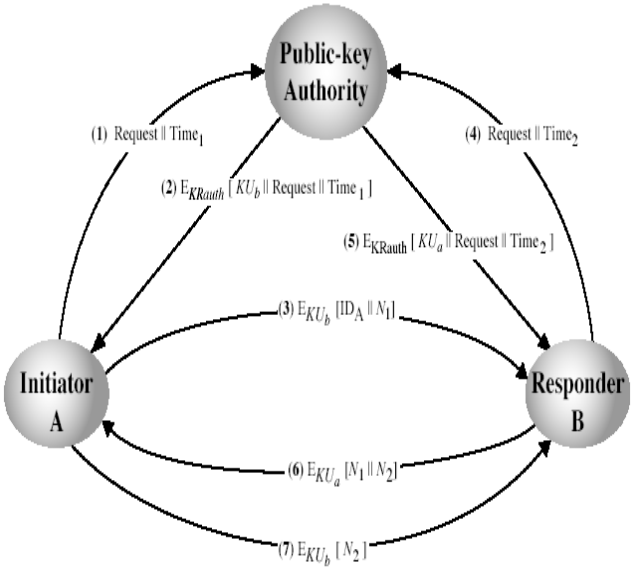
• Can obtain greater security by registering keys with a public directory
• Directory must be trusted with properties:
o Contains {name, public-key} entries
o Participants register securely with directory
o Participants can replace key at any time
o Directory is periodically published
o Directory can be accessed electronically
• Still vulnerable to tampering or forgery

Public-Key Authority

• Improve security by tightening control over distribution of keys from directory
• Has properties of directory
• Requires users to know public key for the directory
• Users interact with directory to obtain any desired public key securely
o Does require real-time access to directory when keys are needed

Public-Key Certificates

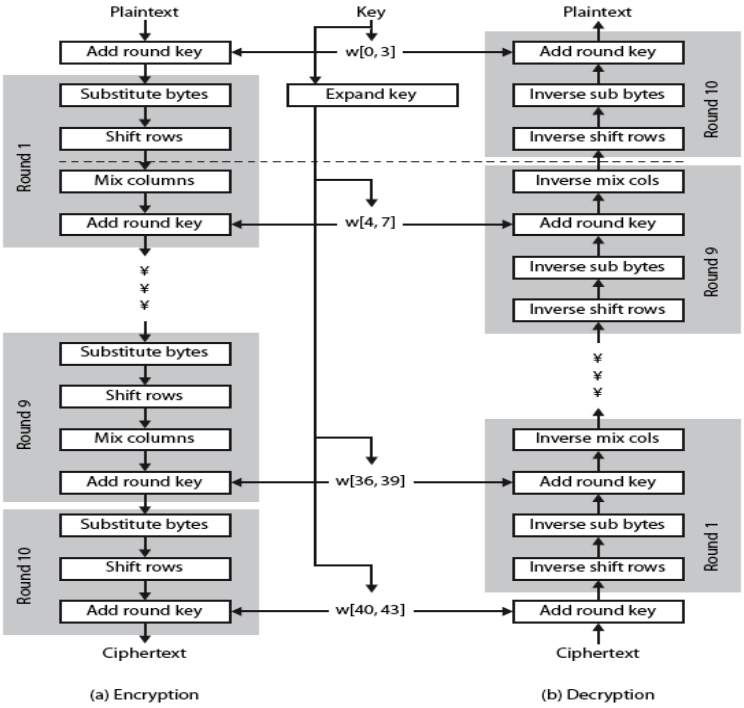
• Certificates allow key exchange without real-time access to public-key authority
• A certificate binds **identity** to **public key**
o Usually with other info such as period of validity, rights of use etc
• With all contents **signed** by a trusted Public-Key or Certificate Authority (CA)
Can be verified by anyone who knows the public-key authorities public-key



9. Explain in detail about Advanced Encryption standard in detail.

- initial criteria:
  - security – effort for practical cryptanalysis
  - cost – in terms of computational efficiency
  - algorithm & implementation characteristics
- final criteria
  - general security
  - ease of software & hardware implementation
  - implementation attacks
  - flexibility (in en/decrypt, keying, other factors)
- designed by Rijmen-Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data

- an **iterative** rather than **feistel** cipher
  - processes data as block of 4 columns of 4 bytes
  - operates on entire data block in every round
- designed to be:
  - resistant against known attacks
  - speed and code compactness on many CPUs
  - design simplicity
- data block of 4 columns of 4 bytes is state
- key is expanded to array of words
- has 9/11/13 rounds in which state undergoes:
  - byte substitution (1 S-box used on every byte)
  - shift rows (permute bytes between groups/columns)
  - mix columns (subs using matrix multipy of groups)
  - add round key (XOR state with key material)
  - view as alternating XOR key & scramble data bytes
- initial XOR key material & incomplete last round
- with fast XOR & table lookup implementation



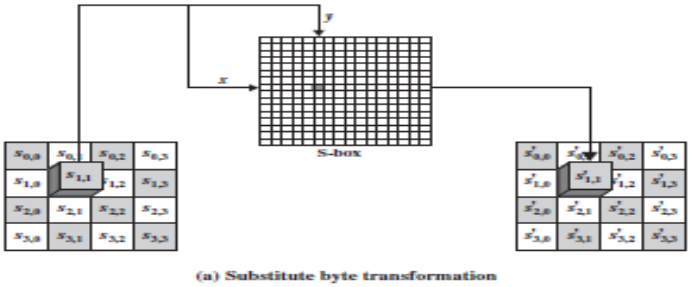
The input to the encryption and decryption algorithms is a single 128-bit block.In FIPS PUB 197, this block is depicted as a square matrix of bytes.This blockis copied into the **State** array, which is modified at each stage of encryption or decryption.After the final stage, **State** is copied to an output matrix. Similarly, the key is depicted as a square matrix of bytes.This key is then expanded into an array of key schedule words.Each word is four bytes, and the total key schedule is 44 words for the 128-bit key. Note that the ordering of bytes within a matrix is by column.The first four bytes of a 128-bit plaintext input to the encryption cipher occupy the first column of the **in** matrix, the second four bytes occupy the second column, and so on. Similarly, the first four bytes of the expanded key, which form a word, occupy the first column of the **w** matrix.

The cipher consists of rounds, where the number of rounds depends on thekey length.The first rounds consist of four distinct transformation functions: SubBytes, ShiftRows, MixColumns, and AddRoundKey.The final round contains only three transformations, and there is a initial single transformation (AddRoundKey) before the first round,

Substitute Bytes Transformation

forward substitute byte transformation, called SubBytes, is a simple table lookup .AESdefines a matrix of byte values, called an S-box ,that contains a permutation of all possible 256 8-bit values. Each individual byte of **State** is mapped into a new byte in the following way: The leftmost 4 bits of the byte are used as a row value

and the rightmost 4 bits are used as a column value.These row and column values serve as indexes into the S-box to select a unique 8-bit output value. For example, the hexadecimal value3 {95} references row 9, column 5 of the S-box, which contains the value .{2A}



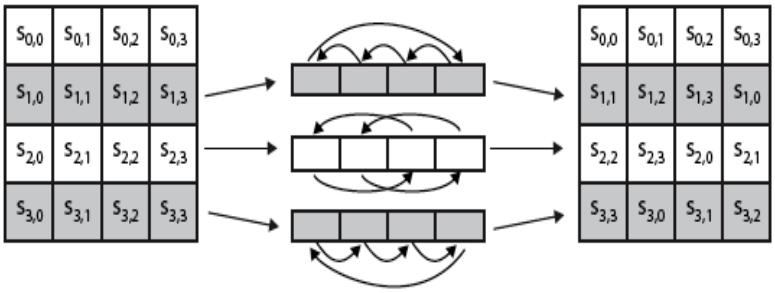
(a) Substitute byte transformation

ShiftRows Transformation

FORWARD AND INVERSE TRANSFORMATIONS

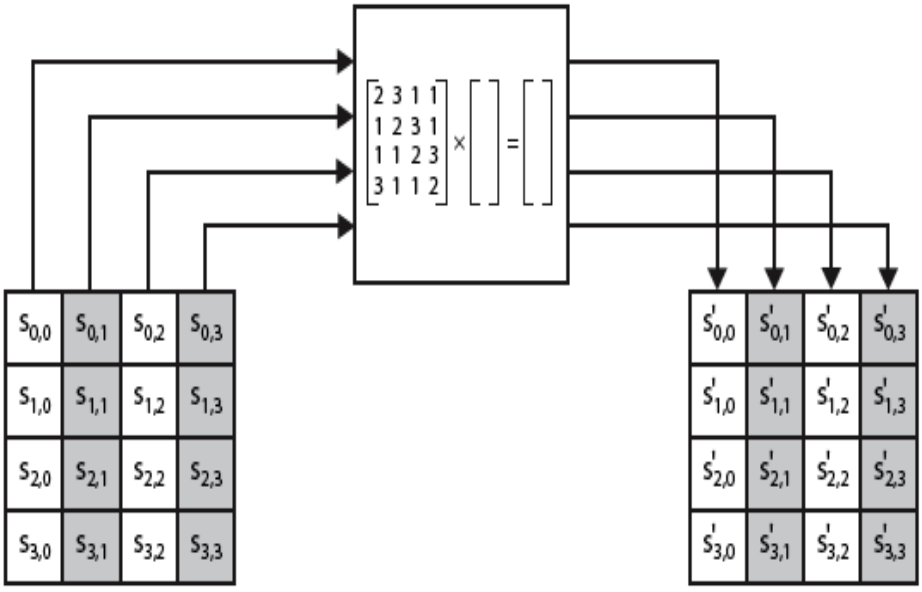
The **forward shift row transformation**, also called ShiftRows.The first row of **State** is not altered. For the second row, a 1-byte circular left shift is performed. For the third row, a 2-byte circular left shift is performed. For the fourth row, a 3-byte circular left shift is performed.The following is an example of ShiftRows.

The **inverse shift row transformation**, called InvShiftRows, performs the circular shifts in the opposite direction for each of the last three rows, with a 1-byte circular right shift for the second row, and so on.



MixColumns Transformation

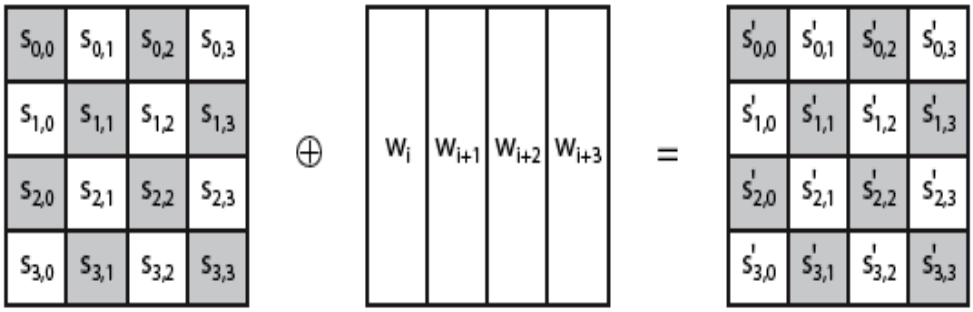
The **forward mix column transformation**, called MixColumns, operates on each column individually. Each byte of a column is mapped into a new value that is a function of all four bytes in that column. The transformation can be defined by the following matrix multiplication on **State** Each element in the product matrix is the sum of products of elements of one row and one column. In this case, the individual additions and multiplications5 are performed



➢

AddRoundKey Transformation

FORWARD AND INVERSE TRANSFORMATIONS In the **forward add round key transformation**, called AddRoundKey, the 128 bits of **State** are bitwise XORed with the 128 bits of the round key.the operation is viewed as a columnwise operation between the 4 bytes of a **State** column and one word of the round key; it can also be viewed as a byte-level operation. The first matrix is **State**, and the second matrix is the round key.



10. Explain about Blowfish algorithm.

An encryption algorithm plays an important role in securing the data in storing or transferring it. The encryption algorithms are categorized into Symmetric (secret) and Asymmetric (public) keys encryption.

- In Symmetric key encryption or secret key encryption, only one key is used for both encryption and decryption of data.
  Eg: Data encryption standard(DES), Triple DES, Advanced Encryption Standard(AES) and Blowfish Encryption Algorithm
- In asymmetric key encryption or public key encryption uses two keys, one for encryption and other for decryption.
  Eg: RSA

Blowfish Encryption Algorithm

Blowfish was designed in 1993 by Bruce Scheier as a fast, alternative to existing encryption algorithms such AES, DES and 3 DES etc.

Blowfish is a symmetric block encryption algorithm designed in consideration with,

- **Fast :** It encrypts data on large 32-bit microprocessors at a rate of 26 clock cycles per byte.
- **Compact:** It can run in less than 5K of memory.
- **Simple:** Ituses addition, XOR, lookup table with 32-bit operands.
- **Secure:** The key length is variable ,it can be in the range of 32~448 bits: default 128 bits key length.
- It is suitable for applications where the key does not change often, like communication link or an automatic file encryptor.
- Unpatented and royality-free.

Description of Algorithm:

Blowfish symmetric block cipher algorithm encrypts block data of 64-bits at a time.it will follows the feistel network and this algorithm is divided into two parts.

1. Key-expansion
2. Data Encryption

1 Key-expansion:

It will converts a key of at most 448 bits into several subkey arrays totaling 4168 bytes. Blowfish uses large number of subkeys.

These keys are generate earlier to any data encryption or decryption.

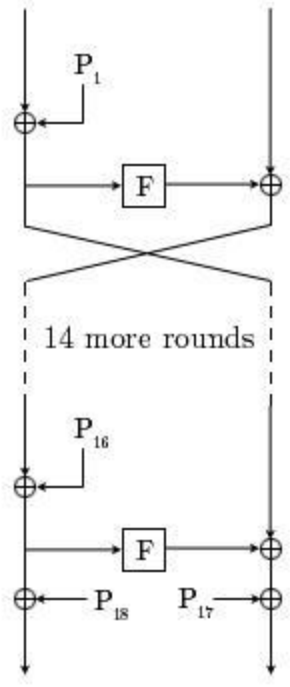


Fig 1: The Feistel structure of Blowfish

The p-array consists of 18, 32-bit subkeys:

P1,P2,……,P18

Four 32-bit S-Boxes consists of 256 entries each:

S1,0, S1,1,……. S1,255

S2,0, S2,1,…….. S2,255

S3,0, S3,1,…….. S3,255

S4,0, S4,1,.....S4,255

Generating the Subkeys:

The subkeys are calculated using the Blowfish algorithm:

1. Initialize first the P-array and then the four S-boxes, in order, with a fixed string. This string consists of the hexadecimal digits of pi (less the initial 3): P1 = 0x243f6a88, P2 = 0x85a308d3, P3 = 0x13198a2e, P4 = 0x03707344, etc.
2. XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the key bits until the entire P-array has been XORed with key bits. (For every short key, there is at least one equivalent longer key; for example, if A is a 64-bit key, then AA, AAA, etc., are equivalent keys.)
3. Encrypt the all-zero string with the Blowfish algorithm, using the subkeys described in steps (1) and (2).
4. Replace P1 and P2 with the output of step (3).
5. Encrypt the output of step (3) using the Blowfish algorithm with the modified subkeys.
6. Replace P3 and P4 with the output of step (5).
7. Continue the process, replacing all entries of the P array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm.

In total, 521 iterations are required to generate all required subkeys. Applications can store the subkeys rather than execute this derivation process multiple times.

2 Data Encryption:

It is having a function to iterate 16 times of network. Each round consists of key-dependent permutation and a key and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookup tables for each round.

Algorithm:Blowfish Encryption

Divide x into two 32-bit halves: xL, xR

For i = 1to 16:

xL = XL XOR Pi

xR = F(XL) XOR xR

Swap XL and xR

Swap XL and xR (Undo the last swap.)

xR = xR XOR P17

xL = xL XOR P18

Recombine xL and xR

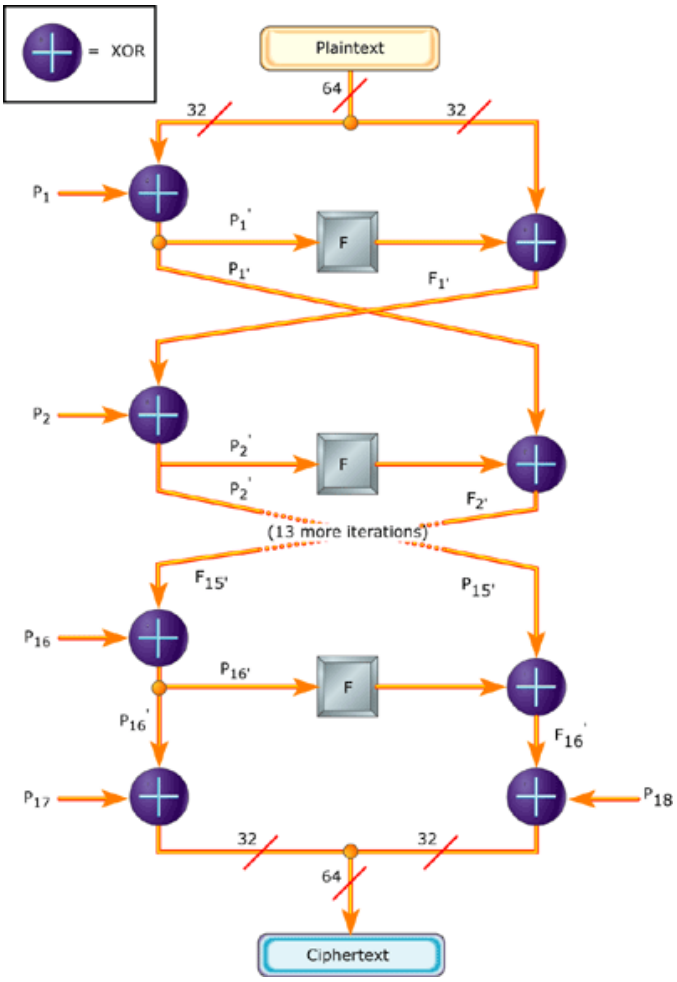


Fig 2: Blowfish Encrypt**ion**

UNIT-3
PART-A

1. What is meant by Authentication?

Authentication is a mechanism or service used to verify the integrity of a message. Message authentication assures that data received are exactly as sent by (i.e., contain no modification, insertion, deletion, or replay) and that the purported identity of the sender is valid.

2. What are common techniques for message authentication?

Message Authentication Code (MAC)

MAC is an algorithm that requires the use of a secret key. A MAC takes a variable-length message and a secret key as input and produces an authentication code. A recipient in possession of the secret key can generate an authentication code to verify the integrity of the message.

Hash function

A Hash function maps a variable-length message into a fixed length hash value, or message digest. For message authentication, a secure hash function must be combined in some fashion with a secret key.

3. List out different Authentication Requirements?

1. Disclosure
2. Traffic analysis
3. Masquerade
4. Content modification
5. Sequence modification
6. Timing modification
7. Source repudiation
8. Destination repudiation

4. Contrast the DSS for generating digital signatures to that used with RSA.

In Digital signature standard, it provides more secure authentication for sending data so that, it can be used for RSA also. Now the RSA easy to implement with secure way.

5. To what extent is MD5 stronger than MD4? State the reason.

The following changes are made in MD5 to make it stronger than MD4:

- A fourth round has been added.
- Each step now has a unique additive constant.
- Each step now adds in the result of the previous step. This promotes a faster "avalanche effect".
- The order in which input words are accessed in rounds 2 and 3 is changed, to make these patterns less like each other.

6. Define digital signature.

A digit signature is a message digest derived from the original message to provide authenticity of the document, has following important properties:

- The digest is difficult to reverse
- It is hard to find a different message that computed to the same digest value.

7. List out the message authentication function

- Message encryption: The ciphertext of the entire message serves as its authenticator
- Message authentication code (MAC): A function of the message and a secret key that produces a fixed-length value that serves as the authenticator
- Hash function: A function that maps a message of any length into a fixed-length hash value, which serves as the authenticator

8. Define Message Authentication Code (MAC).

A MAC, also known as a cryptographic checksum, is generated by a function C of the form

MAC = C(K, M)

Where M is a variable-length message, K is a secret key shared only by sender and receiver, and C(K, M) is the fixed-length authenticator. The MAC is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by recomputing the MAC.

9. Write a simple 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:

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

Where

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

10. What are birthday attacks?

A Birthday attack is a cryptanalytic technique. Birthday attacks can be used to find collision in a cryptographic function. For instance suppose we have a hash function which, when supplied with a random input returns one of K equally likely values. By repeatedly evaluating the function on $1.2(\sqrt{k})$ different inputs, it is likely we will find some pair of inputs that produce the same output (a collision).

11. Explain man in the middle attack?

If A and B exchange message, means E intercept the message and receive the B's public key and B's user id, E sends its own message with its own public key and B' s user ID based on the private key and Y. B compute the secret key and A compute k2 based on private key of A and Y.

12. Write a comparison table of different versions SHA parameters.

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	64	80	80
Security	80	128	192	256

13. Define Secure Hash Algorithm (SHA).

SHA is a cryptographic message digest algorithm similar to the MD4 family of hash functions developed by Rivest. It differs in that it adds an additional expansion operation, an extra round and the whole transformation was designed to accomodate the DSS block size for efficiency. The Secure Hash Algorithm takes a message of less than 2^{64} bits in length and produces a 160-bit message digest which is designed so that it should be computationaly expensive to find a text which matches a given hash.

14. What are the design objectives for HMAC?

- To use, without modifications, available hash functions.
- To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required.
- To preserve the original performance of the hash function without incurring a significant degradation.
- To use and handle keys in a simple way.
- To have a well understood cryptographic analysis of the strength of the authentication mechanism.

15. What are the properties of Hash function?

- For any given value h, it is computationally infeasible to find x such that H(x) = h – **one way property.**
- For any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x) – **weak collision resistance**.
- It is computationally infeasible to find any pair (x, y) such that H(x) =H(y) – **strong collision property.**

16. Differentiate Message Authentication Code and Hash function.

17.

Message Authentication Code	**Hash function**
MAC guarantees both integrity and authentication	Hash function guarantees only integrity
This applies secret key to the message in order to produce a small fixed size block known as cryptographic checksum or MAC	It accepts a variable size message M as input and produces a fixed size hash code H(M) called as message digest
Example:HMAC	Example: SHA,MD5

What are the properties a digital signature should have?

- It must verify the author and the data and time of signature.
- It must authenticate the contents at the time of signature.
- It must be verifiable by third parties to resolve disputes.

18. What requirements should a digital signature scheme should satisfy?

- The signature must be bit pattern that depends on the message being signed.

- It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature
- It must be practical to retain a copy of the digital signature in storage.

19. What is the role of compression function in hash function?

The hash algorithm involves repeated use of a compression function f, that takes two inputs and produces a n-bit output. At the start of hashing the chaining variable has an initial value that is specified as part of the algorithm. The final value of the chaining variable is the hash value usually b>n; hence the term compression.

20. What are the requirements of the hash function?

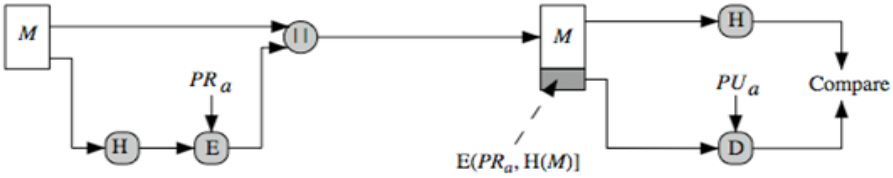
- H can be applied to a block of data of any size.
- H produces a fixed length output.
- H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.

21. Define Hash Functions.

Hash function accept a variable size message M as input and produces a fixed size hash code H(M) called as message digest as output. It is the variation on the message authentication code.A hash value h is generated by a function H of the form H = H(M)eg: SHA,MD5

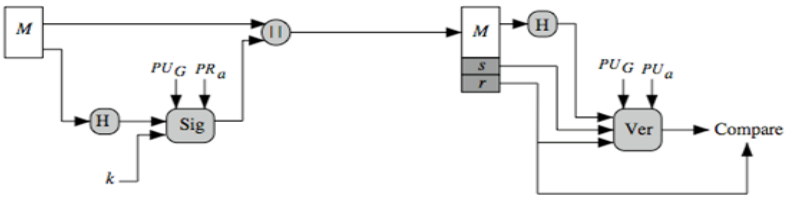
22. What are the two approaches of digital signatures?

RSA appreoach



(a) RSA Approach

DSS Approach



(b) DSS Approach

23. Define CMAC and HMAC.

In cryptography, CMAC (Cipher-based Message Authentication Code) is a block cipher-based message authentication code algorithm. CMAC is a tool for calculating message authentication codes using a block cipher coupled with a secret key. You can use an CMAC to verify both the integrity and authenticity of a message.

In cryptography, a keyed-hash message authentication code (HMAC) is a specific type of message authentication code (MAC) involving a cryptographic hash function (hence the 'H') in combination with a secret cryptographic key. HMAC is a tool for calculating message authentication codes using a cryptographic hash function coupled with a secret key. You can use an HMAC to verify both the integrity and authenticity of a message.

23. What is meant by Suppress-replay Attack?

An attack in which a message is intercepted and suppressed, then later presented to the original recipient by the attacker.

25. What basic arithmetical and logical functions are used in SHA?

Circular shifts, AND, OR, NOT and XOR

26. What two levels of functionality comprise a message authentication or digital signature mechanism?

At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower-level function is then used as primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

27. In what ways can a hash value be secured so as to provide message authentication?

Hash codes can be secured to become a MAC in various ways: HMAC, CBC - MAC and CMAC are examples.

PART-B

1. Explain the classification of authentication function in detail.

MESSAGE AUTHENTICATION FUNCTIONS

Message authentication or digital signature mechanism has two levels of functionality.

At the **lower level**, there must be some sort of function that produces an authenticator: a value to be used to authenticate a 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.

Authentication functions may be grouped into three classes as follows:

- **Hash function:** A function that maps a message of any length into a fixed length hash value, which serves as the authenticator
- **Message encryption:** The ciphertext of the entire message serves as its authenticator
- **Message authentication code (MAC):** A function of the message and a secret key that produces a fixed-length value that serves as the authenticator

Message Encryption

Message encryption by itself can provide a measure of authentication. The analysis differs for symmetric and public-key encryption schemes.

SYMMETRIC ENCRYPTION

A message M transmitted from source A to destination B is encrypted using a secret key K shared by A and B. If no other party knows the key, then confidentiality is provided: No other party can recover the plaintext of the message.Here source A and destination B knows only the key k.so they can be encrypted or decrypted only through K.hence it provides confidentiality and authentication

Given a decryption function D and a secret key , the destination will accept input x and produce output Y=D(K,X) . If X is the ciphertext of a legitimate message produced by the corresponding encryption function, then Y is some plaintext message M

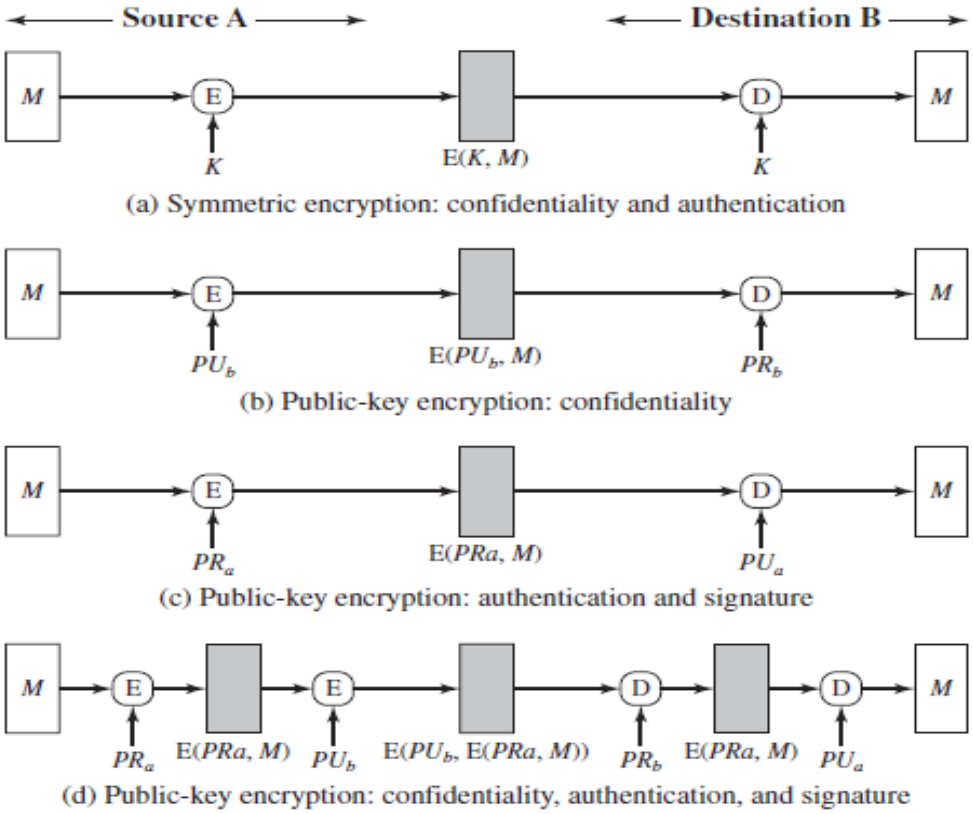


Figure 12.1 Basic Uses of Message Encryption

PUBLIC-KEY ENCRYPTION

The public-key encryption provides confidentiality but not authentication. The source (A) uses the public key ***PUb*** of the destination (B) to encrypt *M*. Because only B has the corresponding private key ***PRb***, only B can decrypt the message. This scheme provides no authentication, because any opponent could also use B's public key to encrypt a message and claim to be A.

To provide authentication,A uses its private key to encrypt the message, and B uses A's public key to decrypt .This provides authentication using the same type of reasoning as in the symmetric encryption case:The message must havecome from A because A is the only party that possesses ***PRa*** and therefore the only party with the information necessary to construct ciphertext that can be decrypted with ***PUa*** .

The scheme of Figure 12.1c does provide authentication. It also provides what is known as digital signature. Only A could have constructed the ciphertext because only A possesses ***PRa***. Not even B, therecipient, could have constructed the ciphertext. Therefore, if B is in possession of the ciphertext, B has the means to prove that the message must have come from A. In effect, A has "signed" the message by using its private key to encrypt. Note that this scheme does not provide confidentiality.

To provide both confidentiality and authentication, A can encrypt first using its private key, which provides the digital signature, and then using B's public key, which provides confidentiality (Figure 12.1d).

MESSAGE AUTHENTICATION CODE

An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a **cryptographic checksum** or MAC, that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key .When A has a message to send to B, it calculates the MAC as a function of the message and the key:

$$\mathbf{MAC = MAC(}K, M\mathbf{)}$$

where

M= input message
C = MAC function
K= shared secret key
MAC = message authentication code

The message plus MAC are transmitted to the intended recipient.The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC (Figure 12.4a). If we assume that only the receiver and the sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC. Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.

2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.

3. If the message includes a sequence number then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.

The process depicted in Figure 12.4a provides authentication but not confidentiality, because the message as a whole is transmitted in the clear. Confidentiality canbe provided by performing message encryption either after (Figure 12.4b) or before(Figure 12.4c) the MAC algorithm. In both these cases, two separate keys are needed,each of which is shared by the sender and the receiver. In the first case, the MAC iscalculated with the message as input and is then concatenated to the message. The entire block is then encrypted. In the second case, the message is encrypted first.Then the MAC is calculated using the resulting ciphertext and is concatenated to the ciphertext to form the transmitted block.Typically, it is preferable to tie the authentication directly to the plaintext, so the method of Figure 12.4b is used.

HASH FUNCTIONS

A hash function maps a variable-length message into a fixed-length hash value, or message digest. A "good" hash function has the property that the results of applying the function to a large set of inputs will produce outputs that are evenly distributed and apparently random. When a hash function is used to provide message authentication, the hash function value is often referred to as a **message digest**

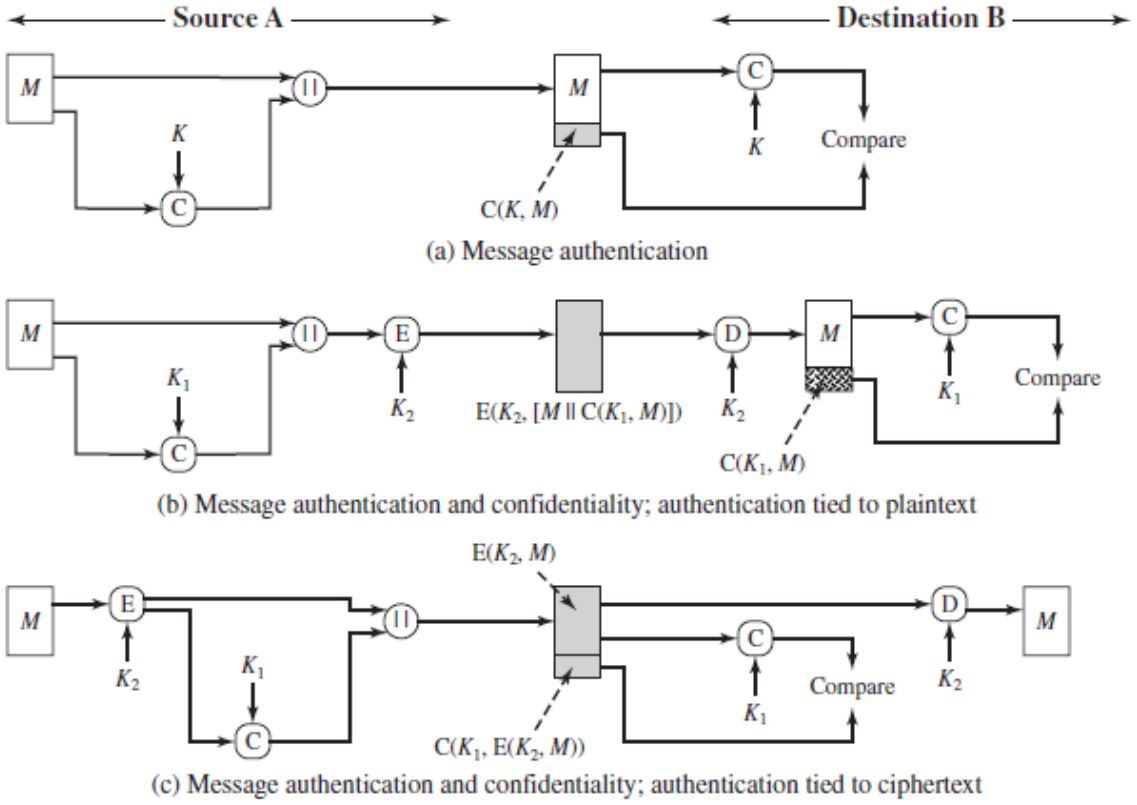


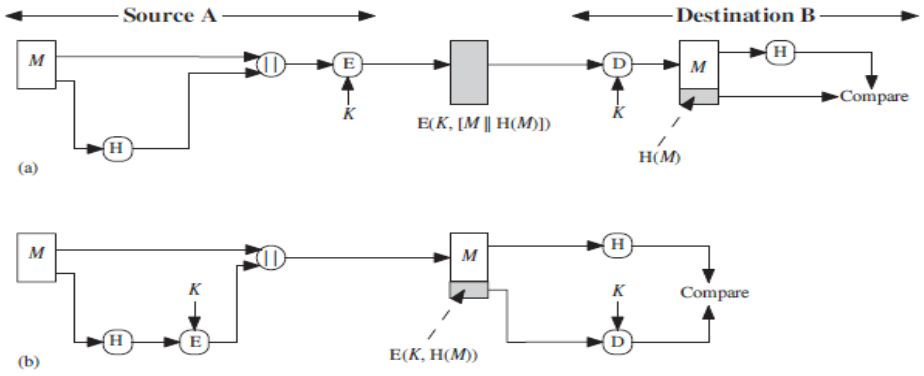
Figure 12.4 Basic Uses of Message Authentication code (MAC)

a. The message plus concatenated hash code is encrypted using symmetric encryption. Because only A and B share the secret key, the message must have come from A and has not been altered. The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

b. Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.

c. It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value *S*.A computes the hash value over the concatenation of *M*and *S* and appends the resulting hash value to . Because B possesses , it can recomputed the hash value to verify. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.

d. Confidentiality can be added to the approach of method (c) by encrypting the entire message plus the hash code



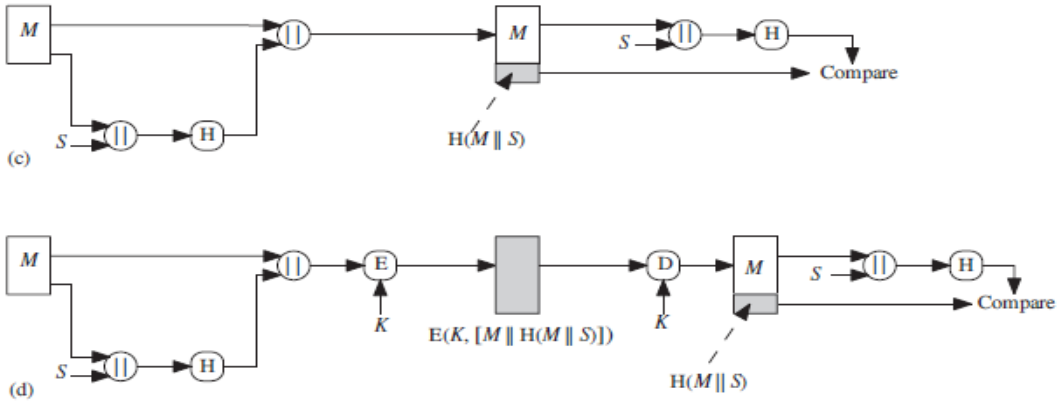


Figure 11.2 Simplified Examples of the Use of a Hash Function for Message Authentication

Digital Signatures

Another important application, which is similar to the message authenticationapplication, is the **digital signature**. Figure 11.3 illustrates, in a simplified fashion, how a hash code is used toprovide a digital signature.

a. The hash code is encrypted, using public-key encryption with the sender's privatekey. It also provides a digital signature, because only the sender could have produced the encrypted hash code. In fact, this is the essence of the digital signature technique.

b. If confidentiality as well as a digital signature is desired, then the message plus the private-key encrypted hash code can be encrypted using a symmetric secret key..

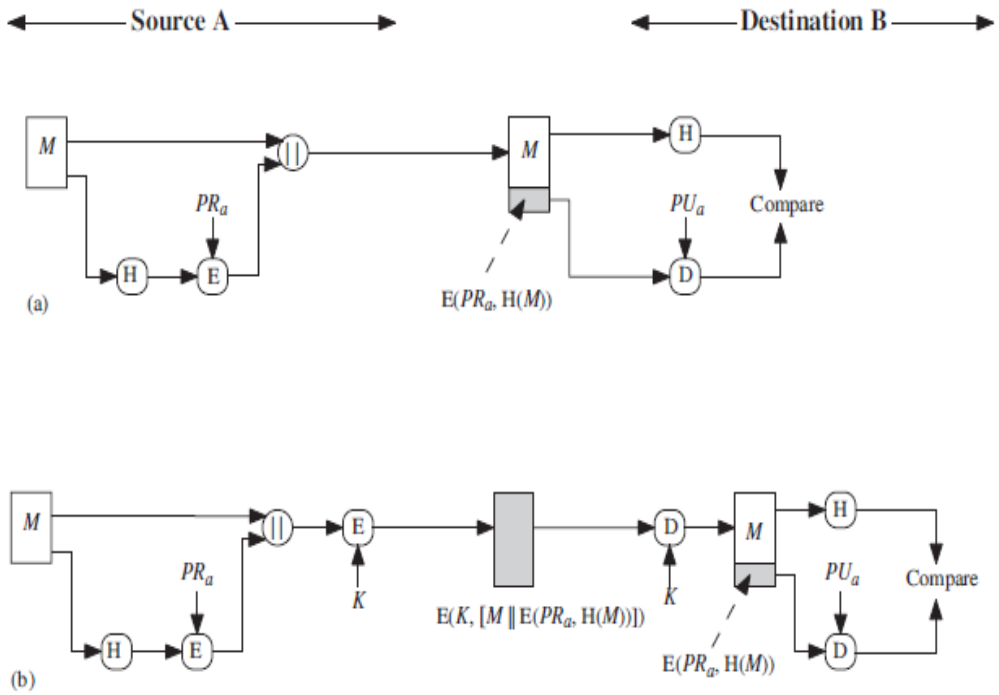


Figure 11.3 Simplified Examples of Digital Signatures

2. Explain about secure hash algorithm (SHA) in detail.

The most widely used hash function has been the Secure Hash Algorithm (SHA). SHA was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993.SHA is based on the hash function MD4, and its design closely models MD4. three new versions of SHA, with hash value lengths of 256, 384, and 512 bits, known as SHA-256, SHA-384, and SHA-512, respectively. Collectively, these hash algorithms are known as **SHA-2**.

SHA-512 Logic

The algorithm takes as input a message with a maximum length of less than bits 2^{128} bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks. The processing consists of the following steps.

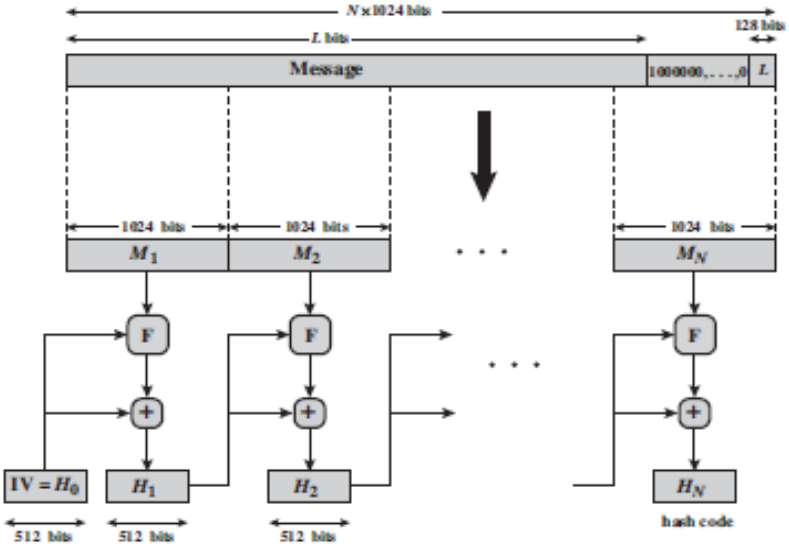
Step 1 Append padding bits. The message is padded so that its length is congruentto 896 modulo 1024 . Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 1024. The padding consists of a single 1 bit followed by the necessary number of 0 bits.

Step 2 Append length. A block of 128 bits is appended to the message.This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding). The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length. The expanded message is represented as the sequence of 1024-bit blocks $M1, M2, Á , MN$, so that the total length of the expanded message is $N * 1024$ bits

.**Step 3 Initialize hash buffer.** A 512-bit buffer is used to hold intermediate and final results of the hash function.The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h).These registers are initialized to the following 64-bit integers (hexadecimal values) These values are stored in **big-endian** format, which is the most significant byte of a word in the low-address (leftmost) byte position.These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eight prime numbers.

Step 4 Process message in 1024-bit (128-word) blocks. The heart of the algorithm is a module that consists of 80 rounds; Each round takes as input the 512-bit buffer value, abcdefgh, and updates the contents of the buffer.At input to the first round, the buffer has the value of the intermediate hash value, *Hi*-1) . Each round t makes use of a 64-bit value W , derived from the current 1024-bit block being processed M_i .These values are derived using a message schedule described subsequently. Each round also makes use of an additive constant Kt ,$0< t< 79$, where indicates one of the 80 rounds.These words represent the first 64 bits of the fractional parts of the cube roots of the first 80 prime numbers.The constants provide a "randomized" set of 64-bit patterns, which should eliminate any regularities in the input data.The output of the eightieth round is added to the input to the first round (*Hi*-1) to produce *Hi*.The addition is done independently for each of the eight words in the buffer with each of the corresponding words in (*Hi*-1), using addition modulo $.2^{64}$

Step 5 Output. After all 1024-bit blocks have been processed, the output from the N th stage is the 512-bit message digest. Thus, in the first 16 steps of processing, the value of $W_t$is equal to the corresponding word in the message block. For the remaining 64 steps, the value of W_t consists of the circular left shift by one bit of the XOR of four of the preceding values of W_t , with two of those values subjected to shift and rotate operations.



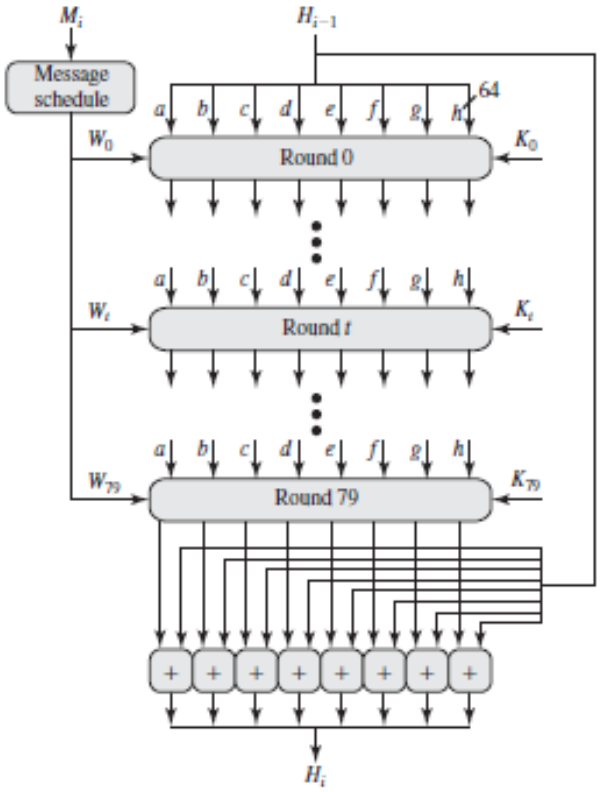


Figure 11.9 SHA-512 Processing of a Single 1024-Bit Block

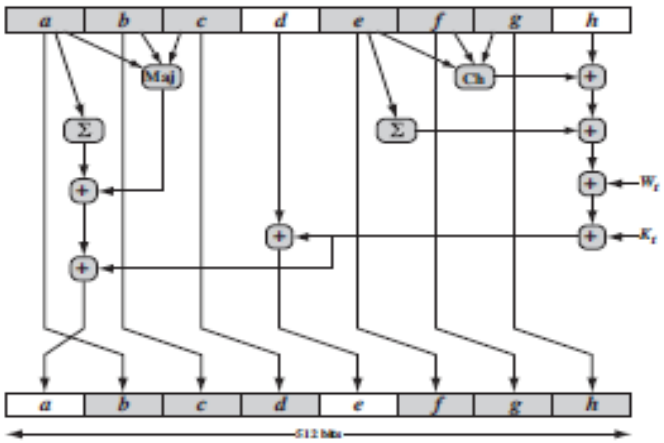
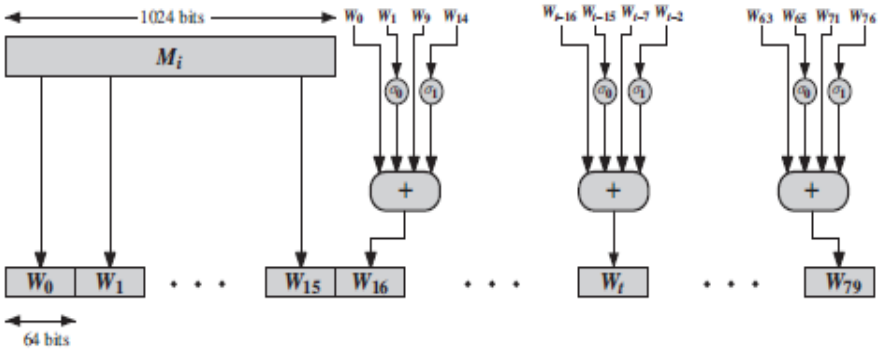


Figure 11.10 Elementary SHA-512 Operation (single round)



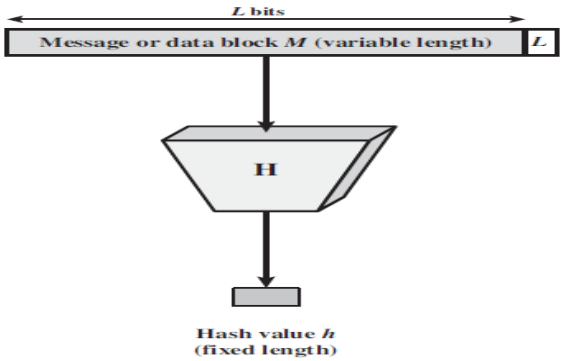
3. Describe about Hash functions.

A hash function maps a variable-length message into a fixed-length hash value, or message digest. A "good" hash function has the property that the results of applying the function to a large set of inputs will produce outputs that are evenly distributed and apparently random.

CRYPTOGRAPHIC HASH FUNCTION.

The kind of hash function needed for security applications is referred to as a **cryptographic hash function**.A cryptographic hash function is an algorithm for which it is computationally infeasible (because no attack is significantly more efficient than brute force) to find either (a) a data object that maps to a pre-specified hash result (the one-way property) or (b) two data objects that map to the same hash result (the collision-free property).

Figure 11.1 depicts the general operation of a cryptographic hash function. Typically, the input is padded out to an integer multiple of some fixed length (e.g., 1024 bits), and the padding includes the value of the length of the original message in bits.The length field is a security measure to increase the difficulty for an attacker to produce an alternative message with the same hash value



Block Diagram of Cryptographic Hash Function; $h = H(M)$

APPLICATIONS OF CRYPTOGRAPHIC HASH FUNCTIONS

Message Authentication

Message authentication is a mechanism or service used to verify the integrity of a message. When a hash function is used to provide message authentication, the hash function value is often referred to as a **message digest.**

The message plus concatenated hash code is encrypted using symmetric encryption. Because only A and B share the secret key, the message must have come from A and has not been altered. The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

b. Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.

c. It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value *S*.A computes the hash value over the concatenation of *M*and *S* and appends the resulting hash value to . Because B possesses , it can recomputed the hash value to verify. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.

d. Confidentiality can be added to the approach of method (c) by encrypting the entire message plus the hash code.

Digital Signatures

Another important application, which is similar to the message authenticationapplication, is the **digital signature**. Figure 11.3 illustrates, in a simplified fashion, how a hash code is used toprovide a digital signature.

a. The hash code is encrypted, using public-key encryption with the sender's privatekey. It also provides a digital signature, because only the sender could have produced the encrypted hash code. In fact, this is the essence of the digital signature technique.

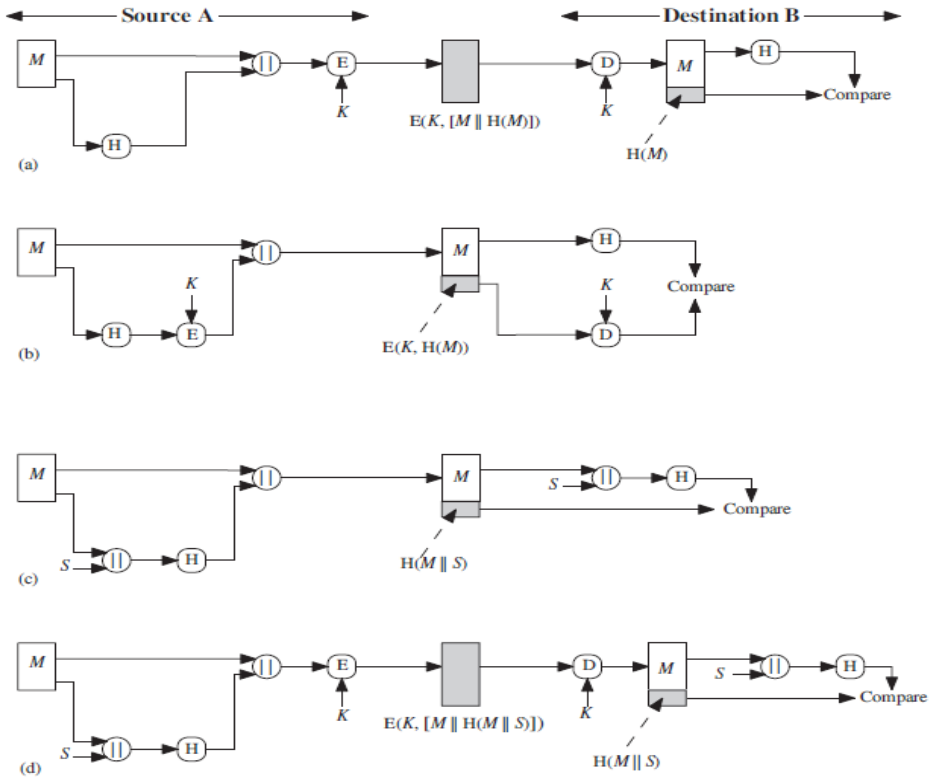


Figure 11.2 Simplified Examples of the Use of a Hash Function for Message Authentication

b. If confidentiality as well as a digital signature is desired, then the message plus the private-key encrypted hash code can be encrypted using a symmetric secret key..

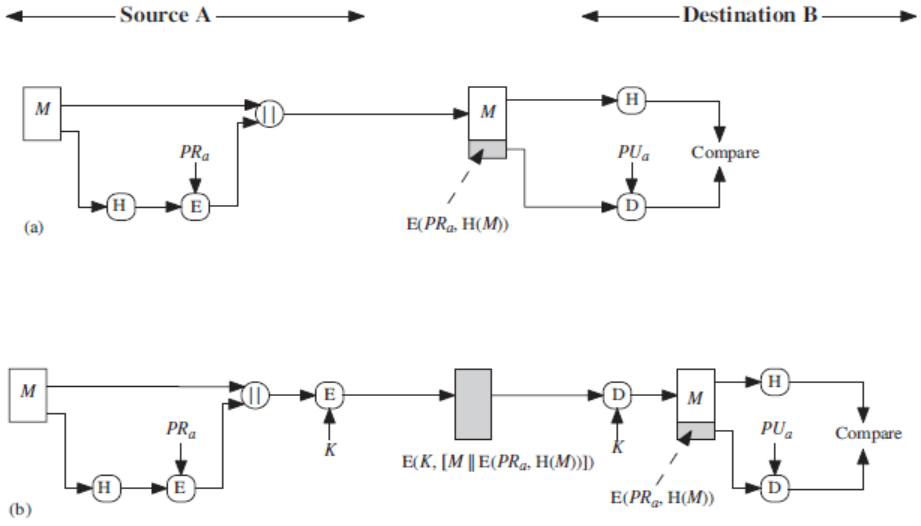


Figure 11.3 Simplified Examples of Digital Signatures

SIMPLE HASH FUNCTIONS

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

Ci = bi1 xor bi2 xor … bim

where

Ci = *i*th bit of the hash code, 1 … *i* … n

M= number of -bit blocks in the input

Bij= ith bit in jth block

1. Initially set the -bit hash value to zero.

2. Process each successive -bit block of data as follows:

a. Rotate the current hash value to the left by one bit.

b. XOR the block into the hash value.

This has the effect of "randomizing" the input more completely and overcoming any regularities that appear in the input

Given a message consisting of a sequence of 64-bit blocks X1 , X2 , … XN , define the hash code h=H(m) as the block-by-block XOR of all blocks and append the hash code as the final block:

$$h = X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N$$

Next, encrypt the entire message plus hash code using CBC mode to produce the encrypted message . Y1,Y2…..YN

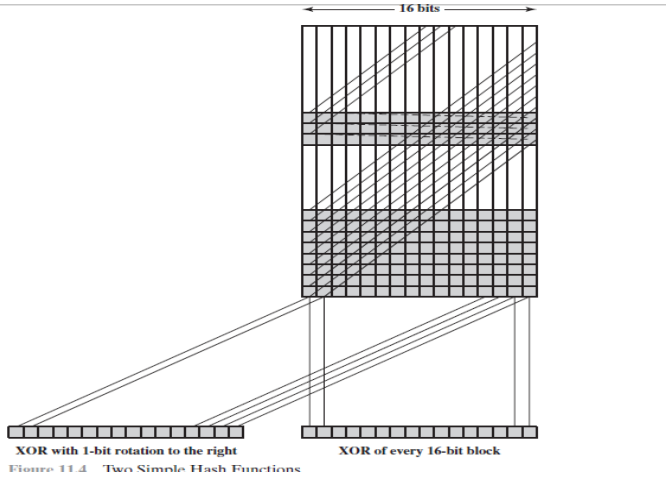


Figure 11.4 Two Simple Hash Functions

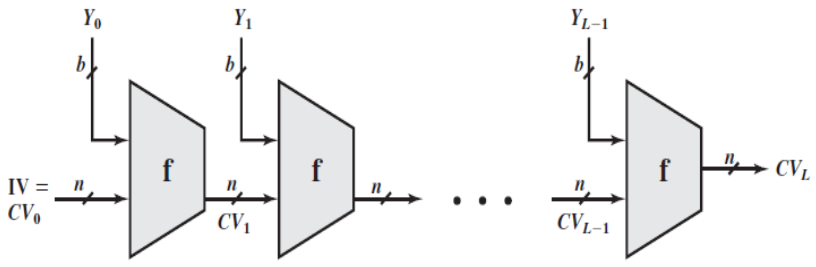
SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC HASH FUNCTIONS

The hash function takes an input message and partitions it into fixed-sized L blocks of b bits each. If necessary, the final block is padded to bits.The final block also includes the value of the total length of the input to the hash function.The inclusion of the length makes the job of the opponent more difficult. Either the opponent must find two messages of equal length that hash to the same value or two messages of differing lengths that, together with their length values, hash to the same value. The hash algorithm involves repeated use of a **compression function**, f, that takes two inputs (an -bit input from the previous step, called the *chaining variable*,and a -bit block) and produces an -bit output. At the start of hashing, the chaining variable has an initial value that is specified as part of the algorithm.The final value of the chaining variable is the hash value. Often b>n hence the term *compression*.The hash function can be summarized as

$CV0 = IV$ = initial n-bit value

$CVi = f(CVi\text{-}1, Yi\text{-}1)$ $1 < i < L$

$CVi = f(CVi\text{-}1, Yi\text{-}1)$ $1 \ldots i \ldots L$



IV = Initial value
CV_i = Chaining variable
Y_i = ith input block
f = Compression algorithm
L = Number of input blocks
n = Length of hash code
b = Length of input block

Figure 11.7 General Structure of Secure Hash Code

4. Explain about HMAC and CMAC

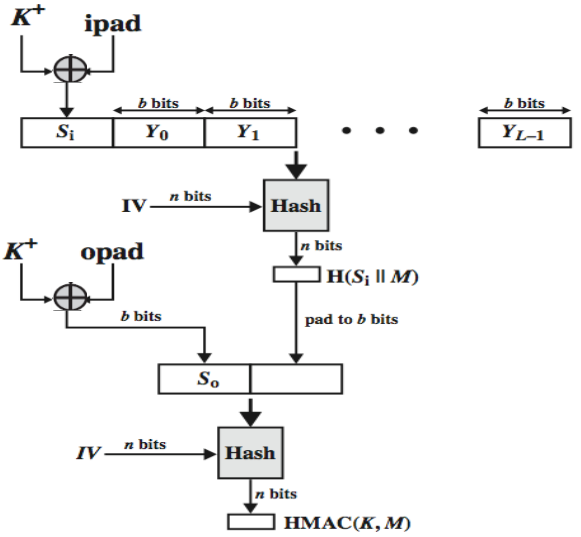
- Use available hash functions without modification.
- Preserve the original performance of the hash function without incurring a significant degradation.
- Use and handle keys in a simple way.
- Allow easy replacement of the underlying hash function in the event that faster or more secure hash functions are later available.
- Have a well-understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions on the underlying hash function.

HMACK = Hash[(K+ □ opad) || Hash[(K+ □ ipad)||M)]]

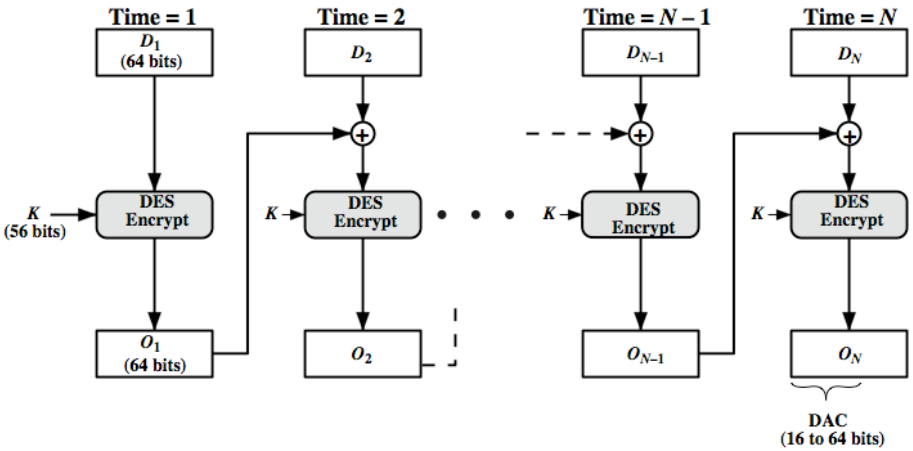
K+ is the key padded out to input block size of the hash function and opad, ipad are specified padding constants

Key size: L/2 < K < L

MAC size: at least L/2, where L is the hash output



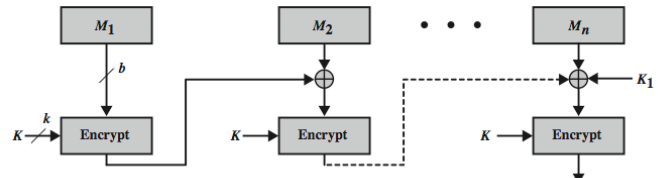
Da



CMAC

- previously saw the DAA (CBC-MAC)
- widely used in government and industry
- but has message size limitation
- can overcome using 2 keys and padding
- thus forming the Cipher-based Message Authentication Code (CMAC)
- adopted by NIST SP800-38B

OVERVIEW



To generate an ℓ-bit CMAC tag (t) of a message (m) using a b-bit block cipher (E) and a secret key (k), one first generates two b-bit sub-keys (k_1 and k_2) using the following algorithm (this is equivalent to multiplication by x and x^2 in a finite field GF(2^b)). Let $\ll$ denote the standard left-shift operator and $\oplus$ denote exclusive or:

1. Calculate a temporary value $k_0 = E_k(0)$.
2. If msb(k_0) = 0, then $k_1 = k_0 \ll 1$, else $k_1 = (k_0 \ll 1) \oplus C$; where C is a certain constant that depends only on b. (Specifically, C is the non-leading coefficients of the lexicographically first irreducible degree-b binary polynomial with the minimal number of ones.)
3. If msb(k_1) = 0, then $k_2 = k_1 \ll 1$, else $k_2 = (k_1 \ll 1) \oplus C$.
4. Return keys (k_1, k_2) for the MAC generation process.

As a small example, suppose $b = 4$, $C = 0011_2$, and $k_0 = E_k(0) = 0101_2$. Then $k_1 = 1010_2$ and $k_2 = 0100 \oplus 0011 = 0111_2$.

The CMAC tag generation process is as follows:

1. Divide message into b-bit blocks $m = m_1 \| \ldots \| m_{n-1} \| m_n$ where $m_1, \ldots, m_{n-1}$ are complete blocks. (The empty message is treated as 1 incomplete block.)
2. If m_n is a complete block then $m_n' = k_1 \oplus m_n$ else $m_n' = k_2 \oplus (m_n \| 10\ldots0_2)$.
3. Let $c_0 = 00\ldots0_2$.
4. For $i = 1, \ldots, n\text{-}1$, calculate $c_i = E_k(c_{i-1} \oplus m_i)$.
5. $c_n = E_k(c_{n-1} \oplus m_n')$
6. Output $t = \text{msb}_t(c_n)$.

5. Write notes on birthday attacks.

A **birthday attack** is a type of cryptographic attack that exploits the mathematics behind the birthday problem in probability theory. This attack can be used to abuse communication between two or more parties It states that if there are 23 people in room,theh chances are more than 50% that two of the people will share the same birthday.then the possibilities for the first person to share the birthday with anyone else among 23 will be 22.if no matches found,then for the second person the possibilities will be 21.then it goes for every person

The probability that the birthdays of any two people are not alike is clearly 364/365 (since there is only one chance in 365 that one person's birthday will coincide with another's). The probability that a third person's birthday will differ from the other two is 363/365; a fourth person's, 362/365; and so on, until we reach the 24th person (342/365).We thus obtain a series of 23 fractions which must be multiplied together to reach the probability that all 24 birthdays are different. The product is a fraction that reduces to about 0.507, or slightly better than 1/2, for a coincidence among 23 people.

To derive this answer formally, let us define

Pr(n,k)=pr[at least one duplicate in items, with each item able to take onone of equally likely values between 1 and n]

Thus, we are looking for the smallest value K of such that P(365, k)>=.5. It iseasier first to derive the probability that there are no duplicates, which we designate as Q(365, k). If k>365 , then it is impossible for all values to be different. So we assume k<=365. Now consider the number of different ways, N, that we can have k values with no duplicates.We may choose any of the 365 values for the first item, any of the remaining 364 numbers for the second item, and so on. Hence, the number of different ways is

$$N = 365 \times 364 \times \ldots (365 - k + 1) = \frac{365!}{(365 - k)!}$$

If we remove the restriction that there are no duplicates, then each item can beany of 365 values, and the total number of possibilities is 365^k. So the probability of no duplicates is simply the fraction of sets of values that have no duplicates out of all possible sets of values:

$$\mathbf{Q(365,K)= 365!(365\text{-}K)!/(365)^K=365!/\ (365\text{-}K)!(\ 365)^K}$$

$$\mathbf{P(365,K)=1-\ Q(365,K)=\ 1\text{-}365!/\ (365\text{-}K)!(\ 365)^K}$$

The probabilities may seem surprisingly large to anyone who has not considered the problem before. Many people would guess that to have a probability greater than 0.5 that there is at least one duplicate, the number of people in the group would have to be about 100. In fact, the number is 23, P(365, 23) = 0.5073 . For k = 100 , the probability of at least one duplicate is 0.9999997.

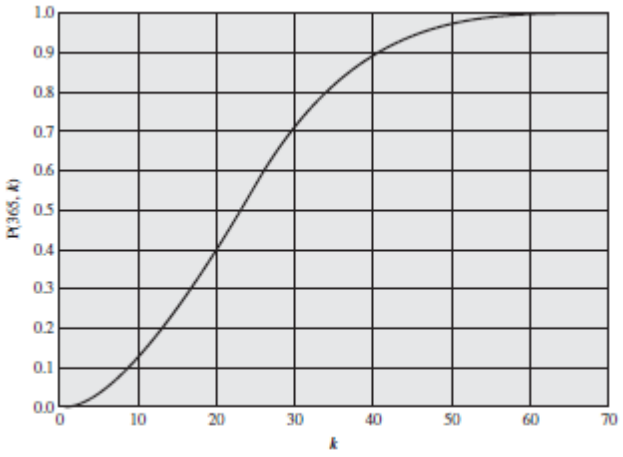


Figure 11.13 The Birthday Paradox

COLLISION

If any two persons having same birthday then it is called collision likewise.while genereatin the hash calues if any two messages produces same hash function,it is said to be collision.In order to avoid such collision problem ,a special security is introduced called as birthday attack

Suppose if we have 64 bit,then there are chances for any two hash could be same in 264 possibilities.with the 64 bit hash code the opponent can try about 264 message to find the one that matches the hash code of the intercepted messages.

Birthday Paradox Strategy as folows

1. The source, A, is prepared to sign a legitimate messageby appending the appropriate -bit hash code and encrypting that hash code with A's private key

2. The opponent generates $2^{m/2}$ variations of the messages, all of which convey essentially the same meaning, and stores the messages and their hash values.

3. The opponent prepares a fraudulent message for which A's signature is desired.

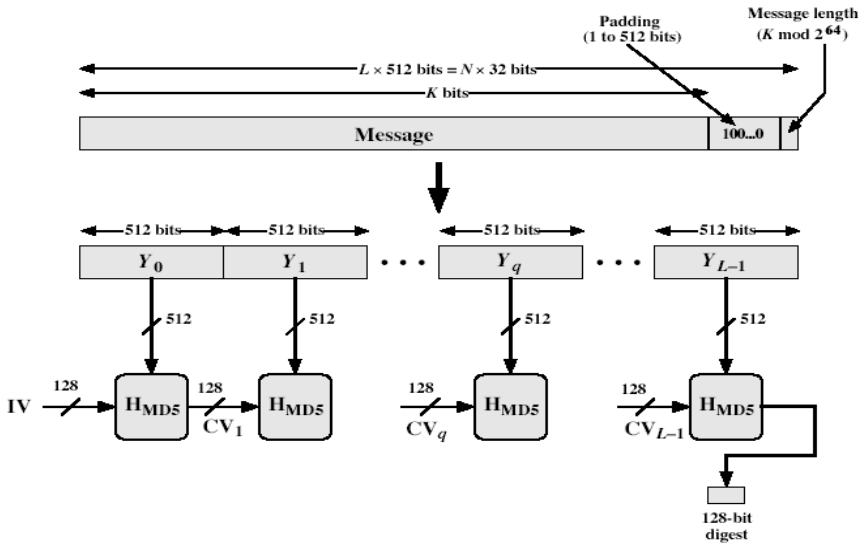
4. The opponent generates minor variations of *y*, all of which convey essentially the same meaning. For each , the opponent computes , checks for matches with any of the values, and continues until a match is found.That is, the process continues until a is generated with a hash value equal to the hash value of one of the values.

5. The opponent offers the valid variation to A for signature. This signature can then be attached to the fraudulent variation for transmission to the intended recipient. Because the two variations have the same hash code, they will produce the same signature; the opponent is assured of success even though the encryption key is not known.

6. Write algorithm of MD5 and explain.

designed by Ronald Rivest (the R in RSA)

- latest in a series of MD2, MD4
- produces a 128-bit hash value
- until recently was the most widely used hash algorithm



- in recent times have both brute-force & cryptanalytic concerns
- specified as Internet standard RFC1321

MD5 Overview

1. pad message so its length is 448 mod 512
   - Padding of 1-512 bits is always used.
   - Padding: 1000….0
2. append a 64-bit length value to message
   - Generate a message with 512L bits in length
3. initialise 4-word (128-bit) MD buffer (A,B,C,D)
4. process message in 16-word (512-bit) blocks:
5. output hash value is the final buffer value
6. append a 64-bit length value to message
   - Generate a message with 512L bits in length
7. initialise 4-word (128-bit) MD buffer (A,B,C,D)
8. process message in 16-word (512-bit) blocks:
9. output hash value is the final buffer value

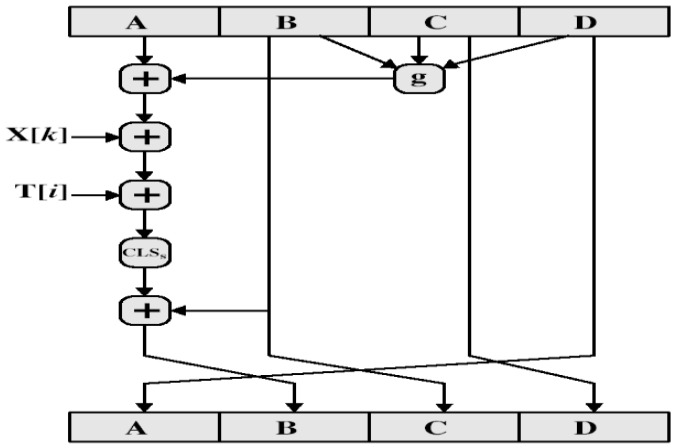
Strength of MD5

- Every hash bit is dependent on all message bits
- Rivest conjectures security is as good as possible for a 128 bit hash
- Given a hash, find a message: $O(2^{128})$ operations
- No disproof exists yet
- known attacks are:
- Berson 92 attacked any 1 round using differential cryptanalysis (but can't extend)
- Boer & Bosselaers 93 found a pseudo collision (again unable to extend)
- Dobbertin 96 created collisions on MD compression function for one block, cannot expand to many blocks

  Brute-force search now considered possible

MD5 Compression Function

- each round has 16 steps of the form:
  a = b+((a+g(b,c,d)+X[k]+T[i])<<<s)
- a,b,c,d refer to the 4 words of the buffer, but used in varying permutations
  - note this updates 1 word only of the buffer
  - after 16 steps each word is updated 4 times
- where g(b,c,d) is a different nonlinear function in each round (F,G,H,I)
- T[i] is a constant value derived from sin
- The point of all this complexity:
  - To make it difficult to generate collisions



7. Write a detailed note on Digital signatures.

DIGITAL SIGNATURE:

A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. Typically the signature is formed by taking the hash of the message and encrypting the message with the creator's private key. The signature guarantees the source and integrity of the message. The digital signature standard (DSS) is an NIST standard that uses the secure hash algorithm (SHA).

PROPERTIES:

The digital signature must have the following

• It must verify the author and the date and time of the signature.

• It must authenticate the contents at the time of the signature.

• It must be verifiable by third parties, to resolve disputes.

Figure 13.1 is a generic model of the process of making and using digital signatures. Bob can sign a message using a digital signature generation algorithm.The inputs to the algorithm are the message and Bob's private key. Any other user, say Alice, can verify the signature using a verification algorithm, whose inputs are the message, the signature, and Bob's public key

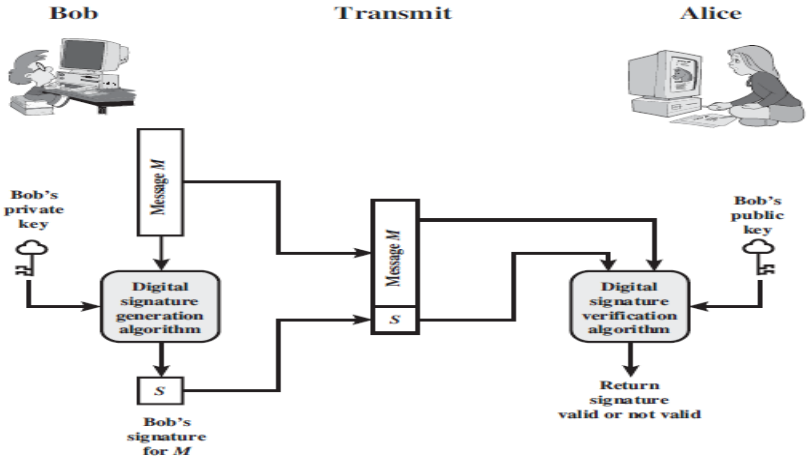


Figure 13.1 Generic Model of Digital Signature Process

DIGITAL SIGNATURE REQUIREMENTS

The following are the requirements for a digital signature.

• The signature must be a bit pattern that depends on the message being signed.

• The signature must use some information unique to the sender to prevent both forgery and denial.

• It must be relatively easy to produce the digital signature.

• It must be relatively easy to recognize and verify the digital signature.

• It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.

• It must be practical to retain a copy of the digital signature in storage.

DIRECT DIGITAL SIGNATURE

The term **direct digital signature** refers to a digital signature scheme that involves only the communicating parties (source, destination). It is assumed that the destination knows the public key of the source.

Confidentiality can be provided by encrypting the entire message plus signature with a shared secret key (symmetric encryption). Note that it is important to perform the signature function first and then an outer confidentiality function. In case of dispute, some third party must view the message and its signature. If the signature is calculated on an encrypted message, then the third party also needs access to the decryption key to read the original message. However, if the signature is the inner operation, then the recipient can store the plaintext message and its signature

for later use in dispute resolution.

Another threat is that some private key might actually be stolen from X at time T. The opponent can then send a message signed with X's signature and stamped with a time before or equal to T.

Arbitrated Digital Signatures

- involves use of arbiter A
  - validates any signed message
  - then dated and sent to recipient
- requires suitable level of trust in arbiter
- can be implemented with either private or public-key algorithms
- arbiter may or may not see message

Authentication Protocols

- used to convince parties of each others identity and to exchange session keys
- may be one-way or mutual

- key issues are
  - confidentiality – to protect session keys
  - timeliness – to prevent replay attacks
- published protocols are often found to have flaws and need to be modified

8. Briefly explain Digital Signature Algorithm

1. Global Public key Components

p - prime no. where $2^{L-1} < p < 2^{L}$ for $512 <= L <= 1024$ and L a multiple of 64

q – prime divisor of (p-1) where $2^{159} < p < 2^{160}$

ie., bit length of 160 bits

$g = h^{(p-1)/q} \bmod p$ where h is any integer with $1 < h < (p-1)$ such that $h^{(p-1)/q} \bmod p > 1$

2. User's Private key

x - random or pseudo random integer with $0 < x < q$

3.User's Public key

$y = g^{x} \bmod p$

4.User's Per Message Secret Number

k = random or pseudo random integer with $0 < k < q$

5.DSA Signature Creation

- to sign a message M the sender:
  - generates a random signature key k, k<q
  - nb. k must be random, be destroyed after use, and never be reused
- then computes signature pair:

  $r = (g^{k} \bmod p) \bmod q$

  $s = [k^{-1}(H(M) + xr)] \bmod q$
- sends **signature (r,s)** with **message M**

6.DSA Signature Verification

- having received M & signature (r,s)
- to **verify** a signature, recipient computes:

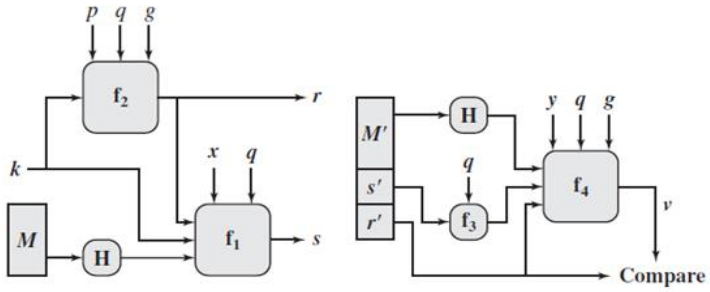
  $w = s^{-1} \bmod q$

  $u1 = [H(M)w] \bmod q$

  $u2 = (rw) \bmod q$

  $v = [(g^{u1} y^{u2}) \bmod p] \bmod q$

  if v=r then signature is verified



(a) Signing (b) Verifying

9. Explain Digital Signature Standards.

DIGITAL SIGNATURE ALGORITHM (DSA)

The DSS makes use of the Secure Hash Algorithm (SHA) and presents a new digital signature technique, the the **Digital Signature Algorithm (DSA)**.

- creates a 320 bit signature
- with 512-1024 bit security
- smaller and faster than RSA
- a digital signature scheme only
- security depends on difficulty of computing discrete logarithms

variant of ElGamal & Schnorr schemes

RSA APPROACH

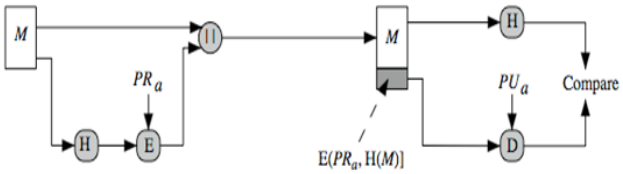
In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key to form the signature. Both the message and the signature are then transmitted. The recipient takes the message and produces a hash code. The recipient also decrypts the signature using the sender's public key. If the calculated hash code matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.

M – Message H(M) – Hash function of M C – Cipher text

C = $M \| E_{PRa}[H(M)]$

v1 = $D_{PUa}[H(M)]$

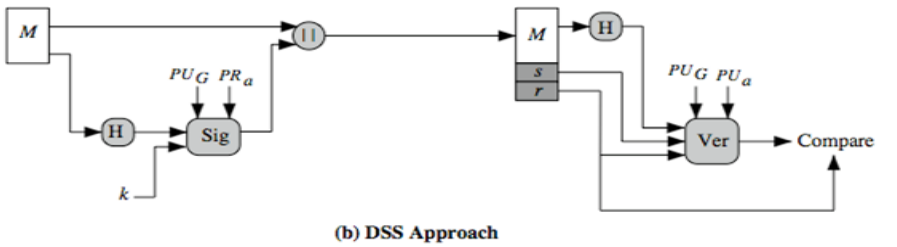
v2 = H(M) if v1= v2 Then signature is verified



(a) RSA Approach

DSS APPROACH

The DSS approach also makes use of a hash function. The hash code is provided as input to a signature function along with a random number k rgenerated for this particular signature.The signature function also depends on the sender's private key (PR_a) and a set of parameters known to a group of communicating principals. We can consider this set to constitute a global public key(PU_G) The result is a signature consisting of two components, labeled *s* and *r*. At the receiving end, the hash code of the incoming message is generated.This plus the signature is input to a verification function. The verification function also depends on the global public key as well as the sender's public key (PU_a), which is paired with the sender's private key. The output of the verification function is a value that is equal to the signature component if the signature is valid. The signaturefunction is such that only the sender, with knowledge of the private key, could have produced the valid signature.



(b) DSS Approach

THE DIGITAL SIGNATURE ALGORITHM

1. Global Public key Components

p - prime no. where $2^{L-1} < p < 2^L$ for 512 <= L <=1024 and L a multiple of 64

q – prime divisor of (p-1) where $2^{159} < p < 2^{160}$

ie., bit length of 160 bits

$g = h^{(p-1)/q} \bmod p$ where h is any integer with $1 < h < (p-1)$ such that $h^{(p-1)/q} \bmod p > 1$

2. User's Private key

x - random or pseudo random integer with $0 < x < q$

3.User's Public key

$y = g^x \bmod p$

4.User's Per Message Secret Number

k = random or pseudo random integer with $0 < k < q$

5.DSA Signature Creation

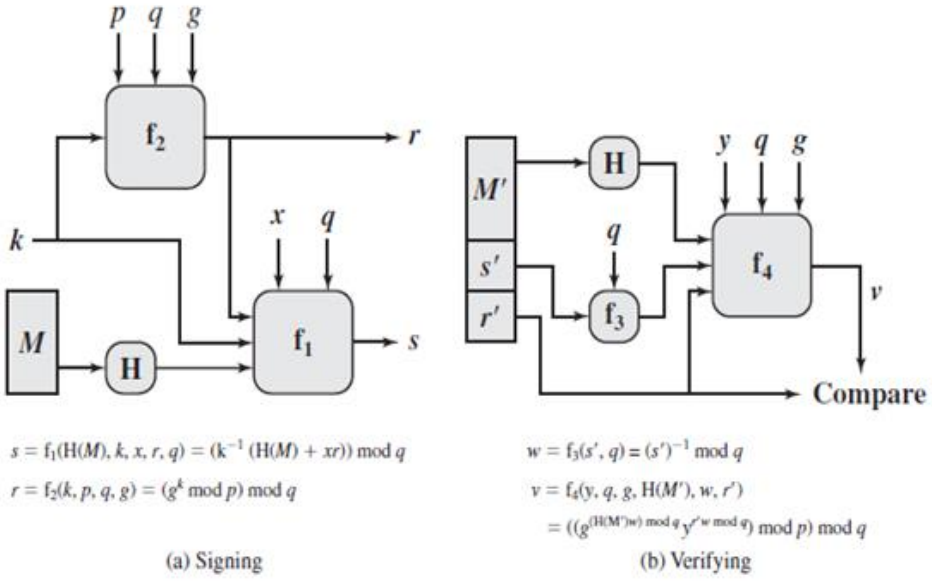
- to sign a message M the sender:
  - generates a random signature key k, k<q

- nb. k must be random, be destroyed after use, and never be reused

➢ then computes signature pair:
$r = (g^k \bmod p) \bmod q$
$s = [k^{-1}(H(M) + xr)] \bmod q$
➢ sends **signature (r,s)** with **message M**

6.DSA Signature Verification

➢ having received M & signature (r,s)
➢ to **verify** a signature, recipient computes:
$w = s^{-1} \bmod q$
$u1 = [H(M)w] \bmod q$
$u2 = (rw) \bmod q$
$v = [(g^{u1} y^{u2}) \bmod p] \bmod q$
if v=r then signature is verified



10. Explain Elgamal and Schnorr digital signatureschemes.

El Gamal Signature Scheme

whilst the ElGamal encryption algorithm is not commutative, a closely related signature scheme exists El Gamal Signature scheme given prime p, public random number g, private (key) random number x, compute

$y = g^x \pmod p$
public key is (y,g,p)
nb (g,p) may be shared by many users p must be large enough so discrete log is hard private key is (x) to **sign** a message M choose a random number k, GCD(k,p-1)=1

compute $a = g^k \pmod p$ use extended Euclidean (inverse) algorithm to solve $M = x.a + k.b \pmod{p-1}$ the signature is (a,b), k must be kept secret (like ElGamal encryption is double the message size) to **verify** a signature (a,b) confirm:

$y^a.a^b \pmod p = g^M \pmod p$

Example of ElGamal Signature Scheme

given p=11, g=2
choose private key x=8

compute $y = g^x \pmod p = 2^8 \pmod{11} = 3$
public key is y=3,g=2,p=11)
to sign a message M=5
choose random k=9
confirm gcd(10,9)=1

compute $a = g^k \pmod p = 2^9 \pmod{11} = 6$

solve

$$M = x.a+k.b(\text{mod } p-1)$$

$$5 = 8.6+9.b(\text{mod } 10)$$

giving $b = 3$

signature is (a=6,b=3) to verify the signature, confirm the following are correct:

$y^{a}.a^{b}(\text{mod } p) = g^{M}(\text{mod } p)$

$3^{6}.6^{3}(\text{mod } 11) = 2^{5}(\text{mod } 11)$

DSA (Digital Signature Algorithm)

DSA was designed by NIST & NSA and is the US federal standard signature scheme (used with SHA hash alg) DSA is the algorithm, DSS is the standard There was considerable reaction to its announcement! debate over whether RSA should have been used debate over the provision of a signature only alg DSA is a variant on the ElGamal and Schnorr algorithms description of DSA

$p = 2^{L}$ a prime number, where L= 512 to 1024 bits and is a multiple of 64

q a 160 bit prime factor of p-1

$g = h^{(p-1)/q}$ where h is any number less than p-1 with $h^{(p-1)/q}(\text{mod } p)> 1$

x a number less than q

$y = g^{x}(\text{mod } p)$ to **sign** a message M

generate random k, k<q

compute

$r = (g^{k}(\text{mod } p))(\text{mod } q)$

$s = k^{-1}.SHA(M)+ x.r \text{ (mod } q)$ the signature is (r,s) to **verify** a signature:

$w = s^{-1}(\text{mod } q)$

$u1= (SHA(M).w)(\text{mod } q)$

$u2= r.w(\text{mod } q)$

$v = (gu1.yu2(\text{mod } p))(\text{mod } q)$

if v=r then the signature is verified comments on DSA was originally a suggestion to use a common modulus, this would make a tempting target, discouraged it is possible to do both ElGamal and RSA encryption using DSA routines, this was probably not intended. DSA is patented with royalty free use, but this patent has been contested, situation unclear Gus Simmons has found a subliminal channel in DSA, could be used to leak the private key from a library - make sure you trust your library implementer

UNIT 4
PART-A

1. Define Kerberos

Kerberos is an authentication service developed as part of project Athena at MIT. It was designed to allow the workstations to allow network resources in a secure manner.

2. Why Kerberos is needed.

In an open distributed environment, users at work stations wish to access services on servers distributed throughout the network. So, servers must be able to restrict access to authorized users and must be able to authenticate requests for service. In this environment, a workstation cannot be trusted to identify its users correctly to network services.

3. Give requirements of Kerberos

- Secure: Should be strong enough that a potential opponent does not find it to be the weak link.
- Reliable: Should be highly reliable and should employ a distributed server architecture, with one system able to back up another.
- Transparent: User should not be aware that authentication is taking place, beyond the requirement to enter a password.
- Scalable: The system should be capable of supporting large numbers of clients and servers.

4. Give the simple authentication dialogue

- C -> AS: IDc || Pc || IDv
- AS ->C: Ticket
- C ->V: IDc || Ticket [Ticket= Ekv[IDc|| ADc|| IDv]]

5. Give disadvantages of simple Authentication Dialogue of Kerberos version4

- More number of times that a user has enter the password
- Plain text transmission of password

6. Define TGS

TGS issues tickets to users who have been authenticated to the authentication server. This ticket is used by the client to request the server.

7. Give Advantage of V-5 over Version 4.

- Environmental shortcomings removal
- Technical deficiency overcome in V-5

8. Give Reasons for environmental shortcomings in V-4.

- Encryption system dependence
- Internet Protocol dependence
- Message Byte ordering
- Ticket lifetime
- Authentication forwarding
- Interrealm authentication

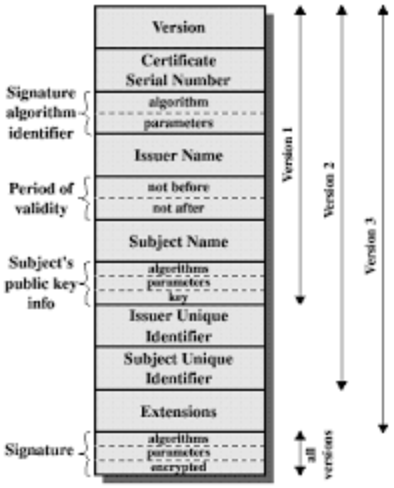
9. List of technical deficiencies in the version 4.

- Double encryption
- PCBC encryption
- Password attacks
- Session keys

10. What are the reason to revoke a certificate?

- The user's private key is assumed to be compromised.
- The user is no longer certified by this CA.
- The CA's certificate is assumed to be compromised.

11. Draw a general format of x.509 certificate.



12. List out the three classes of intruders.

- Masquerader: An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account
- Misfeasor: A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges
- Clandestine user: An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection

13. Define Intruder?

An Intruder is a person who attempts to gain unauthorized access to a system, to damage that system, or to disturb data on that system. In summary, this person attempts to violate Security by interfering with system Availability, data Integrity or data Confidentiality.

14. Define IDS

An intrusion detection system (IDS) is a type of security software designed to automatically alert administrators when someone or something is trying to compromise information system through malicious activities or through security policy violations.

15. What are two common techniques used to protect a password file?

- One-way function: The system stores only the value of a function based on the user's password. When the user presents a password, the system transforms that password and compares it with the stored value.
- Access control: Access to the password file is limited to one or a very few accounts.

16. What are three benefits that can be provided by an intrusion detection system?

- If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised. Even if the detection is not sufficiently timely to preempt the intruder, the sooner that the intrusion is detected, the less the amount of damage and the more quickly that recovery can be achieved.
- An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.
- Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility

17. What is a honeypot?

Honeypots are decoy systems that are designed to lure a potential attacker away from critical systems. Honeypots are designed to

- Divert an attacker from accessing critical systems
- Collect information about the attacker's activity
- Encourage the attacker to stay on the system long enough for administrators to respond

18. What is the role of compression in the operation of a virus?

A virus may use compression so that the infected program is exactly the same length as an uninfected version.

19. What is the role of encryption in the operation of a virus?

The mutation engine creates a random encryption key to encrypt the remainder of the virus. The key is stored with the virus, and the mutation engine itself is altered. When the virus replicates, a different random key is selected. Because the bulk of the virus is encrypted with a different key for each instance, there is no consistent bit pattern to observe.

20. What are typical phases of operation of a virus or worm?

- Dormant phase: The virus is idle, it will eventually be activated by some event, such as a date or another program.
- Propagation phase: The virus places a copy of itself into other programs or into certain system areas on the disk.
- Triggering phase: The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events.
- Execution phase: The function is performed. The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files.

21. What is a digital immune system?

This system provides a general-purpose emulation and virus-detection system. The objective is to provide rapid response time so that viruses can be stamped out almost as soon as they are introduced. When a new virus enters an organization, the immune system automatically captures it, analyzes it, adds detection and shielding for it, removes it, and passes information about that virus to systems running a general antivirus program so that it can be detected before it is allowed to run elsewhere.

22. List three design goals for a firewall.

- All traffic from inside to outside, and vice versa, must pass through the firewall. This is achieved by physically blocking all access to the local network except via the firewall.
- Only authorized traffic, as defined by the local security policy, will be allowed to pass.
- The firewall itself is immune to penetration. This implies the use of a hardened system with a secured operating system

23. List four techniques used by firewalls to control access and enforce a security policy.

- Service control: Determines the types of Internet services that can be accessed, inbound or outbound. The firewall may filter traffic on the basis of IP address, protocol, or port number.
- Direction control: Determines the direction in which particular service requests are allowed to flow.
- User control: Controls access to a service according to which user is attempting to access it.
- Behavior control: Controls how particular services are used. Eg, may filter e-mail to eliminate spam.

24. In the context of access control, what is the difference between a subject and an object?

A subject is an entity capable of accessing objects (eg. user, application, process). An object is resource to which access is controlled. An object is an entity used to contain information (eg. records, files, directories, processors, communication ports)

25. What is an access right?

An access right describes the way in which a subject may access an object. Eg. read, write, execute, delete.

26. What are three common types of firewalls?

Packet filters, application-level gateways, and circuit-level gateways

27. What is mean by SET? What are the features of SET?

Secure Electronic Transaction (SET) is an open encryption and security specification designed to protect credit card transaction on the internet.

Features are:

- Confidentiality of information
- Integrity of data
- Cardholder account authentication
- Merchant authentication

28. In the content of Kerberos, what is realm?

A full service Kerberos environment consisting of a Kerberos server, a no. of clients, no. of application server requires the following: The Kerberos server must have user ID and hashed password of all participating users in its database. The Kerberos server must share a secret key with each server. Such an environment is referred to as "Realm".

29. What are the different approaches to intrusion detection?

- Statistical anomaly detection: Involves the collection of data relating to the behavior of legitimate users over a period of time.
    - Threshold detection: This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.
    - Profile based: A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.
- Rule-based detection: Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.
    - Anomaly detection: Rules are developed to detect deviation from previous usage patterns.
    - Penetration identification: An expert system approach that searches for suspicious behavior.

PART B

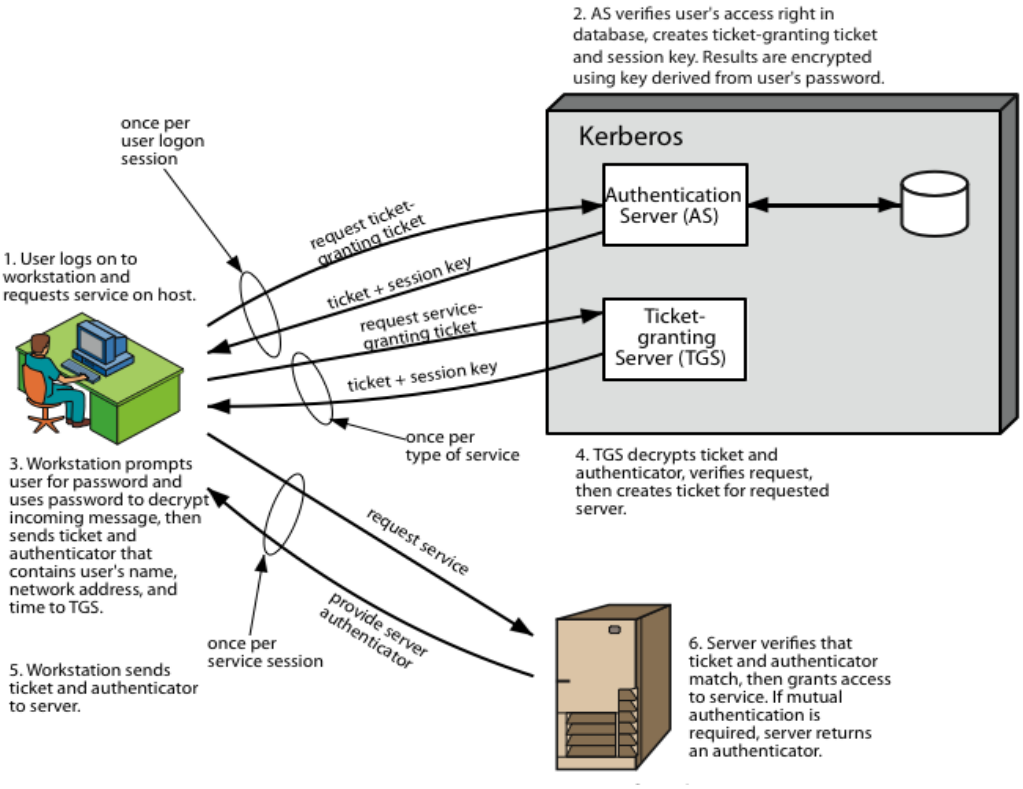
1. Explain kerberos authentication mechanism with suitable diagram?

Kerberos Requirements

- secure
- reliable
- transparent
- scalable

Kerberos v4 Overview

- a basic third-party authentication scheme
- have an Authentication Server (AS)
    - users initially negotiate with AS to identify self
    - AS provides a non-corruptible authentication credential (ticket granting ticket TGT)
- have a Ticket Granting server (TGS)
    - users subsequently request access to other services from TGS on basis of users TGT



Simple Authentication Dialogue

1. C →AS :IDc || Pc || IDv
2. AS → C : Ticket
3. C → V : IDc || Ticket

Ticket = Ekv [IDc || ADc || IDv]

Where C Client V Server
AS Authentication Server
IDc Identifier of user on C
IDv identifier of V
Pc Password of user on C
ADc network address of C
Kv Key shared b/w AS and V

Secure Authentication Dialogue

Once per user logon session

1. C →AS :IDc || IDtgs
2. AS → C : E Kc [$Ticket_{tgs}$]

Once per type of Service

3. C →TGS :IDc|| IDv || $Ticket_{tg}$
4. TGS → C : $Ticket_{v}$

Once per Service Session

5. C → V : IDc || $Ticket_{v}$
6. $Ticket_{tgs}$ = E Ktgs [IDc || ADc || IDtgs || TS1 || LifeTime 1]

$Ticket_{v}$ = Ekv [IDc || ADc || IDv || TS2 || LifeTime 2]

(a) Authentication Service Exchange: to obtain ticket-granting ticket
(1) C → AS: $ID_c \| ID_{tgs} \| TS_1$
(2) AS → C: $E_{K_c}\left[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}\right]$
$Ticket_{tgs} = E_{K_{tgs}}\left[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2\right]$
(b) Ticket-Granting Service Exchange: to obtain service-granting ticket
(3) C → TGS: $ID_v \| Ticket_{tgs} \| Authenticator_c$
(4) TGS → C: $E_{K_{c,tgs}}\left[K_{c,v} \| ID_v \| TS_4 \| Ticket_v\right]$
$Ticket_{tgs} = E_{K_{tgs}}\left[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2\right]$
$Ticket_v = E_{K_v}\left[K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4\right]$
$Authenticator_c = E_{K_{tgs}}\left[ID_C \| AD_C \| TS_3\right]$
(c) Client/Server Authentication Exchange: to obtain service
(5) C → V: $Ticket_v \| Authenticator_c$
(6) V → C: $E_{K_{c,v}}\left[TS_5 + 1\right]$ (for mutual authentication)
$Ticket_v = E_{K_v}\left[K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4\right]$
$Authenticator_c = E_{K_{c,v}}\left[ID_C \| AD_C \| TS_5\right]$

A Kerberos realm is a set of managed nodes that share the same Kerberos database. The Kerberos database resides on the Kerberos master computer system, which should be kept in a physically secure room. A read-only copy of the Kerberos database might also reside on other Kerberos computer systems. However, all changes to the database must be made on the master computer system. Changing or accessing the contents of a Kerberos database requires the Kerberos master password.

A related concept is that of a Kerberos principal, which is a service or user that is known to the Kerberos system. Each Kerberos principal is identified by its principal name. Principal names consist of three parts: a service or user name, an instance name, and a realm name

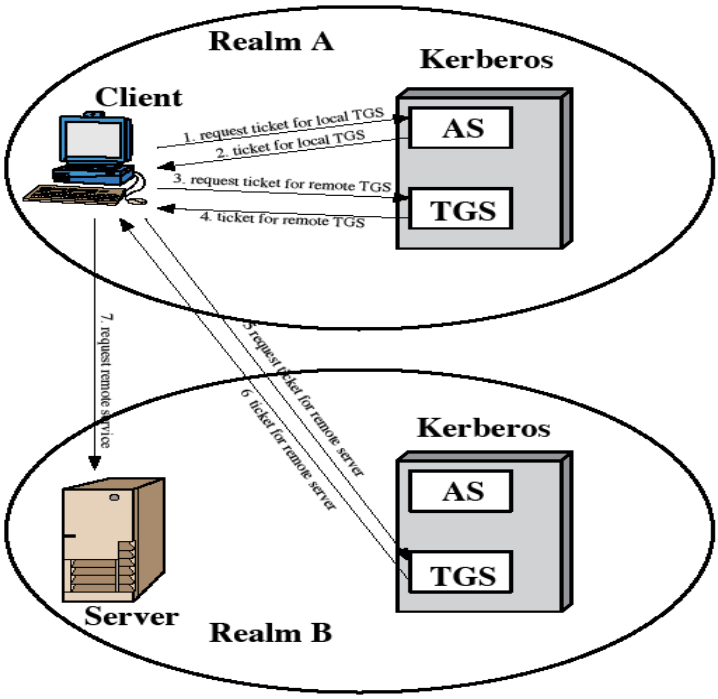
Networks of clients and servers under different administrative organizations typically constitute different realms. That is, it generally is not practical, or does not conform to administrative policy, to have users and servers in one administrative domain registered with a Kerberos server elsewhere.

However, users in one realm may need access to servers in other realms, and some servers may be willing to provide service to users from other realms, provided that those users are authenticated.

Kerberos version5

Version 5 of Kerberos provides a number of improvements over version 4.

- developed in mid 1990's
- provides improvements over v4
  - addresses environmental shortcomings
  - and technical deficiencies
- specified as Internet standard RFC 1510



2. What is the differences between kerberos versions 4 and 5 and explain version 5 authentication Dialogue?

Version 5 is intended to address the limitations of version 4 in two areas: environmental shortcomings and technical deficiencies.

Environmental shortcomings

1. **Encryption system dependence:** Version 4 requires the use of DES. Export restriction on DES as well as doubts about the strength of DES were thus of concern. In version 5, ciphertext is tagged with an encryption type identifier so that any encryption technique may be used. Encryption keys are tagged with a type and a length, allowing the same key to be used in different algorithms and allowing the specification of different variations on a given algorithm.
2. **Internet protocol dependence:** Version 4 requires the use of Internet Protocol (IP) addresses. Other address types, such as the ISO network address, are not accommodated. Version 5 network addresses are tagged with type and length, allowing any network address type to be used.
3. **Message byte ordering:** In version 4, the sender of a message employs a byte ordering of its own choosing and tags the message to indicate least significant byte in lowest address or most significant byte in lowest address. This techniques works but does not follow established conventions. In version 5, all message structures are defined using Abstract Syntax Notation One (ASN.1) and Basic Encoding Rules (BER), which provide an unambiguous byte ordering.
4. **Ticket lifetime:** Lifetime values in version 4 are encoded in an 8-bit quantity in units of five minutes. Thus, the maximum lifetime that can be expressed is 2^8 x 5 = 1280 minutes, or a little over 21 hours. This may be inadequate for some applications (e.g., a long-running simulation that requires valid Kerberos credentials throughout execution). In version 5, tickets include an explicit start time and end time, allowing tickets with arbitrary lifetimes.
5. **Authentication forwarding:** Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client. This capability would enable a client to access a server and have that server access another server on behalf of

the client. For example, a client issues a request to a print server that then accesses the client's file from a file server, using the client's credentials for access. Version 5 provides this capability.

6. **Interrealm authentication:** In version 4, interoperability among N realms requires on the order of N^2 Kerberos-to-Kerberos relationships, as described earlier. Version 5 supports a method that requires fewer relationships, as described shortly.

Technical deficiencies

1. **Double encryption:** [messages (2) and (4)] that tickets provided to clients are encrypted twice, once with the secret key of the target server and then again with a secret key known to the client. The second encryption is not necessary and is computationally wasteful.
2. **PCBC encryption:** Encryption in version 4 makes use of a nonstandard mode of DES known as propagating cipher block chaining (PCBC). It has been demonstrated that this mode is vulnerable to an attack involving the interchange of cipher text blocks. PCBC was intended to provide an integrity check as part of the encryption operation. Version 5 provides explicit integrity mechanisms, allowing the standard CBC mode to be used for encryption. In particular, a checksum or hash code is attached to the message prior to encryption using CBC.
3. **Session keys:** Each ticket includes a session key that is used by the client to encrypt the authenticator sent to the service associated with that ticket. In addition, the session key may subsequently be used by the client and the server to protect messages passed during that session. However, because the same ticket may be used repeatedly to gain service from a particular server, there is the risk that an opponent will replay messages from an old session to the client or the server. In version 5, it is possible for a client and server to negotiate a subsession key, which is to be used only for that one connection. A new access by the client would result in the use of a new subsession key.
4. **Password attacks:** Both versions are vulnerable to a password attack. The message from the AS to the client includes material encrypted with a key based on the client's password.[8] An opponent can capture this message and attempt to decrypt it by trying various passwords. If the result of a test decryption is of the proper form, then the opponent has discovered the client's password and may subsequently use it to gain authentication credentials from Kerberos. With the same kinds of countermeasures being applicable. Version 5 does provide a mechanism known as preauthentication, which should make password attacks more difficult, but it does not prevent them.

The Version 5 Authentication Dialogue

(1) C →AS $Options||ID_c||Realm_c||ID_{tgs}||Times||Nonce_1$

(2) AS →C $Realm_c||ID_C||Ticket_{tgs}||E(K_c, [K_{c,tgs}||Times||Nonce_1||Realm_{tgs}||ID_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [Flags||K_{c,tgs}||Realm_c||ID_c||AD_c||Times])$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C →TGS $Options||ID_v||Times||||Nonce_2||Ticket_{tgs}||Authenticator_c$

(4) TGS →C $Realm_c||ID_c||Ticket_v||E(K_{c,tgs}, [K_{c,v}||Times||Nonce_2||Realm_v||ID_v])$
$Ticket_{tgs} = E(K_{tgs}, [Flags||K_{C,tgs}||Realm_c||ID_C||AD_C||Times])$

$Ticket_v = E(K_v, [Flags||K_{c,v}||Realm_c||ID_C||AD_c||Times])$
$Authenticator_c = E(K_{c,tgs}, [ID_C||Realm_c||TS_1])$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) C →V $Options||Ticket_v||Authenticator_c$

(6) V →C $E_{Kc,v}[TS_2||Subkey||Seq\#]$

$Ticket_v = E(K_v, [Flags||K_{c,v}||Realm_c||ID_C||AD_C||Times])$

$$Authenticator_c = E(K_{c,v},[ID_C||Realm_c||TS_2||Subkey||Seq\#])$$

(c) Client/Server Authentication Exchange to obtain service

First, consider the **authentication service exchange**. Message (1) is a client request for a ticket-granting ticket. As before, it includes the ID of the user and theTGS. The following new elements are added:

• **Realm:** Indicates realm of user
• **Options:** Used to request that certain flags be set in the returned ticket
• **Times:** Used by the client to request the following time settings in the ticket:
—**from**: the desired start time for the requested ticket
—**till**: the requested expiration time for the requested ticket
—**rtime**: requested renew-till time
• **Nonce:** A random value to be repeated in message (2) to assure that the response is fresh and has not been replayed by an opponent

Message (2) returns a ticket-granting ticket, identifying information for the client, and a block encrypted using the encryption key based on the user's password. This block includes the session key to be used between the client and the TGS, times specified in message (1), the nonce from message (1), and TGS identifying information. The ticket itself includes the session key, identifying information for the client, the requested time values, and flags that reflect the status of this ticket and the requested options. These flags introduce significant new functionality to version 5. For now, we defer a discussion of these flags and concentrate on the overall structure of the version 5 protocol.

Let us now compare the **ticket-granting service exchange** for versions 4 and 5. We see that message (3) for both versions includes an authenticator, a ticket, and the name of the requested service. In addition, version 5 includes requested times and options for the ticket and a nonce—all with functions similar to those of message (1). The authenticator itself is essentially the same as the one used in version 4.

Message (4) has the same structure as message (2). It returns a ticket plus information needed by the client, with the information encrypted using the session key now shared by the client and the TGS.

Finally, for the **client/server authentication exchange**, several new features appear in version 5. In message (5), the client may request as an option that mutual authentication is required. The authenticator includes several new fields:

• **Subkey:** The client's choice for an encryption key to be used to protect this specific application session. If this field is omitted, the session key from the ticket (*Kc,v*) is used.
• **Sequence number:** An optional field that specifies the starting sequence number to be used by the server for messages sent to the client during this session. Messages may be sequence numbered to detect replays.

If mutual authentication is required, the server responds with message (6). This message includes the timestamp from the authenticator. Note that in version 4, the timestamp was incremented by one. This is not necessary in version 5, because the nature of the format of messages is such that it is not possible for an opponent to create message (6) without knowledge of the appropriate encryption keys. The subkey field, if present, overrides the subkey field, if present, in message (5). The optional sequence number field specifies the starting sequence number to be used by the client.

3. Discuss about X.509 authentication service in detail.

ITU-T recommendation X.509 is part of the X.500 series of recommendations that define a directory service. X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates .Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. X.509 is based on the use of public-key cryptography and digital signatures.

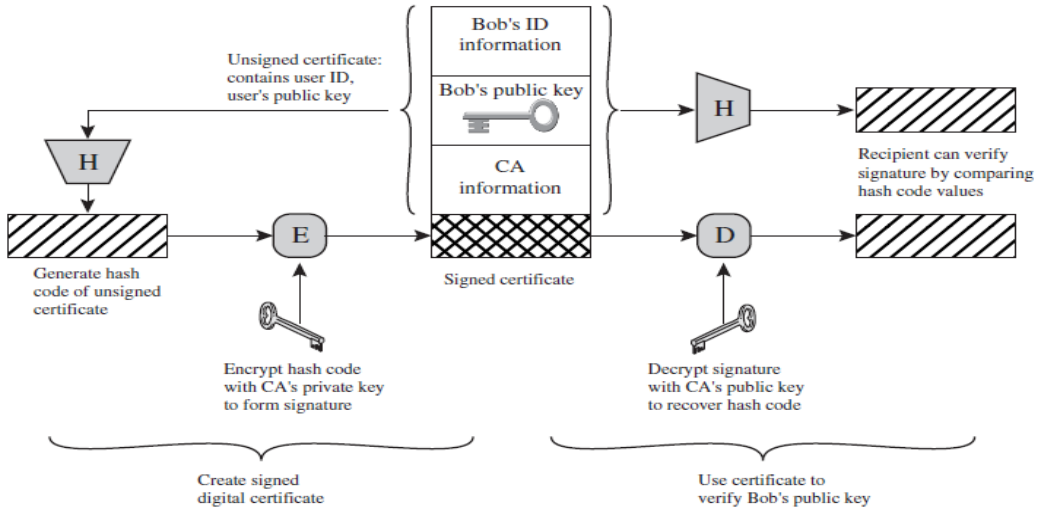


Figure 14.13 Public-Key Certificate Use

Certificates

The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user. The directory server itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.

Version: Differentiates among successive versions of the certificate format; the default is version 1. If the *issuer unique identifier* or *subject unique identifier* are present, the value must be version 2. If one or more extensions are present, the version must be version 3.

• **Serial number:** An integer value unique within the issuing CA that is unambiguously associated with this certificate.

• **Signature algorithm identifier:** The algorithm used to sign the certificate together with any associated parameters.

• **Issuer name:** X.500 is the name of the CA that created and signed this certificate.

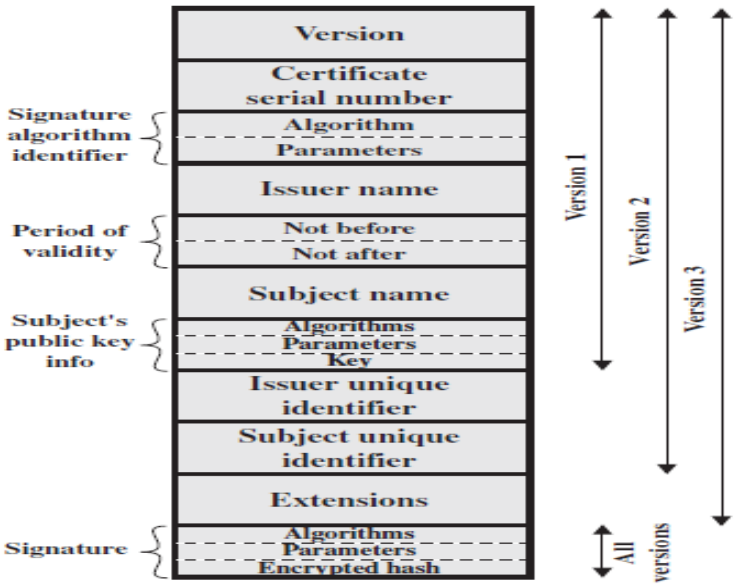
• **Period of validity:** Consists of two dates: the first and last on which the certificate is valid.

• **Subject name:** The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.

• **Subject's public-key information:** The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.

• **Issuer unique identifier:** An optional-bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.

• **Subject unique identifier:** An optional-bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.



(a) X.509 certificate

• **Extensions:** A set of one or more extension fields.
• **Signature:** Covers all of the other fields of the certificate; it contains the hash code of the other fields encrypted with the CA's private key.This field includes the signature algorithm identifier.

The standard uses the following notation to define a certificate:

C<<V>> = CA {V, SN, AI, CA, UCA, A, UA, Ap, TA}

where

Y<<X>> =the certificate of user X issued by certification authority Y
Y {I} =the signing of I by Y. It consists of I with an encrypted hash code appended
V=version of the certificate
SN=serial number of the certificate
AI=identifier of the algorithm used to sign the certificate
CA=name of certificate authority
UCA=optional unique identifier of the CA
A=name of user A
UA=optional unique identifier of the user A
AP=public key of user A
TA =period of validity of the certificate

The CA signs the certificate with its private key. If the corresponding public key is known to a user, then that user can verify that a certificate signed by the CA is valid.

OBTAINING A USER'S CERTIFICATE

User certificates generated by a CA have the following characteristics:

• Any user with access to the public key of the CA can verify the user public key that was certified.
• No party other than the certification authority can modify the certificate without this being detected.

Because certificates are unforgeable, they can be placed in a directory without the need for the directory to make special efforts to protect them. If all users subscribe to the same CA, then there is a common trust of that CA. All user certificates can be placed in the directory for access by all users.

In addition, a user can transmit his or her certificate directly to other users. In either case, once B is in possession of A's certificate, B has confidence that messages it encrypts with A's public key will be secure from eavesdropping and that messages signed with A's private key are unforgeable.

If there is a large community of users, it may not be practical for all users to subscribe to the same CA. Because it is the CA that signs certificates, each participating user must have a copy of the CA's own public key to verify signatures. This public key must be provided to each user in an absolutely secure (with respect to integrity and authenticity) way so that the user has confidence in the associated certificates. Thus, with many users, it may be more practical for there to be a number of CAs, each of which securely provides its public key to some fraction of the users

Now suppose that A has obtained a certificate from certification authority X1 and B has obtained a certificate from CA X2 . If A does not securely know the public key of X2, then B's certificate, issued by X2 , is useless to A.A can read B's certificate, but A cannot verify the signature. However, if the two CAs have securely exchanged their own public keys, the following procedure will enable A to obtain
B's public key.

Step 1 A obtains from the directory the certificate of X2 signed by X1 . Because A securely knows X1's public key,A can obtain X2's public key from its certificate and verify it by means of X1's signature on the certificate.

Step 2 A then goes back to the directory and obtains the certificate of B signed by X2. Because A now has a trusted copy of X2's public key,A can verify the signature and securely obtain B's public key.

A has used a chain of certificates to obtain B's public key.

In the notation of X.509, this chain is expressed as

X1<<X2>>X2<<B>>

In the same fashion, B can obtain A's public key with the reverse chain:

X2<<X1>>X1<<A>>

This scheme need not be limited to a chain of two certificates. An arbitrarily long path of CAs can be followed to produce a chain. All these certificates of CAs by CAs need to appear in the directory, and the user needs to know how they are linked to follow a path to another user's public key certificate.The directory entry for each CA includes two types of certificates:

• **Forward certificates:** Certificates of X generated by other CAs

• **Reverse certificates:** Certificates generated by X that are the certificates of other CAs

REVOCATION OF CERTIFICATES : each certificate includes a period of validity, much like a credit card.Typically, a new certificate is issued just before the expiration of the old one. In addition, it may be desirable on occasion to revoke a certificate before it expires, for one of the following reasons.

1.The user's private key is assumed to be compromised.

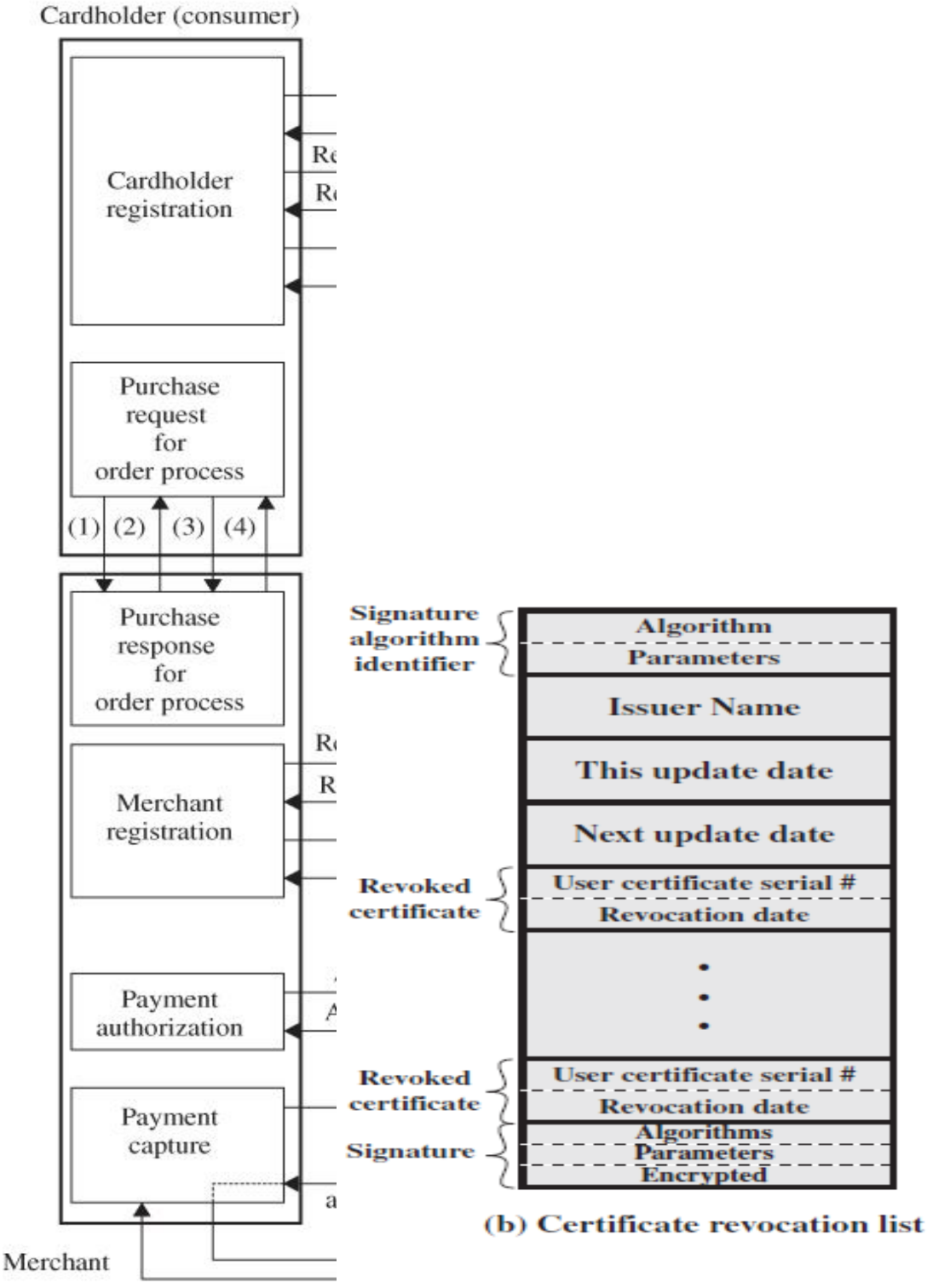
2. The user is no longer certified by this CA.

3. The CA's certificate is assumed to be compromised.

Each certificate revocation list (CRL) posted to the directory is signed by the issuer and includes the issuer's name, the date the list was created, the date the next CRL is scheduled to be issued, and an entry for each revoked certificate.Each entry consists of the serial number of a certificate and revocation date for that certificate.

4. Explain in detail about SET for E-Commerce Transaction.

It describes several transaction protocols needed to securely conduct payment processing by utilizing the cryptographic concepts introduced. Figure shows an overview of secure payment processing and it is worth looking at the outlines of several transaction protocols before reading the following detailed discussion.



1 Cardholder Registration

The cardholder must register with a CA before sending SET messages to the merchant. The cardholder needs a public/private-key pair for use with SET. The scenario of cardholder registration is described in the following.

1. Registration request/response processes

The registration process can be started when the cardholder requests a copy of the CA certificate. When the CA receives the request, it transmits its certificate to the cardholder. The cardholder verifies the CA certificate by traversing the trust chain to the root key. The cardholder holds the CA certificate to use later during the registration process.

- The cardholder sends the *initiate request* to the CA.
- Once the initiate request is received from the cardholder, the CA generates the response and digitally signs it by generating a message digest of the response and encrypting it with the CA's private key.
- The CA sends the *initiate response* along with the CA certificate to the cardholder.
- The cardholder receives the initiate response and verifies the CA certificate by traversing the trust chain to the root key.
- The cardholder verifies the CA certificate by decrypting it with the CA's public key and comparing the result with a newly generated message digest of the initiate response.

2. Registration form process

- The cardholder generates the registration form request.
- The cardholder encrypts the SET message with a random symmetric key (No. 1). The DES key, along with the cardholder's account number, is then encrypted with the CA's public key.
- The cardholder transmits the encrypted registration form request to the CA.
- The CA decrypts the symmetric DES key (No. 1) and cardholder's account number with the CA's private key. The CA then decrypts the registration form request using the symmetric DES key (No. 1).
- The CA determines the appropriate registration form and digitally signs it by generating a message digest of the registration form and encrypting it with the CA's private key.
- The CA sends the registration form and the CA certificate to the cardholder.
- The cardholder receives the registration form and verifies the CA certificate by traversing the trust chain to the root key.
- The cardholder verifies the CA's signature by decrypting it with the CA's public key and comparing the result with a newly generated message digest of the registration form. The cardholder then completes the registration form.

3. Certificate request/response processes

- The cardholder generates the certificate request, including the information entered into the registration form.
- The cardholder creates a message with request, the cardholder's public key and a newly generated symmetric key (No. 2), and digitally signs it by generating a message digest of the cardholder's private key.
- The cardholder encrypts the message with a randomly generated symmetric key (No. 3). This symmetric key, along with the cardholder's account information, is then encrypted with the CA's public key.
- The cardholder transmits the encrypted certificated request messages to the CA.
- The CA decrypts the No. 3 symmetric key and cardholder's account information with the CA's private key and then decrypts the certificate request using this symmetric key.
- The CA verifies the cardholder's signature by decrypting it with the cardholder's public key and comparing the result with a newly generated message digest of the certificate requested.
- The CA verifies the certificate request using the cardholder's account information and information from the registration form.

- Upon verification, the CA creates the cardholder certificate, digitally signing it with the CA's private key.
- The CA generates the certificate response and digitally signs it by generating a message digest of the response and encrypting it with the CA's private key.
- The CA encrypts the certificate response with the No. 2 symmetric key from the cardholder request.
- The CA transmits the certificate response to the cardholder.
- The cardholder verifies the certificate by traversing the trust chain to the root key.
- The cardholder decrypts the response using the symmetric key (No. 2) saved from the cardholder request process.
- The cardholder verifies the CA's signature by decrypting it with the CA's public key and comparing the result with a newly generated message digest of the response.
- The cardholder stores the certificate and information from the response for future e-commerce use.

2 Merchant Registration

Merchants must register with a CA before they can receive SET payment instructions from cardholders. In order to send SET messages to the CA, the merchant must have a copy of the CA's public key which is provided in the CA certificate. The merchant also needs the registration form from the acquirer. The merchant must identify the acquirer to the CA. The merchant registration process consists of five steps as follows: (i) The merchant requests the registration form; (ii) the CA processes this request and sends the registration form; (iii) the merchant requests certificates after receiving the registration certificates; (iv) the CA creates certificates; and (v) the merchant receives certificates.

The detailed steps for the merchant registration are described in what follows.

1. *Registration form process* The registration process starts when the merchant requests the appropriate registration form.

- The merchant sends the initiate request of the registration form to the CA. To register, the merchant fills out the registration form with information such as the merchant's name, address, and ID.
- The CA receives the initiate request.
- The CA selects an appropriate registration form and digitally signs it by generating a message digest of the registration form and encrypting it with the CA's private key.
- The CA sends the registration form along with the CA certificate to the merchant.
- The merchant receives the registration form and verifies the CA certificate by traversing the trust chain to the root key.
- The merchant verifies the CA's signature by decrypting it with the CA's public key and comparing the result with a newly computed message digest of the registration form.
- The merchant creates two public/private-key pairs for use with SET: key encryption and signature.

Thus, the merchant completes the registration form. The merchant takes the registration information (name, address, and ID) and combines it with the public key in a registration message. The merchant digitally signs the registration message. Next, the merchant's software generates a random symmetric key. This random key is used to encrypt the message. The key is then encrypted into the digital envelope using the CA's public key. Finally, the merchant transmits all of these components to the CA.

2. *Certificate request/create process* The merchant starts with the certificate request. When the CA receives the merchant's request, it decrypts the digital envelope to obtain the symmetric encryption key, which it uses to decrypt the registration request.

- The merchant generates the certificate request.
- The merchant creates the message with request and both merchant public keys and digitally signs it by generating a message digest of the certificate request and encrypting it with the merchant's private key.
- The merchant encrypts the message with a random symmetric key (No. 1). This symmetric key, along with the merchant's account data, is then encrypted with the CA's public key.

- The merchant transmits the encrypted certificate request message to the CA.
- The CA decrypts the symmetric key (No. 1) and the merchant's account data with the CA's private key and then decrypts the message using the symmetric key (No. 1).
- The CA verifies the merchant's signature by decrypting it with the merchant's public key and comparing the result with a newly computed message digest of the certificate request.
- The CA confirms the certificate request using the merchant information.
- Upon verification, the CA creates the merchant certificate digitally signing the certificate with the CA's private key.
- The CA generates the certificate response and digitally signs it by generating a message digest of the response and encrypting it with the CA's private key.
- The CA transmits the certificate response to the merchant.
- The merchant receives the certificate response from the CA. The merchant decrypts the digital envelope to obtain the symmetric key. This key is used to decrypt the registration response containing the certificates.
- The merchant verifies the certificates by traversing the trust chain to the root key.
- The merchant verifies the CA's signature by decrypting it with the CA's public key and comparing the result with a newly computed message digest of the certificate response.
- The merchant stores the certificates and information from the response for use in future e-commerce transactions.

3 Purchase Request

The purchase request exchange should take place after the cardholder has completed browsing, selecting, and ordering. Before the end of this preliminary phase occurs, the merchant sends a completed order form to the cardholder (customer). In order to send SET messages to a merchant, the cardholder must have a copy of the certificates of the merchant and the payment gateway. The message from the cardholder indicates which payment card brand will be used for the transaction. The purchase request exchange consists of four messages: initiate request, initiate response, purchase request, and purchase response. The detailed discussions that follow describe each step fully.

1. *Initiate request*
   - The cardholder sends the initiate request to the merchant.
   - The merchant receives the initiate request.
   - The merchant generates the response and digitally signs it by generating a message digest of the response and encrypting it with the merchant's private key.
   - The merchant sends the response along with the merchant and payment gateway certificates to the cardholder.
2. *Initiate response*
   - The cardholder receives the initiate response and verifies the certificates by traversing the trust chain to the root key.
   - The cardholder verifies the merchant's signature by decrypting it with the merchant's public key and comparing the result with a newly computed message digest of the response.
   - The cardholder creates the OM using information from the shopping phase and PM. At this step, the cardholder completes payment instructions.
3. *Purchase request*
   - The cardholder generates a dual signature for the OM and the PM by computing the message digests of both, concatenating the two digests, computing the message digest of the result, and encrypting it using the cardholder's private key.
   - The cardholder generates a random symmetric key (No. 1) and uses it to encrypts the PM. The cardholder then encrypts his or her account number as well as the random symmetric key used to encrypt the PM in a digital envelope using the payment gateway's key.
   - The cardholder transmits the OM and the encrypted PM to the merchant.
   - The merchant verifies the cardholder certificate by traversing the trust chain to the root key.

- The merchant verifies the cardholder's DS on the OM by decrypting it with the cardholder's public key and comparing the result with a newly computed message digest of the concatenation of the message digests of the OM and PM.
- The merchant processes the request, including forwarding the PM to the payment gateway for authorization.

4. *Purchase response*

- The merchant creates the purchase response including the merchant signature certificate and digitally signs it by generating a message digest of the purchase response and encrypting it with the merchant's private key.
- The merchant transmits the purchase response to the cardholder.
- If the transaction was authorized, the merchant fulfills the order to the cardholder.
- The cardholder verifies the merchant signature certificate by traversing the trust chain to the root key.
- The cardholder verifies the merchant's digital signature by decrypting it with the merchant's public key and comparing the result with a newly computed message digest of the purchase response.
- The cardholder stores the purchase response.

4 Payment Authorization

During the processing of an order from a cardholder, the merchant authorizes the transaction. The authorization request and the cardholder payment instructions are then transmitted to the payment gateway.

1. Authorization request

- The merchant creates the authorization request.
- The merchant digitally signs an authorization request by generating a message digest of the authorization request and encrypting it with the merchant's private key.
- The merchant encrypts the authorization request using a random symmetric key (No. 2), which in turn is encrypted with the payment gateway public key.
- The merchant transmits the encrypted authorization request and the encrypted PM from the cardholder purchase request to the payment gateway.
- The gateway verifies the merchant certificate by traversing the trust chain to the root key.
- The payment gateway decrypts the digital envelope of the authorization request to obtain the symmetric encryption key (No. 2) with the gateway private key. The gateway then decrypts the authorization request using the symmetric key (No. 2).
- The gateway verifies the merchant's digital signature by decrypting it with the merchant's public key and comparing the result with a newly computed message digest of the authorization request.
- The gateway verifies the cardholder's certificate by traversing the trust chain to the root key.
- The gateway decrypts the symmetric key (No. 1) and the cardholder account information with the gateway private key. It uses the symmetric key to decrypt the PM.
- The gateway verifies the cardholder's DS on the PM by decrypting it with the cardholder's public key and comparing the result with a newly computed message digest of the concatenation of the message digest of the OM and the PM.
- The gateway ensures consistency between the merchant's authorization request and the cardholder's PM.
- The gateway sends the authorization request through a financial network to the cardholder's financial institution (issuer).

2. Authorization response

- The gateway creates the authorization response message and digitally signs it by generating a message digest of the authorization response and encrypting it with the gateway's private key.
- The gateway encrypts the authorization response with a new randomly generated symmetric key (No. 3). This key is then encrypted with the merchant's public key.

- The gateway creates the capture token and digitally signs it by generating a message digest of the capture token and encrypting it with the gateway's private key.
- The gateway encrypts the capture token with a new symmetric key (No. 4). This key and the cardholder account information are then encrypted with the gateway's public key.
- The gateway transmits the encrypted authorization response to the merchant.
- The merchant verifies the gateway certificate by traversing the trust chain to the root key.
- The merchant decrypts the symmetric key (No. 3) with the merchant's private key and then decrypts the authorization response using the symmetric key (No. 3).
- The merchant verifies the gateway's digital signature by decrypting it with the gateway's public key and comparing the result with a newly computed message digest of the authorization response.
- The merchant stores the encrypted capture token and envelope for later capture processing.
- The merchant then completes processing of the purchase request and the cardholder's order by shipping the goods or performing the services indicated in the order.

5 Payment Capture

After completing the processing of an order from a cardholder, the merchant will request payment. The merchant generates and signs a capture request, which includes the final amount of the transaction, the transaction identifier from the OM, and other information about the transaction. A merchant's payment capture process will be described in detail in the following.

1. Capture request

- The merchant creates the capture request.
- The merchant embeds the merchant certificate in the capture request and digitally signs it by generating a message digest of the capture request and encrypting it with the merchant's private key.
- The merchant encrypts the capture request with a randomly generated symmetric key (No. 5). This key is then encrypted with the payment gateway's public key.
- The merchant transmits the encrypted capture request and encrypted capture token previously stored from the authorization response to the payment gateway.
- The gateway verifies the merchant certificate by traversing the trust chain to the root key.
- The gateway decrypts the symmetric key (No. 5) with the gateway's private key and then decrypts the capture request using the symmetric key (No. 5).
- The gateway verifies the merchant's digital signature by decrypting it with the merchant's public key and comparing the result with a newly computed message digest of the capture request.
- The gateway decrypts the symmetric key (No. 4) with the gateway's private key and then decrypts the capture token using the symmetric key (No. 4).
- The gateway ensures consistency between the merchant's capture request and the capture token.
- The gateway sends the capture request through a financial network to the cardholder's issuer (financial institution).

2. Capture response

- The gateway creates the capture response message, including the gateway signature certificate, and digitally signs it by generating a message digest of the capture response and encrypting it with the gateway's private key.
- The gateway encrypts the capture response with a newly generated symmetric key (No. 6). This key is then encrypted with the merchant's public key.
- The gateway transmits the encrypted capture response to the merchant.
- The merchant verifies the gateway certificate by traversing the trust chain to the root key.
- The merchant decrypts the symmetric key (No. 6) with the merchant's private key and then decrypts the capture response using the symmetric key (No. 6).
- The merchant verifies the gateway's digital signature by decrypting it with the gateway's public key and comparing the result with a newly generated message digest of the capture response.

5. Explain the types of Intrusion Detection Systems.

One of the most publicized attacks to security is the intruder, generally referred to as hacker or cracker.

THREE CLASSES OF INTRUDERS

Masquerader – an individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account.

- **Misfeasor** – a legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuse his or her privileges.
- **Clandestine user** – an individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection.

Two principle countermeasures:

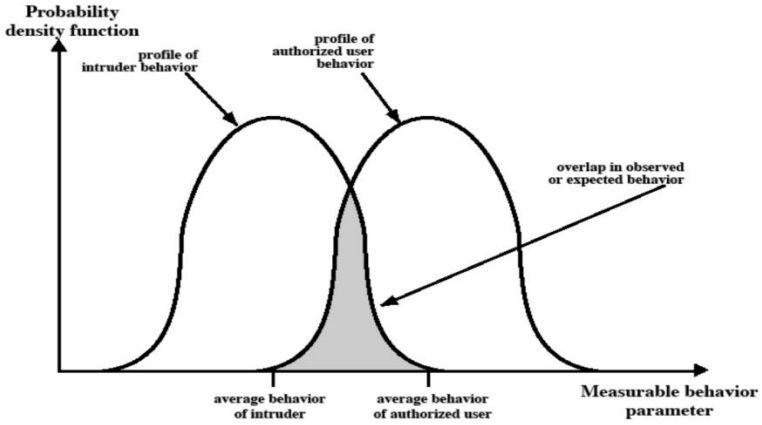
- Detection – concerned with learning of an attack, either before or after its success.
- Prevention – challenging security goal and an uphill bottle at all times.

1. If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised.
2. An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.
3. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

Intrusion detection is based on the assumption that the behaviour of the intruder differs from that of a legitimate user in ways that can be quantified.

APPROACHES TO INTRUSION DETECTION

- Statistical anamoly detection
- Rule based detection



STATISTICAL ANOMALY DETECTION

As was mentioned, statistical anomaly detection techniques fall into two broad categories: threshold detection and profile-based systems. **Threshold detection involves** counting the number of occurrences of a specific event type over an interval of time. If the count surpasses what is considered a reasonable number that one might expect to occur, then intrusion is assumed.Threshold analysis, by itself, is a crude and ineffective detector of even moderately sophisticated attacks. Both the threshold and the time interval must be determined.

Profile-based anomaly detection focuses on characterizing the past behavior of individual users or related groups of users and then detecting significant deviations. A profile may consist of a set of parameters, so that deviation on just a single parameter may not be sufficient in itself to signal an alert.

. Examples of metrics that are useful for profile-based intrusion detection are the following:

- **Counter:** A nonnegative integer that may be incremented but not decremented until it is

reset by management action. Typically, a count of certain event types is kept over a particular period of time. Examples include the number of logins by a single user during an hour, the number of times a given command is executed during a single user session, and the number of password failures during a minute.

- **Gauge:** A nonnegative integer that may be incremented or decremented. Typically, a gauge is used to measure the current value of some entity. Examples include the number of logical connections assigned to a user application and the number of outgoing messages queued for a user process.
- **Interval timer**: The length of time between two related events. An example is the length of time between successive logins to an account.
- **Resource utilization**: Quantity of resources consumed during a specified period.Examples include the number of pages printed during a user session and total time consumed by a program execution.

RULE BASED DETECTION

Rule-based techniques detect intrusion by observing events in the system and applying a set of rules that lead to a decision regarding whether a given pattern of activity is or is not suspicious.

- **Rule-based anomaly detection** is similar in terms of its approach and strengths to statistical anomaly detection. With the rule-based approach, historical audit records are analyzed to identify usage patterns and to generate automatically rules that describe those patterns. Rules may represent past behavior patterns of users, programs, privileges, time slots, terminals, and so on. Current behavior is then observed, and each transaction is matched against the set of rules to determine if it conforms to any historically observed pattern of behavior.
- As with statistical anomaly detection, rule-based anomaly detection does not require knowledge of security vulnerabilities within the system. Rather, the scheme is based on observing past behavior and, in effect, assuming that the future will be like the past
- **Rule-based penetration identification** takes a very different approach to intrusion detection, one based on expert system technology. The key feature of such systems is the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses.

6. Explain about virus structures and types of viruses.

Definition of virus:

- piece of software that infects programs
  - modifying them to include a copy of the virus
  - so it executes secretly when host program is run
- specific to operating system and hardware
  - taking advantage of their details and weaknesses

Phases of virus:

- a typical virus goes through phases of:
  - dormant
  - propagation
  - triggering
  - execution

Types of virus structures

- components:
  - infection mechanism - enables replication
  - trigger - event that makes payload activate
  - payload - what it does, malicious or benign
- prepended / postpended / embedded
- when infected program invoked, executes virus code then original program code
- can block initial infection (difficult) or propogation (with access controls

```
    program V :=

{goto main;
    1234567;

    subroutine infect-executable :=
        {loop:
        file := get-random-executable-file;
        if (first-line-of-file = 1234567)
           then goto loop
           else prepend V to file; }

    subroutine do-damage :=
        {whatever damage is to be done}

    subroutine trigger-pulled :=
        {return true if some condition holds}

main:   main-program :=
        {infect-executable;
        if trigger-pulled then do-damage;
        goto next;}

next:

}
```

In this case, the virus code, V, is prepended to infected programs, and it is assumed that the entry point to the program, when invoked, is the first line of the program. An infected program begins with the virus code and works as follows. The first line of code is a jump to the main virus program. The second line is a special marker that is used by the virus to determine whether or not a potential victim program has already been infected with this virus. When the program is invoked, control is immediately transferred to the main virus program. The virus program first seeks out uninfected executable files and infects them. Next, the virus may perform some action, usually detrimental to the system. This action could be performed every time the program is invoked, or it could be a logic bomb that triggers only under certain conditions. Finally, the virus transfers control to the original program. If the infection phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and uninfected program.

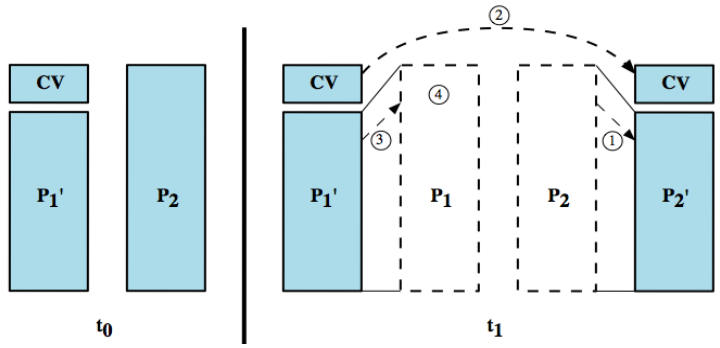
Compression Virus

```
    program CV :=

{goto main;
    01234567;

    subroutine infect-executable :=
          {loop:
               file := get-random-executable-file;
          if (first-line-of-file = 01234567) then goto loop;
       (1)    compress file;
       (2)    prepend CV to file;
       }

main:   main-program :=
          {if ask-permission then infect-executable;
       (3)    uncompress rest-of-file;
       (4)    run uncompressed file;}
       }
```



A way to thwart such a simple means of detecting a virus is to compress the executable file so that both the infected and uninfected versions are of identical length. The code shown from Figure 21.2 shows in general terms the logic required. The key lines in this virus are numbered, and

Figure 21.3 illustrates the operation. In this example, the virus does nothing other than propagate. As in the previous example, the virus may include a logic bomb. We assume that program P_1 is infected with the virus CV. When this program is invoked, control passes to its virus, which performs the following steps:

1. For each uninfected file P_2 that is found, the virus first compresses that file to produce , which is shorter than the original program by the size of the virus.

2. A copy of the virus is prepended to the compressed program.

3. The compressed version of the original infected program, , is uncompressed.

4. The uncompressed original program is executed.

Virus Classification

• **Boot sector infector**:Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.

• **File infector**: Infects files that operating system or shell consider to be executable.

• **Macro virus**: Infects files with macro code that is interpreted by an application.

A virus classification by concealment strategy includes the following categories:

• **Encrypted virus**: the virus creates a random encryption key, stored with the virus, and encrypts the remainder of the virus. When an infected program is invoked, the virus uses the stored random key to decrypt the virus. When the virus replicates, a different random key is selected.

• **Stealth virus:** A form of virus explicitly designed to hide itself from detection by antivirus software. Thus,the entire virus, not just a payload is hidden.

• **Polymorphic virus**: A virus that mutates with every infection, making detection by the "signature"of the virus impossible.

• **Metamorphic virus:** As with a polymorphic virus ,a metamorphic virus mutates with every infection. The difference is that a metamorphic virus rewrites itself completely at each iteration, increasing the difficulty of detection. Metamorphic viruses may change their behavior as well as their appearance.

7. Write about virus and related threats in detail.

VIRUSES AND RELATED THREATS

Perhaps the most sophisticated types of threats to computer systems are presented by programs that exploit vulnerabilities in computing systems.

Malicious Programs

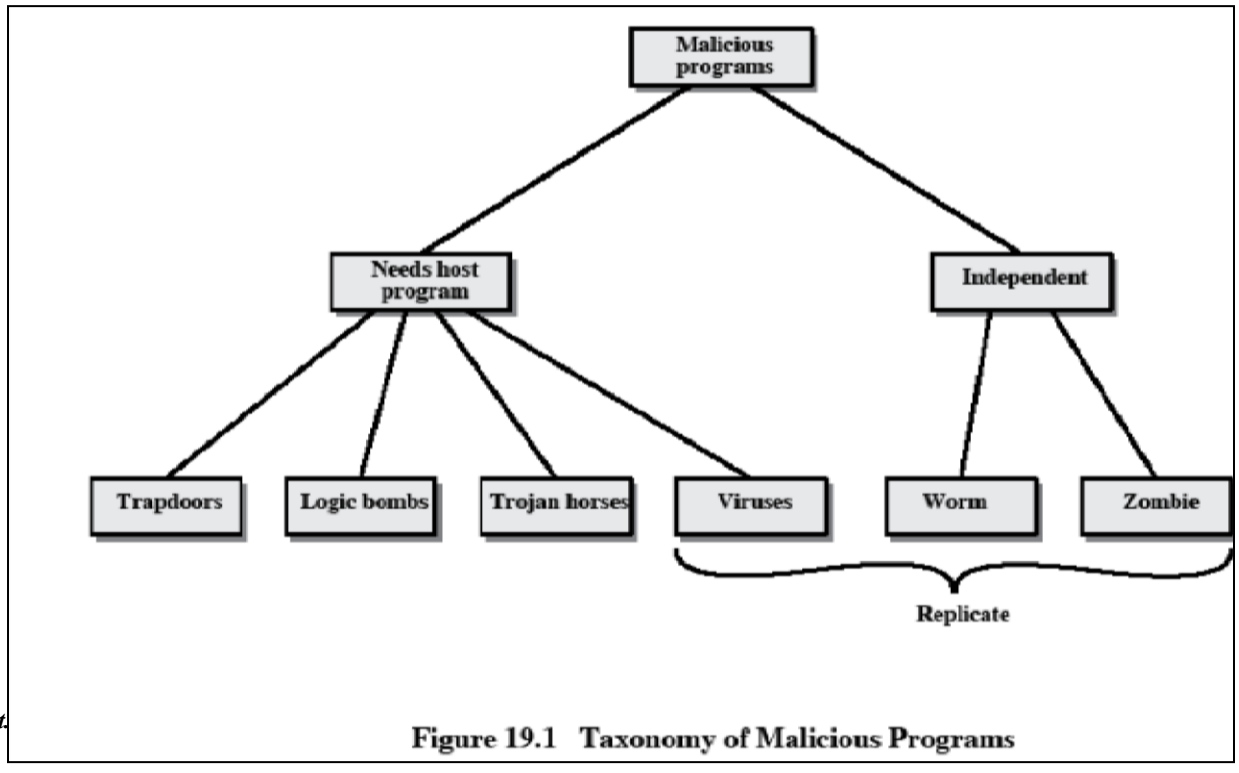


Figure 19.1 Taxonomy of Malicious Programs

Name	Description
Virus	Attaches itself to a program and propagates copies of itself to othe programs
Worm	Program that propagates copies of itself to other computers
Logic bomb	Triggers action when condition occurs
Trojan horse	Program that contains unexpected additional functionality
Backdoor	Program modification that allows unauthorized access t functionality
Exploits	Code specific to a single vulnerability or set of vulnerabilities
Downloaders	Program that installs other items on a machine that is under attack Usually, a downloader is sent in an e-mail.
Auto-rooter	Malicious hacker tools used to break into new machines remotely
Kit (viru generator)	Set of tools for generating new viruses automatically
Spammer programs	Used to send large volumes of unwanted e-mail
Flooders	Used to attack networked computer systems with a large volume o traffic to carry out a denial of service (DoS) attack
Keyloggers	Captures keystrokes on a compromised system
Rootkit	Set of hacker tools used after attacker has broken into a compute system and gained root-level access

Malicious software can be divided into two categories: those that need a host program, and those that are independent.

The former are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program. Viruses, logic bombs, and backdoors are examples. The latter are self-contained programs that can be scheduled and run by the operating system. Worms and zombie programs are examples.

The Nature of Viruses

A virus is a piece of software that can "infect" other programs by modifying them; the modification includes a copy of the virus program, which can then go on to infect other programs. A virus can do anything that other programs do. The only difference is that it attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function, such as erasing files and programs.

During its lifetime, a typical virus goes through the following four phases:

Dormant phase: The virus is idle. The virus will eventually be activated by some

event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit. Not all viruses have this stage.

Propagation phase: The virus places an identical copy of itself into other programs or into certain system areas on the disk. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase.

Triggering phase: The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself.

Execution phase: The function is performed. The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files.

Virus Structure

A virus can be prepended or postpended to an executable program, or it can be embedded in some other fashion. The key to its operation is that the infected program, when invoked, will first execute the virus code and then execute the original code of the program.

An infected program begins with the virus code and works as follows.

The first line of code is a jump to the main virus program. The second line is a special marker that is used by the virus to determine whether or not a potential victim program has already been infected with this virus.

When the program is invoked, control is immediately transferred to the main virus program. The virus program first seeks out uninfected executable files and infects them. Next, the virus may perform some action, usually detrimental to the system.

This action could be performed every time the program is invoked, or it could be a logic bomb that triggers only under certain conditions.

Finally, the virus transfers control to the original program. If the infection phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and uninfected program.

A virus such as the one just described is easily detected because an infected version of a program is longer than the corresponding uninfected one. A way to thwart such a simple means of detecting a virus is to compress the executable file so that both the infected and uninfected versions are of identical length.. The key lines in this virus are numbered, and Figure 19.3 [COHE94] illustrates the operation. We assume that program P1 is infected with the virus CV. When this program is invoked, control passes to its virus, which performs the following steps:

1. For each uninfected file P2 that is found, the virus first compresses that file to produce P'2, which is shorter than the original program by the size of the virus.
2. A copy of the virus is prepended to the compressed program.
3. The compressed version of the original infected program, P'1, is uncompressed.
4. The uncompressed original program is executed.

In this example, the virus does nothing other than propagate. As in the previous example, the virus may include a logic bomb.

Initial Infection

Once a virus has gained entry to a system by infecting a single program, it is in a position to infect some or all other executable files on that system when the infected program executes. Thus, viral infection can be completely prevented by preventing the virus from gaining entry in the first place. Unfortunately, prevention is extraordinarily difficult because a virus can be part of any program outside a system. Thus, unless one is content to take an absolutely bare piece of Iron and write all one's own system and application programs, one is vulnerable.

8. Explain briefly about trusted systems.

One way to enhance the ability of a system to defend against intruders and malicious programs is to implement trusted system technology.

DATA ACCESS CONTROL

Following successful logon, the user has been granted access to one or set of hosts and applications. This is generally not sufficient for a system that includes sensitive data in its database. Through the user access control procedure, a user can be identified to the system. Associated with each user, there can be a profile that specifies permissible operations and file accesses.

The operating system can then enforce rules based on the user profile. The database management system, however, must control access to specific records or even portions of records. The operating system may grant a user permission to access a file or use an application, following which there are no further security checks, the database management system must make a decision on each individual access attempt. That decision will depend not only on the user's identity but also on the specific parts of the data being accessed and even on the information already divulged to the user.

A general model of access control as exercised by an file or database management system is that of an access matrix. The basic elements of the model are as follows:

Subject: An entity capable of accessing objects. Generally, the concept of subject equates with that of process.

Object: Anything to which access is controlled. Examples include files, portion of files, programs, and segments of memory.

Access right: The way in which the object is accessed by a subject. Examples are read, write and execute.

One axis of the matrix consists of identified subjects that may attempt data access. Typically, this list will consist of individual users or user groups. The other axis lists the objects that may be accessed. Objects may be individual data fields. Each entry in the matrix indicates the access rights of that subject for that object. The matrix may be decomposed by columns, yielding **access control lists.** Thus, for each object, an access control list lists users and their permitted access rights. The access control list may contain a default, or public, entry.

Decomposition by rows yields **capability tickets**. A capability ticket specifies authorized objects and operations for a user. Each user has a number of tickets and may be authorized to loan or give them to others. Because tickets may be dispersed around the system, they present a greater security problem than access control lists. In particular, the ticket must be unforgeable. One way to accomplish this is to have the operating system hold all tickets on behalf of users. These tickets would have to be held in a region of memory inaccessible to users.

	Program1	...	SegmentA	SegmentB
Process1	Read Execute		Read Write	
Process2				Read
.				
.				
.				

a. Access matrix

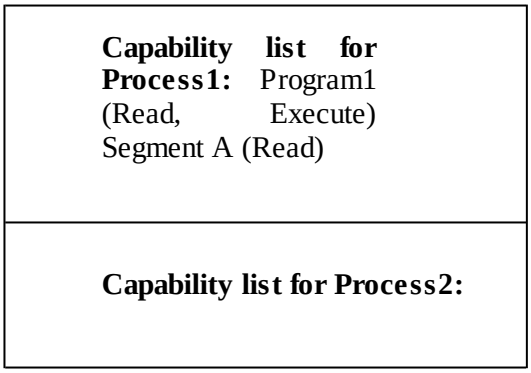
Access control list for Program1:
Process1 (Read, Execute)

Access control list for Segment A:
Process1 (Read, Write)

Access control list for Segment B:

Process2 (Read)

b. Access control list



The concept of Trusted Systems

A multilevel secure system must enforce:

- **No read up:** A subject can only read an object of less or equal security level. This is referred to as **simple security property.**
- **No write down:** A subject can only write into an object of greater or equal security level. This is referred to as ***-property (star property).**

These two rules, if properly enforced, provide multilevel security.

REFERENCE MONITOR CONCEPT

The reference monitor is a controlling element in the hardware and operating system of a computer that regulates the access of subjects to objects on the basis of security parameters of the subject and object. The reference monitor has access to a file, known as the security kernel database that lists the access privileges (security clearance) of each subject and the protection attributes (classification level) of each object.

9. Explain in detail about different types of firewalls and discuss its configuration.

Firewall characteristics:

All traffic from inside to outside, and vice versa, must pass through the firewall.

This is achieved by physically blocking all access to the local network except via the firewall. Various configurations are possible.

only authorized traffic, as defined by the local security policy, will be allowed to pass.

Various types of firewalls are used, which implement various types of security policies.

the firewall itself is immune to penetration. This implies that use of a trusted system with a secure operating system. This implies that use of a trusted system with a secure operating system.

Four techniques that firewall use to control access and enforce the site's security policy is as follows:

Service control – determines the type of internet services that can be accessed, inbound or outbound. The firewall may filter traffic on this basis of IP address and TCP port number; may provide proxy software that receives and interprets each service request before passing it on; or may host the server software itself, such as web or mail service.

Direction control – determines the direction in which particular service request may be initiated and allowed to flow through the firewall.

User control – controls access to a service according to which user is attempting to access it.

Behavior control – controls how particular services are used.

Capabilities of firewall

A firewall defines a single choke point that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks.

A firewall provides a location for monitoring security related events. Audits and alarms can be implemented on the firewall system.

A firewall is a convenient platform for several internet functions that are not security related.

A firewall can serve as the platform for IPsec.

Limitations of firewall

The firewall cannot protect against attacks that bypass the firewall. Internal systems may have dial-out capability to connect to an ISP. An internal LAN may support a modem pool that provides dial-in capability for traveling employees and telecommuters.

the firewall does not protect against internal threats. The firewall does not protect against internal threats, such as a disgruntled employee or an employee who unwittingly cooperates with an external attacker.

The firewall cannot protect against the transfer of virus-infected programs or files. Because of the variety of operating systems and applications supported inside the perimeter, it would be impractical and perhaps impossible for the firewall to scan all incoming files, e-mail, and messages for viruses.

Types of firewalls

There are 3 common types of firewalls.

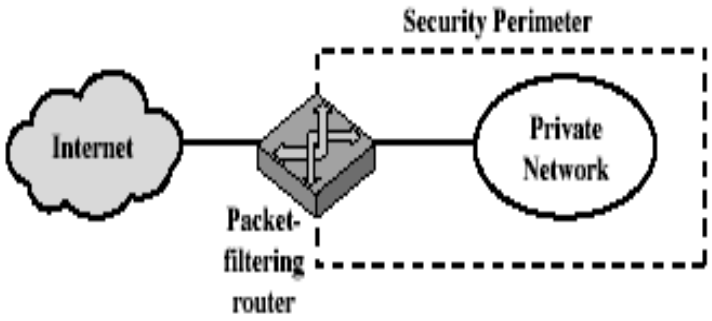
Packet filters
Application-level
gateways Circuit-
level gateways

Packet filtering router

A packet filtering router applies a set of rules to each incoming IP packet and then forwards or discards the packet. The router is typically configured to filter packets going in both directions. Filtering rules are based on the information contained in a network packet:

Source IP address – IP address of the system that originated the IP packet.
Destination IP address – IP address of the system, the IP is trying to
reach. Source and destination transport level address – transport level
port number. IP protocol field – defines the transport protocol.

Interface – for a router with three or more ports, which interface of the router the packet come from or which interface of the router the packet is destined for.



(a) Packet-filtering router

The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header. If there is a match to one of the rules, that rule is invoked to determine whether to forward or discard the packet. If there is no match to any rule, then a default action is taken. Two default policies are possible:

Default = discard: That which is not expressly permitted is prohibited.
Default = forward: That which is not expressly prohibited is permitted.

The default discard policy is the more conservative. Initially everything is blocked, and services

must be added on a case-by-case basis. This policy is more visible to users, who are most likely to see the firewall as a hindrance. The default forward policy increases ease of use for end users but provides reduced security.

Advantages of packet filter router

Simple
Transparent to users Very fast

Weakness of packet filter firewalls

Because packet filter firewalls do not examine upper-layer data, they cannot prevent attacks that employ application specific vulnerabilities or functions.

Because of the limited information available to the firewall, the logging functionality present in packet filter firewall is limited.

It does not support advanced user authentication schemes.

They are generally vulnerable to attacks such as layer address spoofing.

Some of the attacks that can be made on packet filtering routers and the appropriate counter measures are the following:

IP address spoofing –

the intruders transmit packets from the outside with a source IP address field containing an address of an internal host.

Countermeasure: to discard packet with an inside source address if the packet arrives on an external interface.

Source routing attacks – the source station specifies the route that a packet should take as it crosses the internet; i.e., it will bypass the firewall.

Countermeasure: to discard all packets that uses this option.

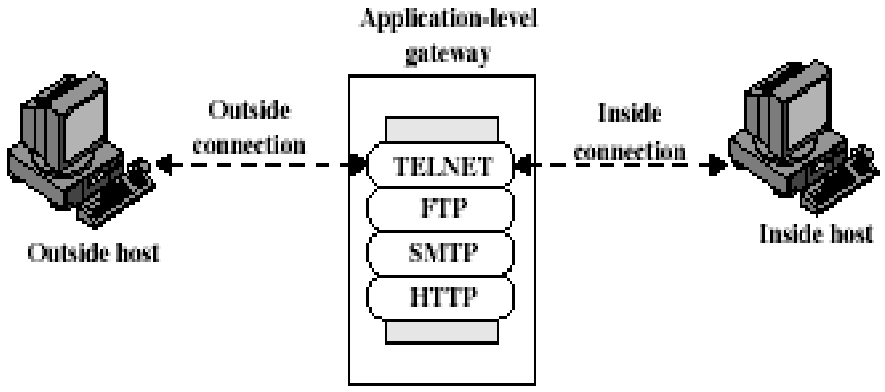
Tiny fragment attacks – the intruder create extremely small fragments and force the TCP header information into a separate packet fragment. The attacker hopes that only the first fragment is examined and the remaining fragments are passed through.

Countermeasure: to discard all packets where the protocol type is TCP and the IP Fragment offset is equal to 1.

Application level gateway

An Application level gateway, also called a proxy server, acts as a relay of application level traffic. The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints.

Application level gateways tend to be more secure than packet filters. It is easy to log and audit all incoming traffic at the application level. A prime disadvantage is the additional processing overhead on each connection.

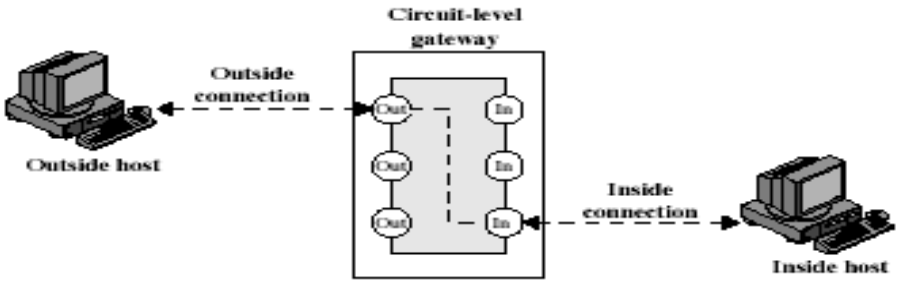


(b) Application-level gateway

Circuit level gateway

Circuit level gateway can be a stand-alone system or it can be a specified function performed by an application level gateway for certain applications. A Circuit level gateway does not permit an end- to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outer host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed.

A typical use of Circuit level gateways is a situation in which the system administrator trusts the internal users. The gateway can be configured to support application level or proxy service on inbound connections and circuit level functions for outbound connections.



(c) Circuit-level gateway

Firewall configurations

There are 3 common firewall configurations.

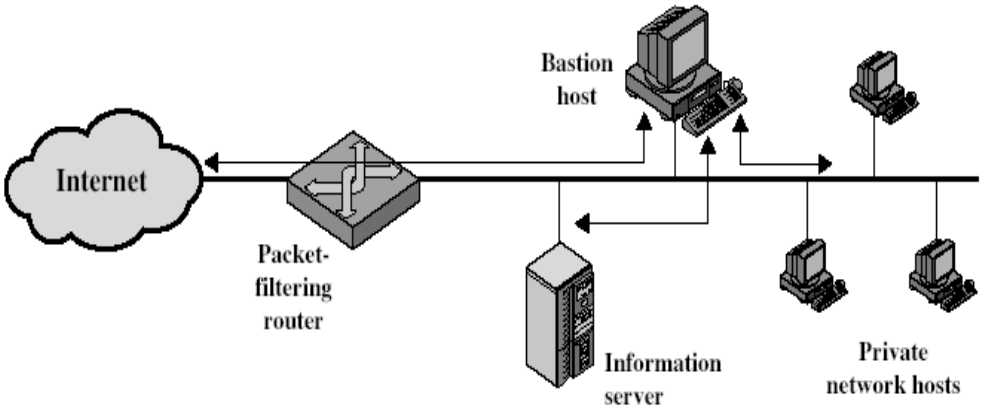
1. Screened host firewall, single-homed basiton configuration

In the **screened host firewall, single-homed bastion** configuration, the firewall consists of two systems: a packet-filtering router and a bastion host. Typically, the router is configured so that

1. For traffic from the Internet, only IP packets destined for the bastion host are allowed in.

2. For traffic from the internal network, only IP packets from the bastion host are allowed out.

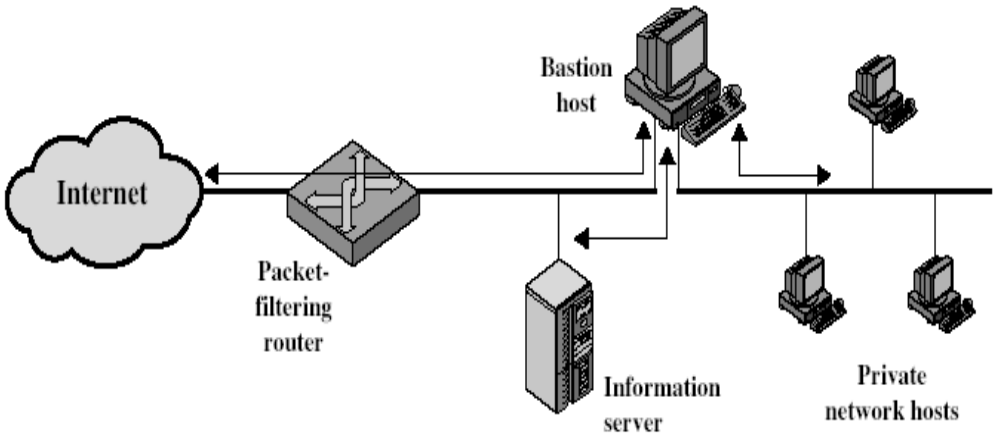
The bastion host performs authentication and proxy functions. This configuration has greater security than simply a packet-filtering router or an application-level gateway alone, for two reasons. First, this configuration implements both packet-level and application-level filtering, allowing for considerable flexibility in defining security policy. Second, an intruder must generally penetrate two separate systems before the security of the internal network is compromised. This configuration also affords flexibility in providing direct Internet access. For example, the internal network may include a public information server, such as a Web server, for which a high level of security is not required. In that case, the router can be configured to allow direct traffic between the information server and the Internet.



(a) Screened host firewall system (single-homed bastion host)

2. Screened host firewall, dual homed basiton configuration

In the previous configuration, if the packet filtering router is compromised, traffic could flow directly through the router between the internet and the other hosts on the private network. This configuration physically prevents such a security break.



(b) Screened host firewall system (dual-homed bastion host)

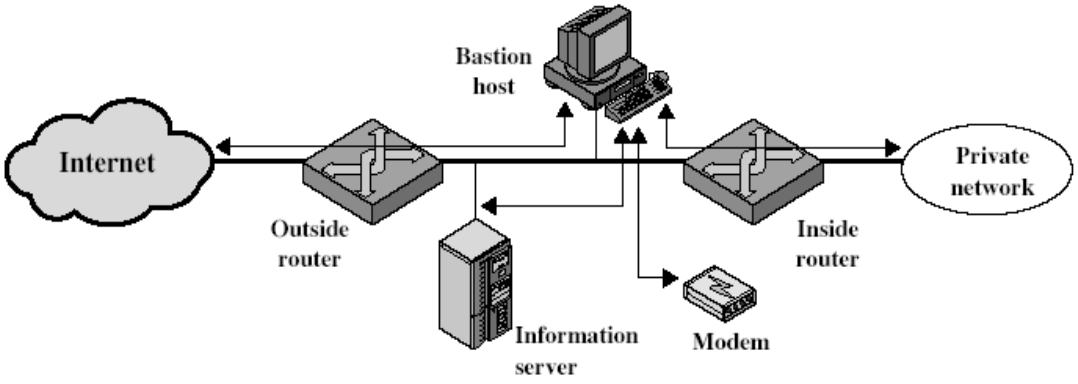
3. Screened subnet firewall configuration

In this configuration, two packet filtering routers are used, one between the basiton host and internet and one between the basiton host and the internal network. This configuration creates an isolated subnetwork, which may consist of simply the basiton host but may also include one or more information servers and modems for dial-in capability. Typically both the internet and the internal network have access to hosts on the screened subnet, but traffic across the screened subnet is blocked. This configuration offers several advantages:

There are now three levels of defense to thwart intruders.

The outside router advertises only the existence of the screened subnet to the internet; therefore the internal network is invisible to the internet.

Similarly, the inside router advertises only the existence of the screened subnet to the internal network; therefore the systems on the internal network cannot construct direct routes to the internet.



(c) Screened-subnet firewall system

10. (i). Explain any two advanced anti-virus techniques in detail.

Generic decryption

- use CPU simulator to check program signature & behavior before actually running it
- To detect virus structure, executable files are run through a GD scanner, which contains the following elements:
- CPU emulator: A software-based virtual computer. Instructions in an executable file are interpreted by the emulator rather than executed on the underlying processor. The emulator includes software versions of all registers and other processor hardware, so that the underlying processor is unaffected by programs interpreted on the emulator.
- Virus signature scanner: A module that scans the target code looking for known virus signatures.
- Emulation control module: Controls the execution of the target code.

Digital immune system (IBM)

- general purpose emulation & virus detection
- any virus entering org is captured, analyzed, detection/shielding created for it, removed .
- The monitoring program in PC forwards a copy of any program thought to be infected to an administrative machine within the organization.
- The administrative machine encrypts the sample and sends it to a central virus analysis machine.
- The virus analysis machine then produces a prescription for identifying and removing the virus.
- The resulting prescription is sent back to the administrative machine.
- The administrative machine forwards the prescription to the infected client.
- The prescription is also forwarded to other clients in the organization.
- Subscribers around the world receive regular antivirus updates that protect them from the new virus.

Behavior-Blocking Software

- behavior-blocking software integrates with the operating system of a host computer and monitors program behavior in real-time for malicious actions.
- If the behavior blocker detects that a program is initiating would-be malicious behaviors as it runs, it can block these behaviors in real-time and/or terminate the offending software

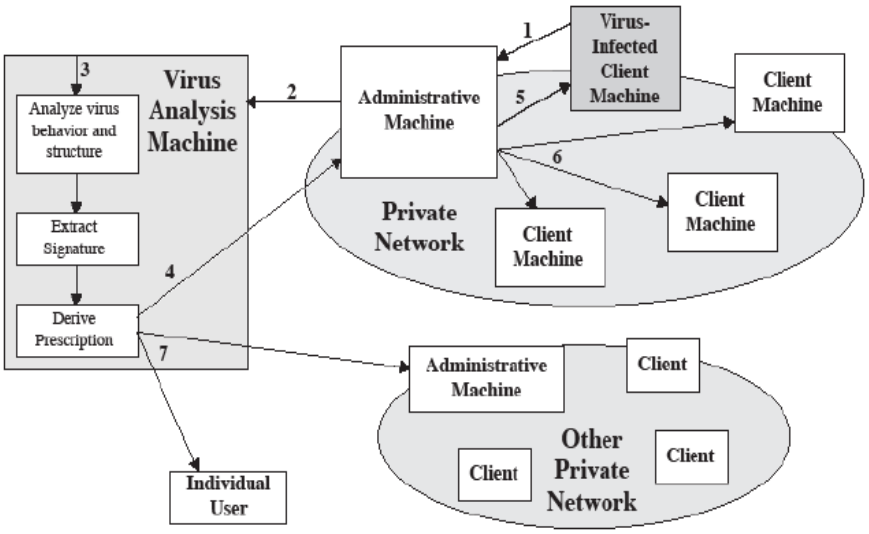


Figure 19.5 Digital Immune System

(ii).Write down the four generations of antivirus software.

There are four generations of antivirus software: First generation: simple scanners
Second generation: heuristic scanners Third generation: activity traps
Fourth generation: full-featured protection

A first-generation scanner requires a virus signature to identify a virus.. Such signature-specific scanners are limited to the detection of known viruses. Another type of first-generation scanner maintains a record of the length of programs and looks for changes in length.

A second-generation scanner does not rely on a specific signature. Rather, the scanner uses heuristic rules to search for probable virus infection. One class of such scanners looks for fragments of code that are often associated with viruses.

Another second-generation approach is integrity checking. A checksum can be appended to each program. If a virus infects the program without changing the checksum, then an integrity check will catch the change. To counter a virus that is sophisticated enough to change the checksum when it infects a program, an encrypted hash function can be used. The encryption key is stored separately from the program so that the virus cannot generate a new hash code and encrypt that. By using a hash function rather than a simpler checksum, the virus is prevented from adjusting the program to produce the same hash code as before.

Third-generation programs are memory-resident programs that identify a virus by its actions rather than its structure in an infected program. Such programs have the advantage that it is not necessary to develop signatures and heuristics for a wide array of viruses. Rather, it is necessary only to identify the small set of actions that indicate an infection is being attempted and then to intervene.

Fourth-generation products are packages consisting of a variety of antivirus techniques used in conjunction. These include scanning and activity trap components. In addition, such a package includes access control capability, which limits the ability of viruses to penetrate a system and then limits the ability of a virus to update files in order to pass on the infection.

UNIT-5
PART A

1. List out any four security services for e-mail

- Privacy
- Authentication
- Integrity
- Non-Repudiation
- Proof of Submission
- Proof of delivery
- Message flow confidentiality

2. State the way in which threat to Privacy in e-mail occur?

- An eavesdropper can listen to the message
- Relay nodes (Routers or Mail Forwarders) have the capability to store messages for monitoring purpose ,which is a threat

3. How one can achieve end-to-end privacy in e-mail?

End-to-end encryption (E2EE) is a method of secure communication that prevents third-parties from accessing data while it's transferred from one end system or device to another.

4. List out the ways by which authentication of the source be done?

- Source authentication based on public key technologies
- Source authentication based on secret keys
- Source authentication with distribution lists

5. State the difference between repudiation and non-repudiation?

- Repudiation : act of denying that we sent the message
- Non-Repudiation : sender can't deny later that she/he did not send that message

6. Realize Backdating problem in with an example

A dishonest buyer may report that the key has been stolen to renege an online shopping purchase order

7. List out the mutable fields of IPv4 and IPv6 Authentication Header?

IPv4 - Type of Service, Flags, Fragment Offset,TTL,Header Checksum
IPv6 - Type of Service, FlowLabel, Hop Limit

8. State the role of ESP

- Encapsulating Security Payload
- It performs Encryption and/or integrity protection

9. Do we need and AH for Secure Communication?

- AH protects IP Header also.
- Whereas ESP protects ESP Header alone

10. What is the role of IKE?

- Internet Key Exchange
- Is a protocol for mutual authentication and to establish a session key
- Alternatives to IKE are :
    - Photuris – a signed Diffie-Hellman Key Exchange, using an stateless cookie
    - SKIP (Simple Key –Management for Internet Protocols) – uses long term Diffie-Hellman public keys

11. What are the phases of IKE?

- Phase 1
    - Does mutual authentication and establish session keys
    - Expensive
    - Phase 1 exchange is also known as ISAKMP SA [Internet Security Association and Key Management Protocol] or IKE SA
- Phase 2
    - Simpler and cheaper
    - Use the session key created out of phase-1 exchange

12. What are the types of Phase 1 Exchange of IKE?

- Main Mode - uses six messages
  - Message 1 - Alice to Bob – sends a cookie and request crypto suite support
  - Message 2 - Bob to Alice – responds with a cookie and his supporting Crypto Suite
  - Message 3 & 4 – Diffie-Hellman Key Exchange
  - Message 5 & 6 – revealing identity & Proof
- Aggressive Mode - uses only three messages
  - First two messages include key exchange
  - Message 2 & 3 - each party prove themselves to each other

13. List out the types of keys used during Phase I of IKE

- Pre-shared secret key
- Public encryption key
- Public signature key

14. Write about Phase 2 of IKE

- It uses "Quick mode"
- Once an IKE SA is set up between Alice and Bob, either Alice or Bob can initiate an IPsec SA through Quick mode
- Establishes an ESP and/or AH SA

15. List out the protocols involved in IKE Phase 1

1. Public Signature keys, main mode
2. Public Signature keys, aggressive mode
3. Public Encryption key, main mode, original
4. Public Encryption key, aggressive mode, original
5. Public Encryption key, main mode, revised
6. Public Encryption key, aggressive mode, revised
7. Shared secret key, main mode
8. Shared secret key, aggressive mode

16. State the role of SSL/TLS

9. Secure Socket Layer/Transport Layer Security
10. Allows two parties to authenticate and establish a session key to cryptographically protect the remainder of the session
11. Runs as a user level process. So, no changes need to be done with the underlying OS
12. Runs on top of TCP. So, reliable communication is possible

13. There are three versions : SSL v1, SSL v2, SSL v3

17. List out the attacks fixed by SSL v3

- Downgrade Attack
- Truncation Attack

18. Define: Downgrade Attack

- It occurs in SSLv2 and is fixed in SSLv3
- No integrity protection for the initial handshake
- An active attacker can make alice and bob to agree upon a weaker cipher

19. Define: Truncation Attack

- SSLv2 depends on TCP Connection Closing to indicate there is no more data to send.
- TCP Connection close is not cryptographically protected.
- Attacker may utilize this to close the connection
- SSLv3 added a "finished message" to indicate there is no ore data to send.

20. Define: Ephemeral Key

- Introduced in SSLv3
- Long-term key
- Can be used for many connections
- Useful for exportability and enhances security

21. What is the purpose of S/MIME?

- Abbreviated as : Secure/Multipurpose Internet Mail Extension

- Specified as a standard way of encoding arbitrary data in email such as pictures, rich text,video clips ,binary files etc., along with adding signed and encrypted data

22. Write about PGP?

- Secure Mail Protocol
- Abbreviated as Pretty Good Privacy
- Proposed by Phil Zimmermann
- PGP performs encryption and integrity protection on files
- PGP uses public key cryptography for personal use
- Certificates are optional in PGP

PART-B

1. i) List out the security services for electronic mail?(6)

Privacy—the ability to keep anyone but the intended recipient from reading the message.

Authentication—reassurance to the recipient of the identity of the sender.

Integrity—reassurance to the recipient that the message has not been altered since it was transmitted by the sender.

Non-repudiation—the ability of the recipient to prove to a third party that the sender really did send the message. This feature is also sometimes called **third party authentication**. The term non-repudiation means that the sender cannot later deny sending the message.

Proof of submission—verification given to the sender that the message was handed to the mail delivery system (the same basic idea as sending certified mail through the U.S. postal service). With certified postal mail you just receive proof that you sent something to a particular address on a particular date, but with electronic mail it is possible to have the mail system verify acceptance of the contents of a particular message, perhaps by signing the message digest of the contents of the message.

Proof of delivery—verification that the recipient received the message. Postal mail has a similar feature (return receipt requested), but again it only verifies that something was delivered on a particular date to the recipient. With electronic mail it is possible to verify the contents, as we mentioned under proof of submission.

Message flow confidentiality—an extension of privacy such that Carol not only cannot know the content of the message Alice sent Bob, but cannot even determine whether Alice sent Bob a message. A**nonymity**—the ability to send a message so that the recipient can't find out the identity of the sender.

Containment—the ability of the network to keep certain security levels of information from leaking out of a particular region.

Audit—the ability of the network to record events that might have some security relevance, such as that Alice sent a message to Bob on a particular date. This would be fairly straightforward to implement, but is not mentioned in any of the secure mail standards, so we don't have a section on it.

Accounting—the ability of the mail system to maintain system usage statistics. In addition to providing clues for system resource management, this information allows the mail system to charge its clients according to their usage. For example, the system might charge by number of messages sent, as long as the system itself authenticates the source of each message to ensure that the proper party is billed. Again, there's not much to say about this, so we don't have a separate section on it.

S**elf destruct**—an option allowing a sender to specify that a message should be destroyed after delivery to the recipient. This allows Alice to send a message to Bob that Bob cannot forward or store. The mail system will decrypt and display the message, but then delete it. (*Good morning Mr. Phelps...*). This can be implemented by marking the message as a *self-destruct* message, and having the mail program at the destination cooperate by deleting the message immediately after displaying it.

Message sequence integrity—reassurance that an entire sequence of messages arrived in the order transmitted, without any loss.

ii) Explain in detail about privacy in electronic mail?(10)

End-to-End Privacy

Alice might want to send a message to Bob in such a way that only Bob can read it. She can't depend on the network keeping the message secret, but she can ensure that nobody but Bob can read the message by using cryptography to encrypt the message.

The natural assumption is that if Alice wants to send Bob an encrypted message, she encrypts it using Bob's public key (if public key technology is being used for keys) or by using the key she shares with Bob (if secret key technology is being used). However, this is not how mail encryption is generally done, for several reasons:

- If Alice has a long message to send to multiple recipients, the long message would have to be encrypted once for each recipient, producing a different version to be sent to each recipient.
- Public key encryption is far less efficient than secret key encryption. So it is desirable to encrypt the message with secret key encryption even if Bob's key is a public key.
- *Session Key Establishment*, it's not desirable to use a long-term key more than necessary. So to preserve the useful lifetime of the long-term key Alice and Bob share, it is preferable to encrypt as little as possible with a key that is expensive to replace.

So the way it is done is that Alice chooses a random secret key S to be used only for encrypting that one message. She encrypts the message with S, encrypts S with Bob's key, and transmits both quantities to Bob. Even if the message is being sent to multiple recipients, she only encrypts the message once, with key S. But she encrypts S once for each recipient, with the appropriate key, and includes each encrypted S with the encrypted message.

Let's assume Bob's key is K_{Bob}, Carol's key is K_{Carol}, and Ted's key is K_{Ted}. If public key technology is being used, then K_{Bob} is Bob's public key. If secret key technology is being used, then K_{Bob} is the key that Alice shares with Bob. The message Alice would like to send to Bob, Carol, and Ted is m. S is the secret key Alice chose specifically for encrypting m. (We'll write $K\{x\}$ to indicate x encrypted with key K.) The mail message Alice will send includes:

- Bob's name; $K_{Bob}\{S\}$
- Carol's name; $K_{Carol}\{S\}$
- Ted's name; $K_{Ted}\{S\}$
- $S\{m\}$

Privacy with Distribution List Exploders

Suppose Alice is sending a message to a distribution list which will be remotely exploded, and Bob is only one of the recipients. Assume the distribution list is stored at some node remote from Alice, and that Alice does not even know the individuals on the distribution list. We can't assume she has keys for all the members of the distribution list. Instead, she has a key only for the distribution list exploder. Alice does not need to treat the distribution list exploder any differently than any other recipient of a message. It is merely someone she shares a key with and sends messages to. The distribution list exploder will need keys for all the individuals on the list.

As described in the end-to-end case, Alice will choose a random per-message secret key S, and encrypt the message with S. The distribution list exploder will decrypt S (but it does not need to decrypt the message!), and re-encrypt S with the key for each recipient to whom it is forwarding the message.

Local exploding requires different mechanisms. Alice still has to trust the maintainer of the mailing list, since a bad guy could insert extra names into the distribution list. The distribution list is most likely just a list of names. Alice will not be able to send a secure message to a name without establishing a key for that individual.

2. Write about PGP in detail.

PGP is an open-source, freely available software package for e-mail security.It provides authentication through the use of digital signature, confidentiality through the use of symmetric block encryption, compression using the ZIP algorithm, and e-mail compatibility using the radix-

64 encoding scheme. PGP incorporates tools for developing a public-key trust model and public-key certificate management.

OPERATIONAL DESCRIPTION

AUTHENTICATION

1. The sender creates a message.
2. SHA-1 is used to generate a 160-bit hash code of the message.
3. The hash code is encrypted with RSA using the sender's private key, and the result is prepended to the message.
4. The receiver uses RSA with the sender's public key to decrypt and recover the hash code.
5. The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic

CONFIDENTIALITY Another basic service provided by PGP is confidentiality, which is provided by encrypting messages to be transmitted or to be stored locally as files.
1. The sender generates a message and a random 128-bit number to be used as a session key for this message only.
2. The message is encrypted using CAST-128 (or IDEA or 3DES) with the session key.
3. The session key is encrypted with RSA using the recipient's public key and is prepended to the message.
4. The receiver uses RSA with its private key to decrypt and recover the session key.
5. The session key is used to decrypt the message

CONFIDENTIALITY AND AUTHENTICATION : both services may be used for the same message. First, a signature is generated for the plaintext message and prepended to the message.Then the plaintext message plus signature is encrypted using CAST-128 (or IDEA or 3DES), and the session key is encrypted using RSA (or ElGamal). This sequence is preferable to the opposite: encrypting the message and then generating a signature for the encrypted message. It is generally more convenient to store a signature with a plaintext version of a message. Furthermore, for purposes of third-party verification, if the signature is performed
first, a third party need not be concerned with the symmetric key when verifying the signature. when both services are used, the sender first signs the message with its own private key, then encrypts the message with a session key, and finally encrypts the session key with the recipient's public key.

COMPRESSION As a default, PGP compresses the message after applying the signature but before encryption.This has the benefit of saving space both for e-mail transmission and for file storage.
1. The signature is generated before compression for two reasons:
a. It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification. If one signed a compressed document, then it would be necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required.
b. Even if one were willing to generate dynamically a recompressed message for verification, PGP's compression algorithm presents a difficulty.The algorithm is not deterministic; various implementations of the algorithm achieve different tradeoffs in running speed versus compression ratio and, as a result, produce different compressed forms. However, these different compression algorithms are interoperable because any version of the algorithm can correctly decompress the output of any other version. Applying the hash function and signature after compression would constrain all PGP implementations
to the same version of the compression algorithm.
2. Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult.

E-MAIL COMPATIBILITY

PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters. The scheme used for this purpose is radix-64 conversion. Each group of three

octets of binary data is mapped into four ASCII characters. The use of radix 64 expands a message by 33%.

One noteworthy aspect of the radix-64 algorithm is that it blindly converts the input stream to radix-64 format regardless of content, even if the input happens to be ASCII text. Thus, if a message is signed but not encrypted and the conversion is applied to the entire block, the output will be unreadable to the casual observer, which provides a certain level of confidentiality.

. On transmission (if it is required), a signature is generated using a hash code of the uncompressed plaintext. Then the plaintext (plus signature if present) is compressed. Next, if confidentiality is required, the block (compressed plaintext or compressed signature plus plaintext) is encrypted and prepended with the public-keyencrypted symmetric encryption key. Finally, the entire block is converted to radix-64 format.

On reception, the incoming block is first converted back from radix-64 format to binary. Then, if the message is encrypted, the recipient recovers the session key and decrypts the message.The resulting block is then decompressed. If the message is signed, the recipient recovers the transmitted hash code and compares it to its own calculation of the hash code.

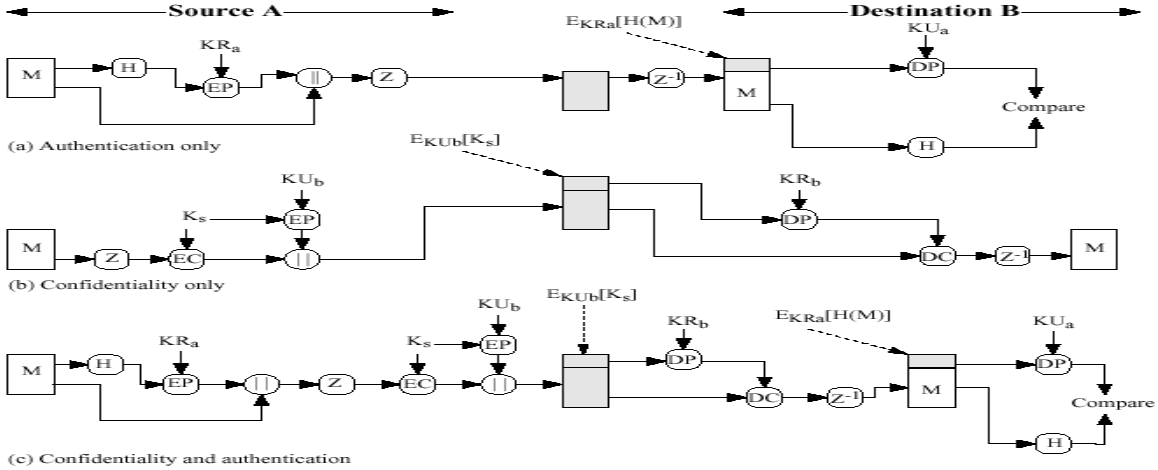
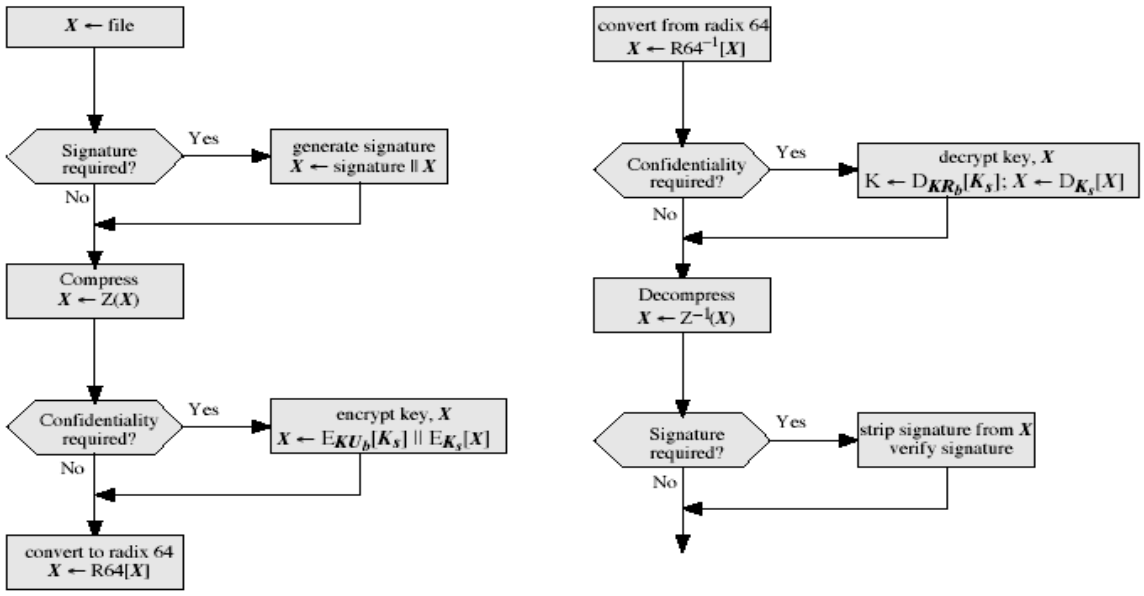


Figure 12.1 PGP Cryptographic Functions



(a) Generic Transmission Diagram (from A) (b) Generic Reception Diagram (to B)

GENERAL FORMAT PGP MESSAGE

The **signature component** includes the following.

• **Timestamp:** The time at which the signature was made.
• **Message digest:** The 160-bit SHA-1 digest encrypted with the sender's private signature key. The digest is calculated over the signature timestamp concatenated with the data portion of the message component.
Leading two octets of message digest: Enables the recipient to determine if the correct public key was used to decrypt the message digest for authentication by comparing this plaintext copy of the first two octets with the first two octets of the decrypted digest. These octets also serve as a 16-bit frame check
sequence for the message.
• **Key ID of sender's public key:** Identifies the public key that should be used to decrypt the message digest and, hence, identifies the private key that was used to encrypt the message digest

The **session key component** includes the session key and the identifier of the recipient's public key that was used by the sender to encrypt the session key.
The entire block is usually encoded with radix-64 encoding

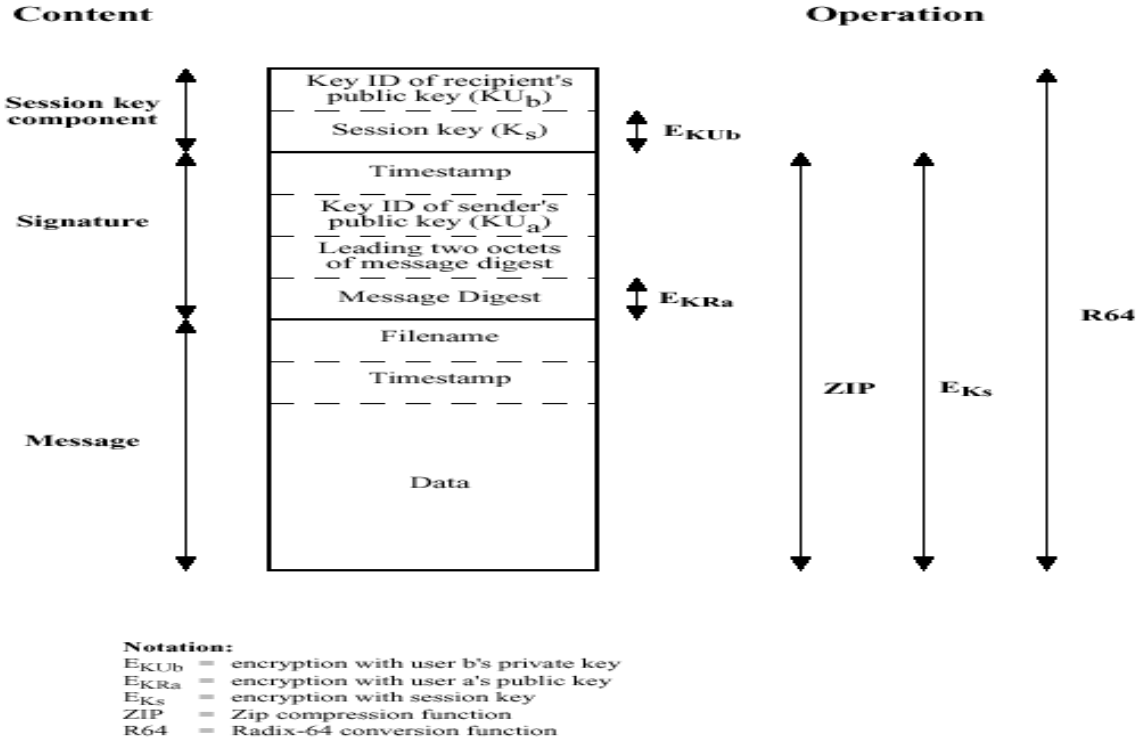


Figure 12.3 General Format of PGP Message (from A to B)

PGP MESSAGE GENERATION

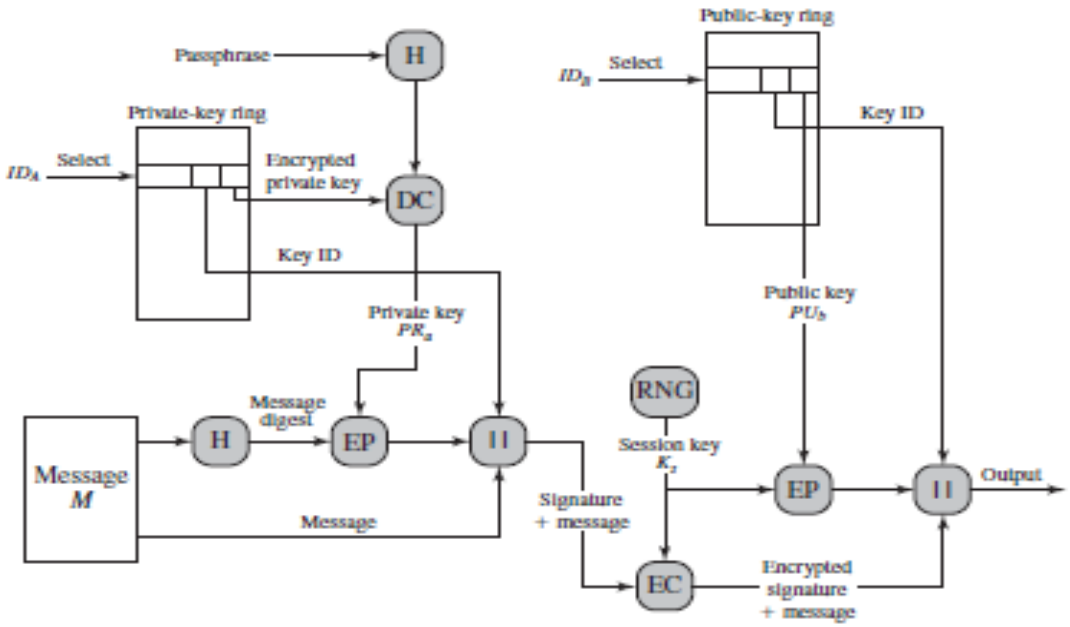


Figure 18.5 PGP Message Generation (from User A to User B: no compression or radix-64 conversion)

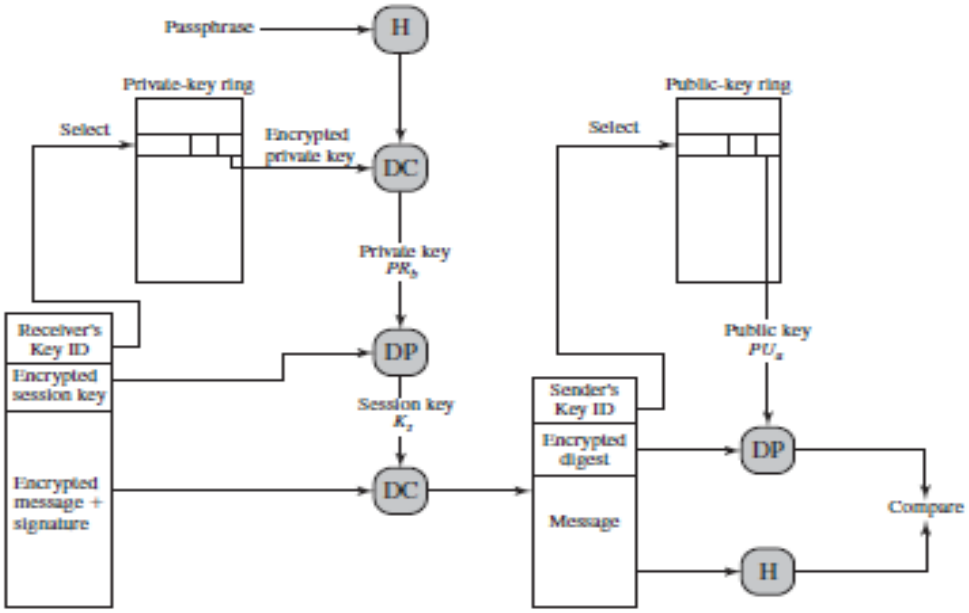


Figure 18.6 PGP Message Reception (from User A to User B; no compression or radix-64 conversion)

3. Discuss about S/MIME in detail.

1. Secure/Multipurpose Internet Mail Extension (S/MIME) is a security enhancement to the MIME Internet e-mail format standard .

MULTIPURPOSE INTERNET MAIL EXTENSIONS

Multipurpose Internet Mail Extension (MIME) is an extension to the RFC 5322 framework that is intended to address some of the problems and limitations of the use of Simple Mail Transfer Protocol (SMTP)

1. SMTP cannot transmit executable files or other binary objects.

2. SMTP cannot transmit text data that includes national language characters,

3. SMTP servers may reject mail message over a certain size.

4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.
5. SMTP gateways to X.400 electronic mail networks cannot handle nontextual data included in X.400 messages.

MIME – HEADER FILES

The five header fields defined in MIME are

• **MIME-Version:** Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.
• **Content-Type:** Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise deal with the data in an appropriate manner.
• **Content-Transfer-Encoding:** Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.
• **Content-ID:** Used to identify MIME entities uniquely in multiple contexts.
• **Content-Description:** A text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

MIME CONTENT TYPES

Type	Subtype	Description
Text	Plain	Unformatted text; may be ASCII or ISO 8859.
	Enriched	Provides greater format flexibility.
Multipart	Mixed	The different parts are independent but are to be transmitted together. They should be presented to the receiver in the order that they appear in the mail message.
	Parallel	Differs from Mixed only in that no order is defined for delivering the parts to the receiver.
	Alternative	The different parts are alternative versions of the same information. They are ordered in increasing faithfulness to the original, and the recipient's mail system should display the "best" version to the user.
	Digest	Similar to Mixed, but the default type/subtype of each part is message/rfc822.
Message	rfc822	The body is itself an encapsulated message that conforms to RFC 822.
	Partial	Used to allow fragmentation of large mail items, in a way that is transparent to the recipient.
	External-body	Contains a pointer to an object that exists elsewhere.
Image	jpeg	The image is in JPEG format, JFIF encoding.
	gif	The image is in GIF format.
Video	mpeg	MPEG format.
Audio	Basic	Single-channel 8-bit ISDN mu-law encoding at a sample rate of 8 kHz.
Application	PostScript	Adobe Postscript
	octet-stream	General binary data consisting of 8-bit bytes.

MIME TRANSFER ENCODINGS

Table 18.4 MIME Transfer Encodings

7bit	The data are all represented by short lines of ASCII characters.
8bit	The lines are short, but there may be non-ASCII characters (octets with the high-order bit set).
binary	Not only may non-ASCII characters be present, but the lines are not necessarily short enough for SMTP transport.
quoted-printable	Encodes the data in such a way that if the data being encoded are mostly ASCII text, the encoded form of the data remains largely recognizable by humans.
base64	Encodes data by mapping 6-bit blocks of input to 8-bit blocks of output, all of which are printable ASCII characters.
x-token	A named nonstandard encoding.

S/MIME FUNCTIONALITY

Enveloped data: This consists of encrypted content of any type and encrypted content encryption keys for one or more recipients.

• **Signed data:** A digital signature is formed by taking the message digest of the content to be signed and then encrypting that with the private key of the signer. The content plus signature are then encoded using base64 encoding..

• **Clear-signed data:** As with signed data, a digital signature of the content is formed. However, in this case, only the digital signature is encoded using base64.As a result, recipients without S/MIME capability can view the message content, although they cannot verify the signature.

• **Signed and enveloped data:** Signed-only and encrypted-only entities may be nested, so that encrypted data may be signed and signed data or clear-signed data may be encrypted.

CRYPTOGRAPHIC ALGORITHMS

- Create message digest to form digital signature
  - Must use SHA-1, Should support MD5
- Encrypt message digest to form signature
  - Must support DSS, Should support RSA
- Encrypt session key for transmission
  - Should support Diffie-Hellman, Must support RSA
- Encrypt message for transmission with one-time session key
  - Must support triple DES, Should support AES, Should support RC2/40
- Create a message authentication code
  - Must support HMAC with SHA-1, Should support HMAC with SHA-1

S/MIME – ENHANCED SECURITY SERVICES

- Signed receipts --The receiver returns a signed receipt back to the sender to verify the message arrived
- Security labels --Permission, priority or role of message being sent
- Secure mailing lists--Sending to multiple recipients at once securely by using a public key for the whole mailing list

S/MIME CERTIFICATE PROCESSING

- S/MIME uses X.509 v3 certificates
- managed using a hybrid of a strict X.509 CA hierarchy & PGP's web of trust
- each client has a list of trusted CA's certs
- and own public/private key pairs & certs
- certificates must be signed by trusted CA's

S/MIME Messages

SECURING A MIME ENTITY

S/MIME secures a MIME entity with a signature, encryption, or both. A MIME entity may be an entire message or if the MIME content type is multipart, then a MIME entity is one or more of the subparts of the message.The MIME entity is prepared according to the normal rules for MIME message preparation.

ENVELOPED DATA

The steps for preparing an envelopedData MIME entity are

1. Generate a pseudorandom session key for a particular symmetric encryption algorithm (RC2/40 or triple DES).

2. For each recipient, encrypt the session key with the recipient's public RSA key.

3. For each recipient, prepare a block known as RecipientInfo that contains an identifier of the recipient's public-key certificate,4 an identifier of the algorithm used to encrypt the session key, and the encrypted session key.

4. Encrypt the message content with the session key.

The RecipientInfo blocks followed by the encrypted content constitute the envelopedData. This information is then encoded into base64.

SIGNED DATA The signedData smime-type can be used with one or more signers. The steps for preparing a signedData MIME entity are

1. Select a message digest algorithm (SHA or MD5).
2. Compute the message digest (hash function) of the content to be signed.
3. Encrypt the message digest with the signer's private key.
4. Prepare a block known as SignerInfo that contains the signer's publickey certificate, an identifier of the message digest algorithm, an identifier of the algorithm used to encrypt the message digest, and the encrypted message digest.
The signedData entity consists of a series of blocks, including a message digest algorithm identifier, the message being signed, and SignerInfo. The signedData entity may also include a set of public-key certificates sufficient to constitute a chain from a recognized root or top-level certification authority to the signer. This information is then encoded into base64.

REGISTRATION REQUEST

The certification request includes certification RequestInfo block, followed by an identifier of the public-key encryption algorithm, followed by the signature of the certificationRequestInfo block made using the sender's private key. The certificationRequestInfo block includes a name of the certificate subject (the entity whose public key is to be certified) and a bit-string representation of the user's public key.

CERTIFICATES-ONLY MESSAGE

A message containing only certificates or a certificate revocation list (CRL) can be sent in response to a registration request.The message is an application/pkcs7-mime type/subtype with an smime-type parameter of
degenerate. The steps involved are the same as those for creating a signedData message, except that there is no message content and the signerInfo field is empty.

USER AGENT ROLE

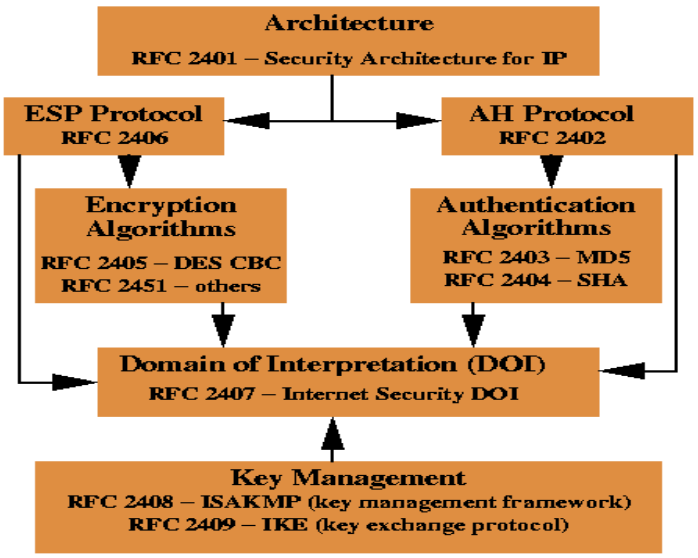
An S/MIME user has several key-management functions to perform.
• **Key generation:** The user of some related administrative utility (e.g., one associated with LAN management) MUST be capable of generating separate Diffie-Hellman and DSS key pairs and SHOULD be capable of generating RSA key pairs. Each key pair MUST be generated from a good source of nondeterministic random input and be protected in a secure fashion. A user agent SHOULD generate RSA key pairs with a length in the range of 768 to 1024 bits and MUST NOT generate a length of less than 512 bits.
• **Registration:** A user's public key must be registered with a certification authority in order to receive an X.509 public-key certificate.

4. Describe about IP security.

Application and advantages:

- Secure remote internet access.
- Setup communication with other organization
- IPsec allows connectivity between various brances of an organization
- It is transparent to end users,Ipsec works at network layer.

- have two security header extensions:
- Authentication Header (AH)
- Encapsulating Security Payload (ESP)



Ipsec Service:
1. Access control

2.Connectionless integrity
3. Data origin authentication
4. Traffic flow confidentiality

Modes of operation:

Tunnel mode:An encrypted tunnel established between two hosts.

Transport mode: it is useful when we are interested in host to host encryption.

Internet Key Exchange protocol is used for key management.

Security Association: It is an agreement between communication parties.it is simplex and unidirectional.

Authentication Header:

- It provides data integrity and authentication of IP packets.
- It is based on MAC protocol.

AH header fields are as follows:

Next header-identifies the type of header.

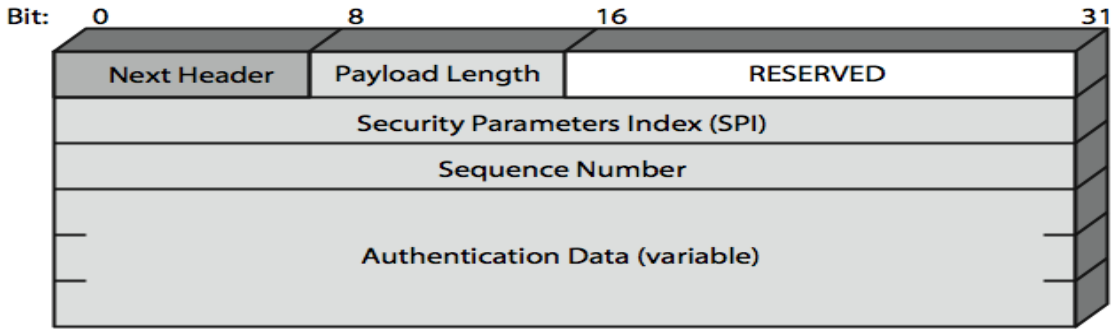
Payload length:8 bit field contains the length of AH in 32 bit words minus 2.

Reserved: 16 bit field reserved for future use.

Security Parameter Index: This is used in combination with source and destination address.

Sequence number::32 bit field to prevent replay attacks.

Authentication data:variable length field contains authentication data called as Integrity check value.



Modes of operation:

AH Transport Mode:

The position of AH is between th e original IP header and original TCP header of IP packet.

AH Tunnel Mode: the entire original IP packet is authenticated and the AH is between the original IP header and new outer ip header. The inner IP header contains the ultimate source and destination IP addresses.Whereas the outer IP header contains different IP addresses.

Scope of AH Authentication IP v4

Before applying AH :

Orig IP Header	TCP	Data

Transport mode :

Orig IP Header	AH	TCP	Data

Tunnel Mode :

New IP Header	AH	Orig IP Header	TCP	Data

Scope of AH Authentication IP v6

Before applying AH :

Orig IP Hdr	Extn Hdr if present	TCP	Data

Transport mode :

Orig IP	Hop by hop, dest	AH	dest	TCP	Data

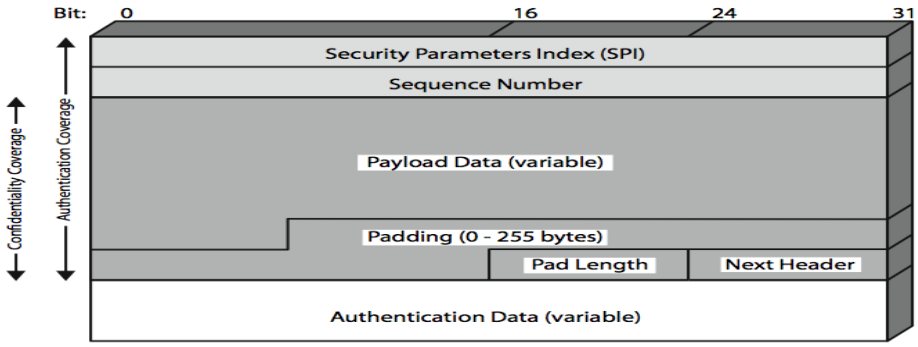
Header	routing, fragment				

Tunnel Mode :

New IP Header	Ext headers	AH	Orig IP Header	Ext headers	TCP	Data

Encapsulating Security Payload (ESP)

ESP provides confidentiality and integrity of messages.



ESP Transport mode:

- Transport mode is used to encrypt and optionally authenticate the data carried by IP.
- ESP is inserted into IP packet before the transport layer header.
- ESP trailer is added after the ip packet.ESP authentication data field is added after the ESP trailer.
- The entire transport layer segments and the ESP trailer are encrypted and cipher text is authenticated.

ESP Tunnel Mode

This encrypts the entire IP packet. Here ESP header is pre fixed to the packet and then packet along with ESP trailer is encrypted

Scope of ESP Encryption and Authentication IP v4

Transport mode :

Orig IP Hdr	ESP Hdr	TCP	Data	ESP trlr	ESP auth

Tunnel Mode :

New IP Header	ESP Header	Orig IP Header	TCP	Data	ESP trlr	ESP auth

Scope of ESP Encryption and Authentication IP v6

Transport mode :

Orig IP Hdr	Hop by hop, dest routing,	AH	dest	TCP	Data	ESP trlr	ESP auth

Tunnel Mode :

New IP Header	Ext headers	ESP Header	Orig IP Header	Ext headers	TCP	Data	ESP trlr	ESP auth

5. Describe all 8 phase 1 IKE protocols with neat diagram?

In the first message, Alice transmits her "cookie" value. After that, all messages start with the cookie pair (initiator cookie, responder cookie), and that pair serves as the IKE connection identifier. Note that in an IKE exchange between Alice and Bob, all messages start with the same cookie pair, in the same order. If Alice initiated the IKE connection, her cookie value always

appears in the "initiator cookie" field. To reduce clutter, we won't write "(initiator cookie, responder cookie)" in the figures for the messages. To reduce clutter, CP indicates crypto proposal, and CPA indicates crypto proposal accepted.

1. Public Signature Keys, Main Mode

In this mode, the two parties have public keys capable of doing signatures. Both endpoint identifiers are hidden from an eavesdropper, but an active attacker can figure out the initiator's identity.

The reason for including nonces in messages 3 and 4 is so that Alice and/or Bob can save themselves computation by using the same Diffie-Hellman private value for many exchanges. If they *always* use the same value, then there will not be perfect forward secrecy, so it's a good idea to change it periodically. Mysteriously (and for no good reason), in the public signature key variants, K is also a function of the cookies.

If Bob instead appended the information from message 6 onto message 4 then the exchange would complete in 5 messages instead of 6. However, there is a disadvantage of doing that, since Alice and Bob can't be computing the Diffie-Hellman key in parallel.

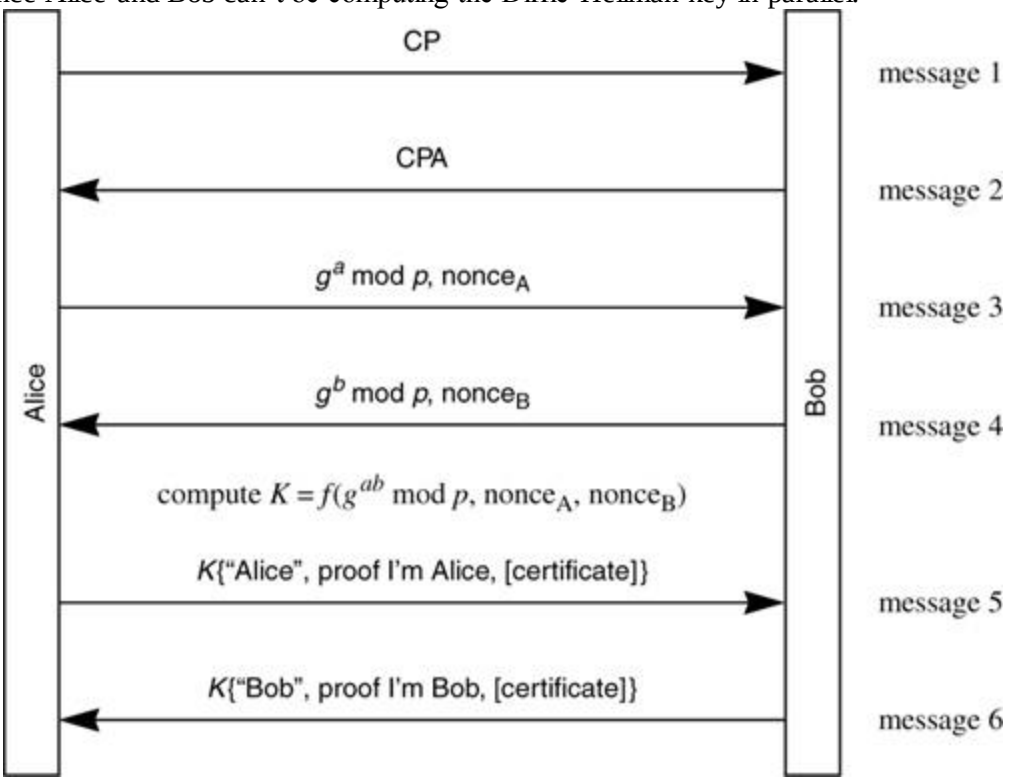


Figure: Public signature keys, main mode

2. Public Signature Keys, Aggressive Mode

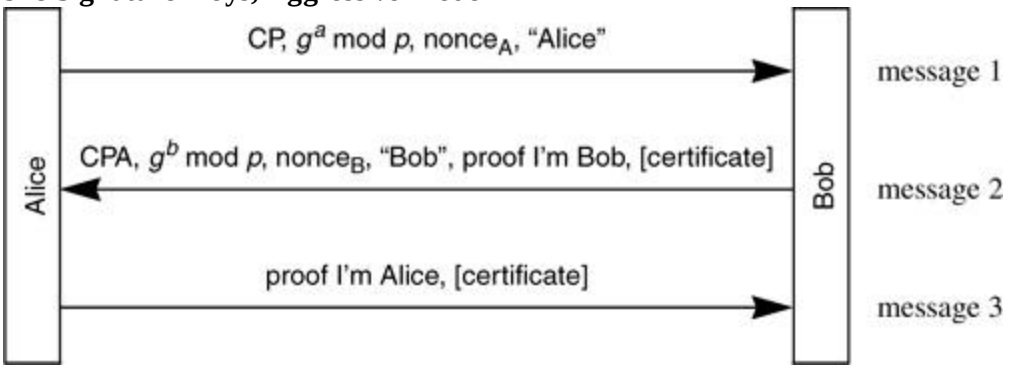


Figure: Public signature keys, aggressive mode

Note that messages 2 and 3 are not encrypted, even though the same information is encrypted in the main mode public signature key variant. The identities could have been encrypted and have the exchange still be 3 messages

3. Public Encryption Key, Main Mode, Original

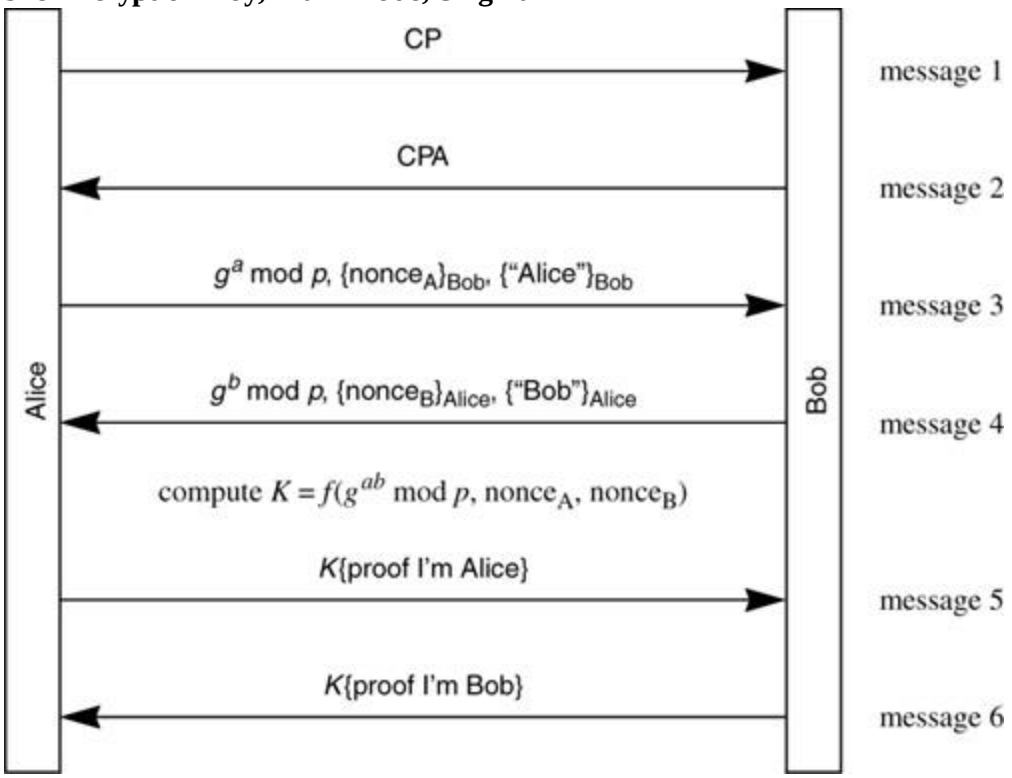


Figure: Public Encryption Keys, main mode, original protocol

IKE specifies 4 different phase-1 protocols for public encryption keys, because the original protocols (main mode and aggressive mode) were inefficient (separately encrypted multiple fields with public keys, requiring multiple private key operations). It's astonishing that they left the original protocols in the spec once they redesigned them.

A problem with this variant is that in message 3 there are two fields separately encrypted with Bob's public key, so he needs to do two private key operations to decrypt it. Likewise Alice needs to do two private key operations to decrypt message 4.

Another problem would occur if a nonce or a name were larger than the public key with which it is being encrypted. The spec could have defined some sort of CBC mode for encrypting something larger than a key, but it didn't. Note that with X.500 names it would not be far-fetched for a name to be very long.

Alice and Bob prove they know their private keys because they are able to decrypt the nonce from the other side. They prove this both by knowing *K* and the hashes used in the proofs in messages 5 and 6, since they are all functions of (among other things) the nonces.

There's no way for either Alice or Bob to ask the other side to send them their certificate! If you don't already know the other side's public key, you can't use this protocol. And if neither side knows the other side's public key without their certificate, there is no way, even if Alice and Bob could request certificates in messages 1 and 2, for them to send their certificates without divulging their identity.

There is an option of having Alice send, in message 3, a hash of Bob's certificate. The reasoning is that Bob might have multiple public keys, that Alice would know that he had multiple public keys, and that he wouldn't have lots of certificates so a hash of the certificate Alice happens to have for Bob's key would be recognized by Bob.

4. Public Encryption Key, Aggressive Mode, Original

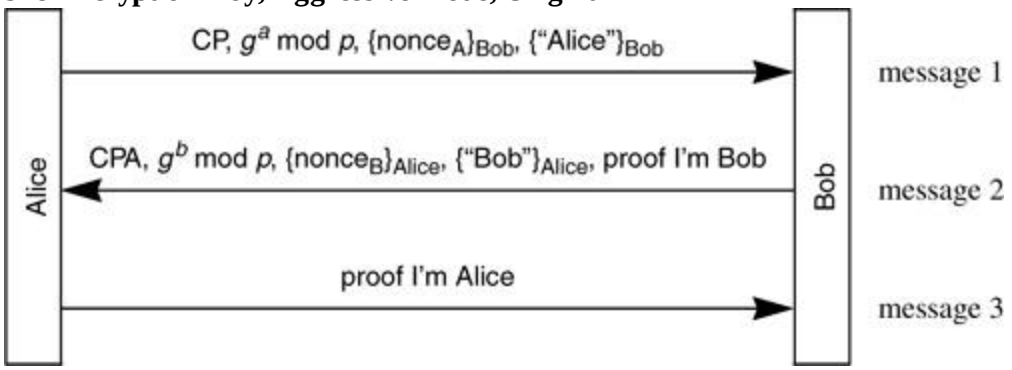


Figure: Public Encryption Keys, aggressive mode, original protocol

This protocol is almost the same as the main mode version except that messages 1 and 2 are removed (and crypto suites other than Diffie-Hellman group are negotiated in parallel with the other information in messages 1 and 2) and Bob provides his proof in message 2 rather than, as in main mode, doing it after Alice presents her proof. The proof consists of a hash of the nonce presented by the other side (which requires knowledge of the private key to decrypt), along with the Diffie-Hellman values and the cookie values.

5. Public Encryption Key, Main Mode, Revised

The public encryption protocol was revised to require only a single private key operation on each side (rather than two in the original). This is done by encrypting with a secret key which is a function of the nonce, and the nonce is encrypted with the other side's public key. Thus the other side uses its private key to retrieve the nonce, but then decrypts the other fields with a secret key.

The revised protocol allows Alice to optionally send Bob her certificate. It still has the problem that Alice needs to know Bob's public key.

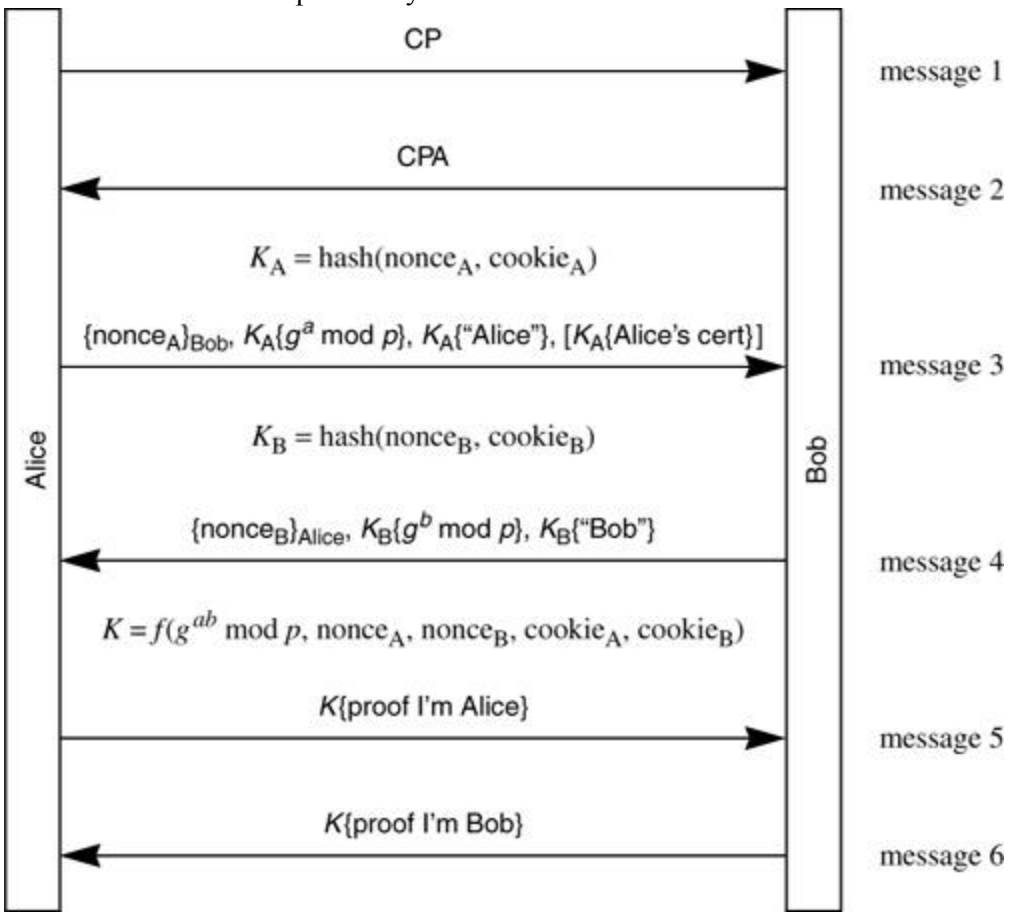


Figure: Public Encryption Keys, main mode, revised protocol

6. Public Encryption Key, Aggressive Mode, Revised

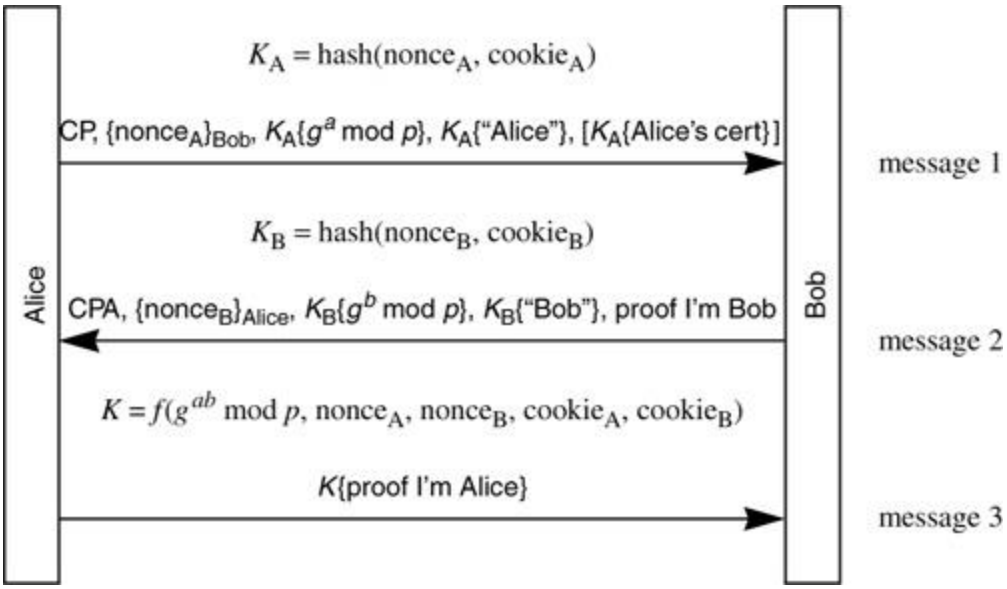


Figure : Public Encryption Keys, aggressive mode, revised protocol

7. Shared Secret Key, Main Mode

This is the one required protocol. And it is the most broken. One situation in which this protocol might be useful is in the "road warrior" case, where an employee's laptop is configured with a shared secret with the company's firewall. This would allow the employee to authenticate to the firewall and establish an encrypted tunnel. But the way this mode is designed requires the identities to be the IP addresses. This makes it useless in the road warrior case, because a road warrior's IP address is dependent on where she is that day. And if the identity has to be the IP address, why go to all the work of hiding it by using the 6-message main mode protocol?

The problem with this protocol is that Alice sends her identity in message 5 encrypted with a key K which is a function of the shared secret J. Bob can't decrypt message 5 in order to find out who he's talking to unless he knows J, which means he needs to know who he's talking to. The working group noticed this, and rather than fixing the protocol (which wouldn't have been hard), they instead said that in this mode Alice's identity has to be her IP address! This makes it almost useless in practice, and it certainly doesn't hide identities.

This protocol can be fixed to allow arbitrary identities by not making K a function of J. J is included in the hash which is the proof of identity, so it's not needed in the encryption key. Perhaps one could claim that there is an advantage of having K be a function of J, since it hides both identities from active attackers. But if Bob doesn't know in advance who will be connecting, or at least have a very small set of candidates, then you're stuck with using the IP address as the identity, which as we said, makes it almost useless, and certainly doesn't hide identities from anyone. And if Bob *does* know in advance who will be connecting, then there's no reason for Alice to send her identity at all. And certainly Alice knows who she's talking to, since she knows what J to use. So Bob doesn't need to tell Alice his identity, either.

8. Shared Secret Key, Aggressive Mode

This protocol doesn't have the problem that the main mode shared secret protocol has, because the identities are not sent encrypted.

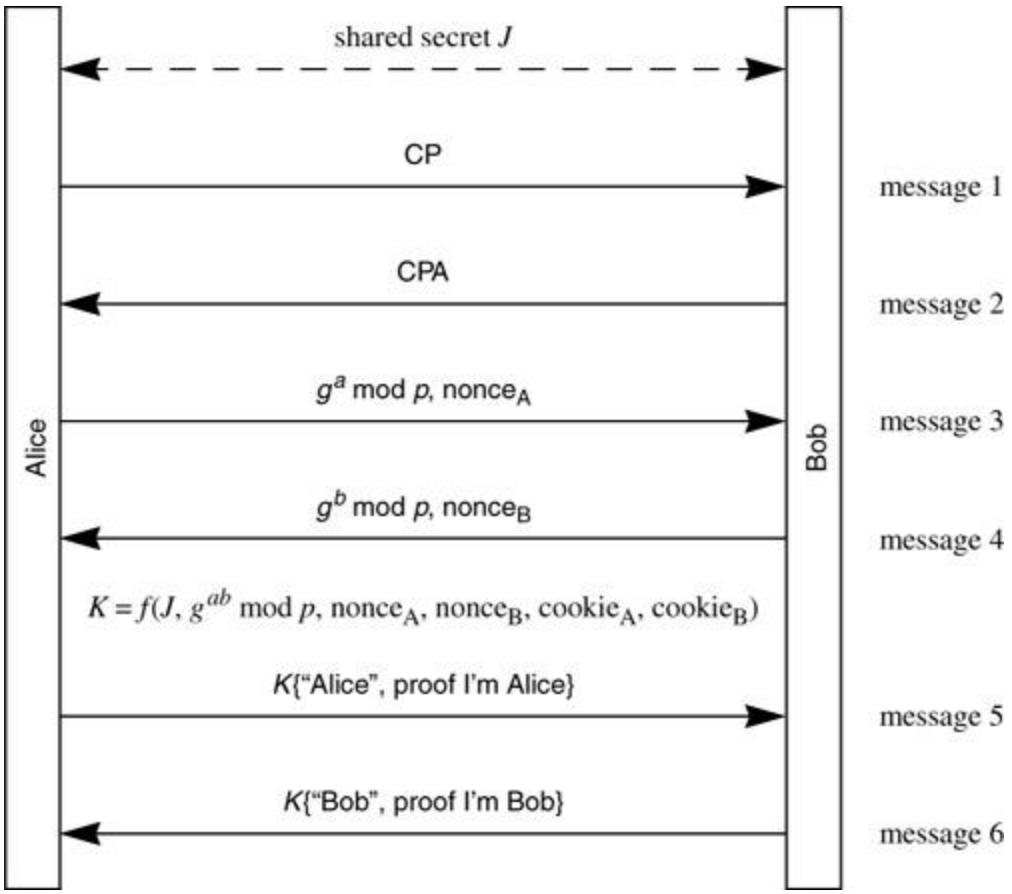


Figure: Pre-shared secret, main mode

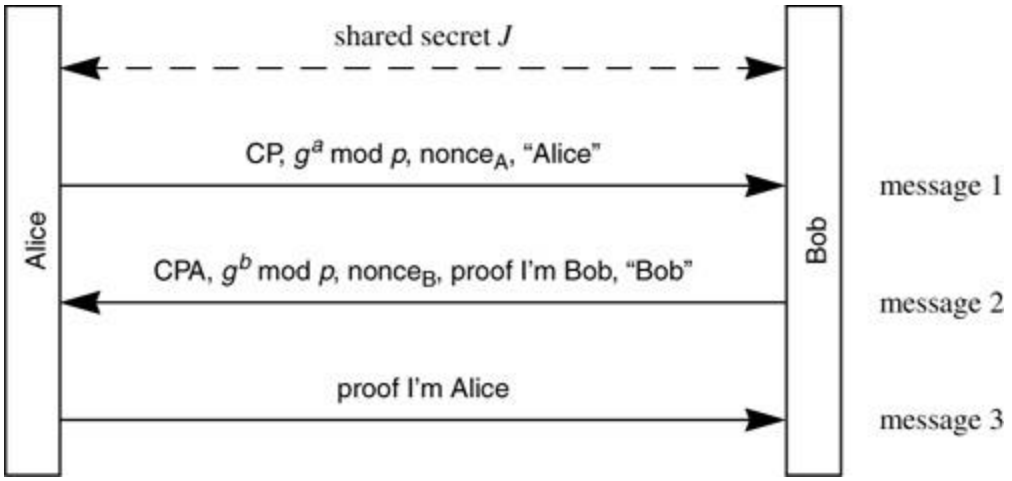
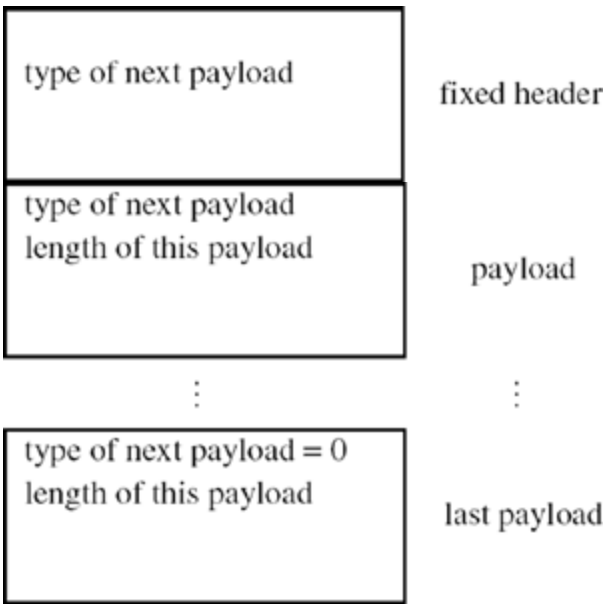


Figure: Pre-shared secret, aggressive mode

6. Explain in detail about ISAKMP/IKE encoding?

Messages have a fixed header, and then a sequence of what ISAKMP refers to as *payloads*. Similar in spirit to IPv6 extension headers, each payload starts with TYPE OF NEXT PAYLOAD and LENGTH OF THIS PAYLOAD.



The payload types are:

- 0 = end (i.e., no next payload)
- 1 = SA (security association): contains DOI and "situation", a modifier of DOI, and must include payloads 2 and 3
- 2 = P (proposal): proposed SPI, or SPI in reverse direction
- 3 = T (transform): cryptographic choices
- 4 = KE (key exchange):, the Diffie-Hellman value
- 5 = ID (endpoint identifier in phase 1, traffic selector in phase 2)
- 6 = CERT (certificate)
- 7 = CR (certificate request) (can include the name of the certifier from whom you'd like a certificate)
- 8 = hash (some sort of checksum)
- 9 = signature
- 10 = nonce
- 11 = notification
- 12 = delete (subtype of notification, meaning you are closing this SPI)
- 13 = vendor ID (can be thrown in to show what implementation you're using). To avoid dealing with a registry of vendor IDs, and allowing the field to be fixed size, this is an MD of some sort of string guaranteed to uniquely describe the vendor, such as its name and telephone number.
- 14–127 reserved for future use
- 128–255 = private use (i.e., so NSA can use it and not publish what they're using it for)

Fixed Header

All messages start with a 28-octet fixed length header.

# octets	
8	initiator's cookie
8	responder's cookie
1	next payload
1	version number (major/minor)
1	exchange type
1	flags

4	message ID	
4	message length (in units of octets)	(after encryption)

The fields are:

- initiator's cookie (8 octets)
- responder's cookie (8 octets). Note this will =0 in the first message, since it is unknown at that point
- next payload type
- version (1 octet). This is worth ranting about. The version number field is divided into two 4-bit fields. The intention is that the top nibble is the major version number and the bottom nibble is the minor version number.
- exchange type (1 octet). The values defined are:
  - 1 = base. An exchange type defined by ISAKMP but not used by IKE. This adds an extra message to aggressive mode, so that Alice (the initiator) can send her proposed parameters before sending her Diffie-Hellman value, so that the Diffie-Hellman group could also be negotiated.
  - 2 = identity protection. This is what is called "main mode" in IKE.
  - 3 = authentication only. Not used by IKE.
  - 4 = aggressive. Same as what's called "aggressive mode" in IKE.
  - 5 = informational. Not really an "exchange", since it's a single message without an acknowledgment, used to tell the other side something such as that you are refusing the connection because you don't like the version number.
  - 6–31 = reserved values by ISAKMP for assignment by IANA as new ISAKMP exchange types
  - 32–239 = to be defined within a particular DOI
  - 240–255 = for private use
- flags:
  bit 0 (LSB): encrypted—whether the fields after the header are encrypted
  bit 1: commit—A flag so badly named, and so confusingly defined in ISAKMP, that IKE wound up using the same bit and the same name for almost the opposite purpose.
  bit 2: authentication only—this means that the fields after the header are not encrypted. This bit gives no additional information over merely not setting the "encrypted" flag.
- Message ID: this is used in phase 2 in order to tie together related packets. In other words, if lots of phase 2 SAs are being negotiated simultaneously within the same phase 1 SA, this field differentiates the messages for the different SAs. The specification says that these values must be chosen randomly so that there probably won't accidentally be two phase 2 exchanges with the same message ID
- message length: Length of entire message, in units of octets.

Payload Portion of ISAKMP Messages

After the fixed header comes a set of ISAKMP "payloads", reminiscent of IPv6 "next headers". Each one starts with four octets consisting of:.

# octets	
1	type of next payload
1	reserved (unused, set to zero)
2	length of this payload

The encoding would be more intuitive to have each payload indicate the type of that payload rather than the following one, but this way works too. It's this way because it looks more like IPv6.

SA Payload

Assembly of SA payload requires great peace of mind. The SA payload for IKE includes the P (proposal) and T (transform) payloads. The encoding is extremely confusing for no good reason. The SA, P and T each look like independent payloads, but ISAKMP defines Ts as being carried inside a P, and Ps carried inside an SA payload. For example, if you have an SA payload

that includes 2 proposals, the first of which includes 4 transforms, and the second of which includes 2 transforms, you'd have the payloads SA, P, T, T, T, T, P, T, T.

Ps and Ts within the SA Payload

The P payload indicates what "protocol" you're trying to negotiate, e.g., phase 1 IKE, ESP, AH, or IP compression. For phase 1 IKE, there would only be a single P within an SA, because you're only trying to negotiate phase 1 IKE. For phase 2 IKE, there might be several Ps within an SA, because you might be making a proposal for AH only, ESP only, AH+ESP, or any of those plus IP compression.

Within a P there are a set of T payloads. Each T payload indicates a complete suite of cryptographic algorithms needed by that P. For instance, for Phase 1 IKE, you need 4 (authentication (integrity protection), hash, encryption, and Diffie-Hellman group), and optionally a lifetime. There is a default lifetime of 8 hours, so if no lifetime appears within a T, it is the same as including it explicitly with 8 hours. For an AH proposal, there's only an authentication (integrity protection) algorithm, and optionally a lifetime.

Each P payload is assigned a number by the initiator. If there are two P payloads with the same number, it implies both payloads constitute a single proposal. For instance, if Alice would like to have the SA do both ESP and AH, she would include two P payloads, both with the same proposal number, and if Bob accepts that proposal number, he is accepting an SA that will do both ESP and AH. Of the P numbers offered by Alice, Bob chooses one or refuses them all.

Each T payload also includes a number. For instance, suppose a P payload has been assigned the number 3, and has associated T payloads for that P numbered 1, 2, 3. The other side might accept proposal #3, transform suite #2.

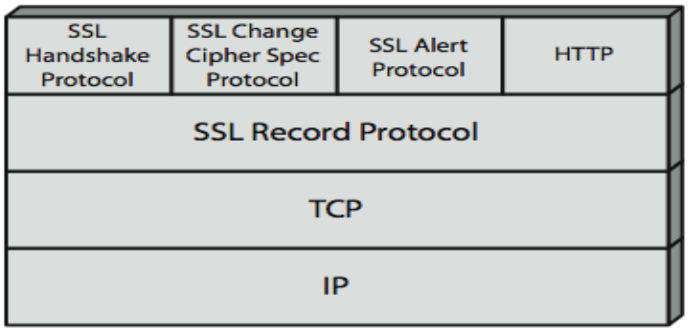
Payload Length in SA, P, and T Payloads

The PAYLOAD LENGTH in the SA payload is the length of the entire set of the payloads consisting of the SA and all Ps and Ts associated with that SA. The payload length of each P is the length of that P payload plus the T payloads that follow. The payload length of each T payload is actually the length of that T payload.

7. Describe the SSL Architecture in detail.

SSL Architecture

- SSL session
  - an association between client & server
  - created by the Handshake Protocol
  - define a set of cryptographic parameters
  - may be shared by multiple SSL connections
- SSL connection
  - a transient, peer-to-peer, communications link
  - associated with 1 SSL session



SSL Record Protocol Operation

- confidentiality
  - using symmetric encryption with a shared secret key defined by Handshake Protocol

    - IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
    - message is compressed before encryption
- message integrity
    - using a MAC (Message Authentication Code) created using a shared secret key and a short message

The first step is **fragmentation**. Each upper-layer message is fragmented into blocks of 214 bytes (16384 bytes) or less. Next, **compression** is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes.1In SSLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm is null. The next step in processing is to compute a **message authentication code** over the compressed data. For this purpose, a shared secret key is used. Next, the compressed message plus the MAC are **encrypted** using symmetricencryption. Encryption may not increase the content length by more than 1024

SSL Handshake Protocol

- allows server & client to:
    - authenticate each other
    - to negotiate encryption & MAC algorithms
    - to negotiate cryptographic keys to be used
- comprises a series of messages in phases
    - Establish Security Capabilities
    - Server Authentication and Key Exchange
    - Client Authentication and Key Exchange
    - Finish

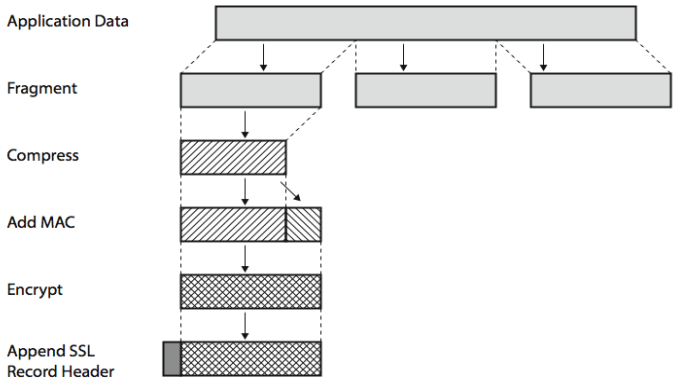


Fig: SSL Record Protocol

The final step of SSL Record Protocol processing is to prepare a header consisting of the following fields:

• **Content Type (8 bits):** The higher-layer protocol used to process the enclosed fragment.
• **Major Version (8 bits):** Indicates major version of SSL in use. For SSLv3, the value is 3.
• **Minor Version (8 bits):** Indicates minor version in use. For SSLv3, the value is 0.
• Compressed Length (16 bits):

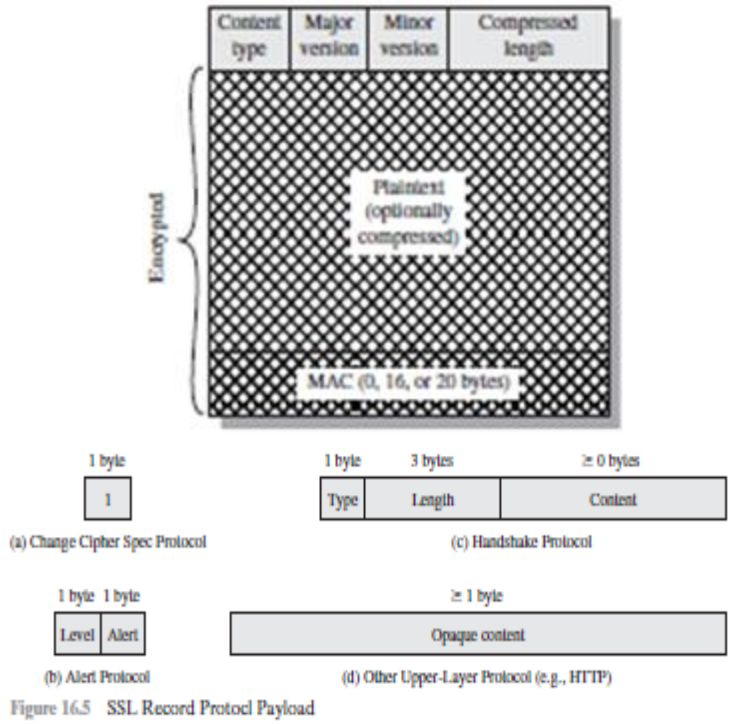


Figure 16.5 SSL Record Protocol Payload

Change Cipher Spec Protocol

The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol, and it is the simplest. This protocol consists of a single message which consists of a single byte with the value 1.The sole purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.

Alert Protocol

The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state.Each message in this protocol consists of two bytes .The first byte takes the value warning (1) or fatal (2) to convey the severity of the message. If the level is fatal, SSL immediately terminates the connection. Other connections on the same session may continue, but no new connections on this session may be established. The second byte contains a code that indicates the specific alert. First, we list those alerts that are always fatal (definitions from the SSL specification):

• **unexpected_message:** An inappropriate message was received.
• **bad_record_mac:** An incorrect MAC was received.
• **decompression_failure:** The decompression function received improper input (e.g., unable to decompress or decompress to greater than maximum allowable length).
• **handshake_failure:** Sender was unable to negotiate an acceptable set of security parameters given the options available.
• **illegal_parameter:** A field in a handshake message was out of range or inconsistent with other fields.

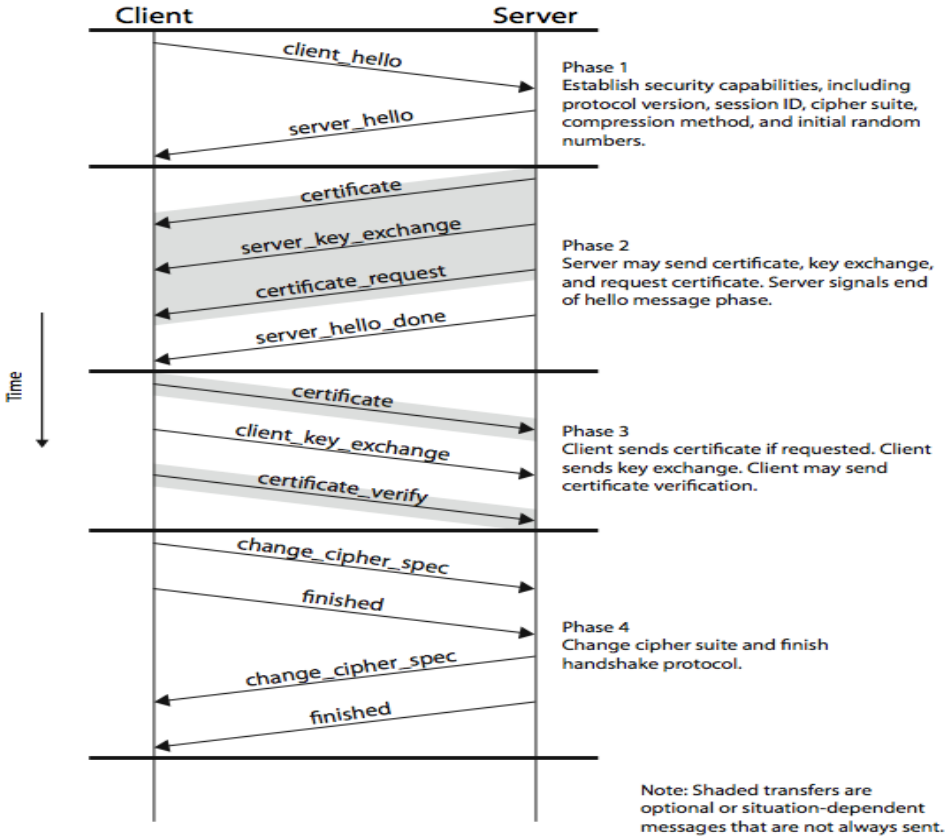
The remaining alerts are the following.

• **close_notify:** Notifies the recipient that the sender will not send any more messages on this connection. Each party is required to send a **close_notify** alert before closing the write side of a connection.
• **no_certificate:** May be sent in response to a certificate request if no appropriate certificate is available.
• **bad_certificate:** A received certificate was corrupt (e.g., contained a signature that did not verify).
• **unsupported_certificate:** The type of the received certificate is not supported.
• **certificate_revoked:** A certificate has been revoked by its signer.

• certificate_expired: **A certificate has expired.**

The Handshake Protocol consists of a series of messages exchanged by client and server. Each message has three fields:
• **Type (1 byte):** Indicates one of 10 messages.
• **Length (3 bytes):** The length of the message in bytes.
• **Content (bytes):** The parameters associated with this message;

SSL Handshake Protocol



8. Write about SSL/TLS Protocol?

Using the reliable octet stream service provided by TCP, SSL/TLS partitions this octet stream into records, with headers and cryptographic protection, to provide a reliable, encrypted, and integrity-protected stream of octets to the application. There are four types of records: user data, handshake messages, alerts (error messages or notification of connection closure), and change cipher spec (which should be a handshake message, but they chose to make it a separate record type).

In the basic protocol the client (Alice) initiates contact with the server (Bob). Then Bob sends Alice his certificate. Alice verifies the certificate, extracts Bob's public key, picks a random number *S* from which the session keys will be computed, and sends *S* to Bob, encrypted with Bob's public key. Then the remainder of the session is encrypted and integrity-protected with those session keys. (There are actually six secrets computed when encryption algorithms that require IVs are used—for each direction: integrity protection key, encryption key, and IV.) First we'll present a simplified form of the protocol, then discuss various issues in the full protocol, and finally discuss the details.

- **Message 1.** Alice says she would like to talk (but doesn't identify herself), and gives a list of cryptographic algorithms she supports, along with a random number R_{Alice}, that will be combined with the *S* in message 3 to form the various keys.

- **Message 2.** Bob sends Alice his certificate, a random number R_{Bob} that will also contribute to the keys, and responds with one of the ciphers Alice listed in message 1 that Bob also supports.
- **Message 3.** Alice chooses a random number *S* (known as the **pre-master secret**) and sends it, encrypted with Bob's public key. She also sends a hash of the **master secret** *K* and the hand-shake messages, both to prove she knows the key and to ensure that tampering of the handshake messages would be detected. The hash is an (unnecessarily) complex function based on one of the early versions of HMAC. To ensure that the keyed hash Alice sends is different from the keyed hash Bob sends, each side includes a constant ASCII string in the hash. The initiator constant is CLNT in SSLv3 and client finished in TLS. The constant Bob will hash into the stew is SRVR in SSLv3 and server finished in TLS. Surprisingly (and unnecessarily), the keyed hash is sent encrypted and integrity-protected. The keys used for encrypting the keyed hash, like the rest of the data in the session will be, are derived from hashing *K*, R_{Alice}, and R_{Bob}. The keys used for transmission are known as *write* keys, and the keys used for receipt are known as *read* keys. So, for instance, Bob's write-encryption key is Alice's read-encryption key. The keys are encryption, integrity, and IV in each direction, so there are six keys derived. (In SSLv2 there are only two session keys, one in each direction, each used both for integrity protection and encryption.)

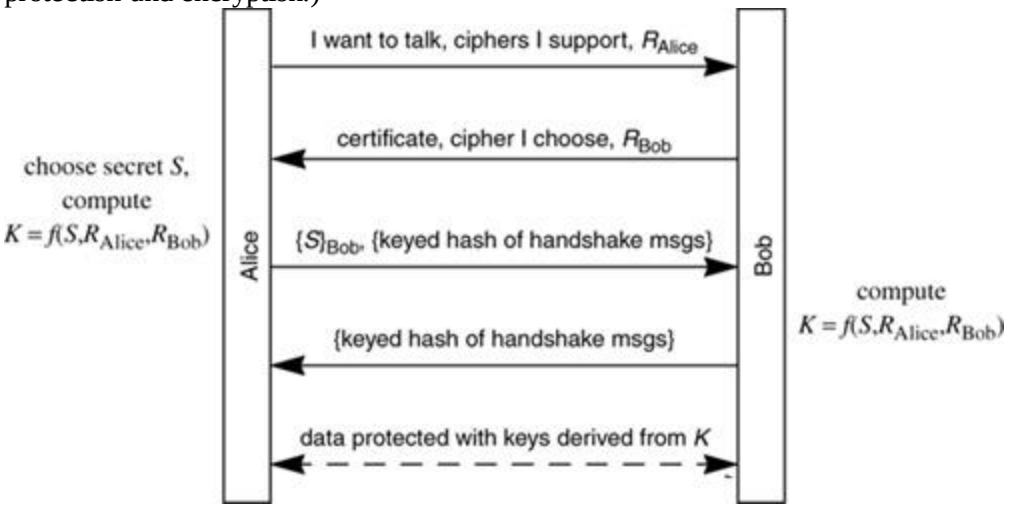


Figure: SSLv3/TLS

- **Message 4.** Bob proves he knows the session keys, and ensures that the early messages arrived intact, by sending a keyed hash of all the handshake messages, encrypted with his write-encryption key, and integrity-protected with his write-integrity key. Since the session keys are derived from *S* this proves he knows Bob's private key, because he needed it in order to extract *S*.

At this point, Alice has authenticated Bob, but Bob has no idea to whom he's talking. As deployed today, authentication is seldom mutual—the client authenticates the server but the server does not authenticate the client. In theory SSL/TLS could be used for mutual authentication and the protocol allows optional authentication of the client if the client has a certificate. But in the most common case today, if the application on the server wishes to authenticate the user, it's usually done by having the user Alice send her name and password to the server Bob, cryptographically protected with the session keys.

9. Explain about packet encoding in SSL/TLS.

The packet encodings in the SSL/TLS specifications are expressed in a pseudo-ASN.1 syntax that is very unpleasant to read and makes it difficult to imagine what the packets actually look like. Hopefully you will appreciate our translating them into pictures here. The problem with the ugly syntax in the spec is that not only does it give the reader a headache, but it resulted in weird formats, for instance redundant fields, and implementation bugs

Since SSL/TLS runs on top of TCP, SSL/TLS can send an arbitrary-length chunk. TCP will handle breaking it up into packets and reassembling it. SSL/TLS has two layers of chunking itself, which operate somewhat independently. The unit of cryptographic protection is the **record**, which contains a header and a body, where the body is cryptographically protected if the protocol is far enough along. Records are limited to 2^{14} octets by the specification, but there are buggy implementations that send records of up to $2^{16}-1$ octets so a conservative implementation should be prepared to accept records up to that size. The parts of the handshake are divided into **messages**, which also contain a header and a body. In principle, a single message could be packaged in multiple records or multiple messages could be placed in a single record. In practice, most implementations combine the two layers and put one message in each record.

It would have been simpler (and more efficient) for each message to be its own record type, and to have only a single header on each handshake message. But instead the designers chose to have four record types, with most of the handshake messages being enclosed in records of type handshake. A handshake record can contain multiple messages in it, whereas the other records are really free-standing messages. If the server is going to send a *ServerHello*, a *Certificate*, and a *ServerHelloDone* (all of which are handshake messages), these could be sent as three records, or lumped together into a single record, or two of them could be in one record and the third in another record.

The four record types are:

- 20=ChangeCipherSpec. This is really part of the handshake, and it's rather strange to have it be its own record type. The reason they did this is that after the *ChangeCipherSpec* message the data is encrypted and integrity-protected with the key just agreed upon. It would not be possible for only part of a record to be encrypted, so *ChangeCipherSpec* had to be the last thing in a record. They could have had this be a handshake message, mandating that it be the final message in the record. But instead they said that it would be its own record type.
- 21=alert (any sort of notification)
- 22=handshake (any of many messages which is part of the handshake)
- 23=application_data (encrypted and integrity-protected data sent after the handshake is complete)

The record header, which is never encrypted, is:

# octets	
1	record type
2	version number
2	length

Following the record header is the contents of that record, which might be encrypted and integrity-protected. In the case of the handshake record, the record contains handshake messages, each message starting with a 1-octet type and a 3-octet length. It's somewhat mysterious why they would choose a 3-octet length field for the messages and a 2-octet length field for the record, but in theory if there were a message that was bigger than 2^{14} octets, it could be split into several records. So the 2-octet length of the record length field does not constrain messages to be smaller than 2^{16} octets. All multi-octet integers are sent in big-endian order (also known as network byte order).

Encrypted Records

Records sent after a *ChangeCipherSpec* record are cryptographically protected with the negotiated cipher suite. Integrity protection is provided using HMAC based on MD5 or SHA-1. Encryption uses any of a variety of encryption algorithms, all of which are block ciphers in CBC mode except RC4, which is a stream cipher.

HMAC takes two arguments, a key and data. The key used is the session integrity key for that direction. The data on which the HMAC is based consists of a 64-bit sequence number followed by the record header followed by the data. The sequence number is not actually transmitted. It only is used in the calculation of the HMAC. It protects against ordering and replay, but does not need to be explicitly transmitted because SSL runs over a reliable protocol (TCP). If

TCP failed somehow and lost, reordered, or duplicated data, the two sides would disagree about the sequence number, the integrity check would fail, and this would result in a connection failure.

If a block cipher is to be used for encryption, the DATA | HMAC is padded to a multiple of the block size. Then DATA | HMAC | padding is encrypted. If a block cipher is used, the initial IV is computed along with the key and the final block of each record is used as the IV for the next.

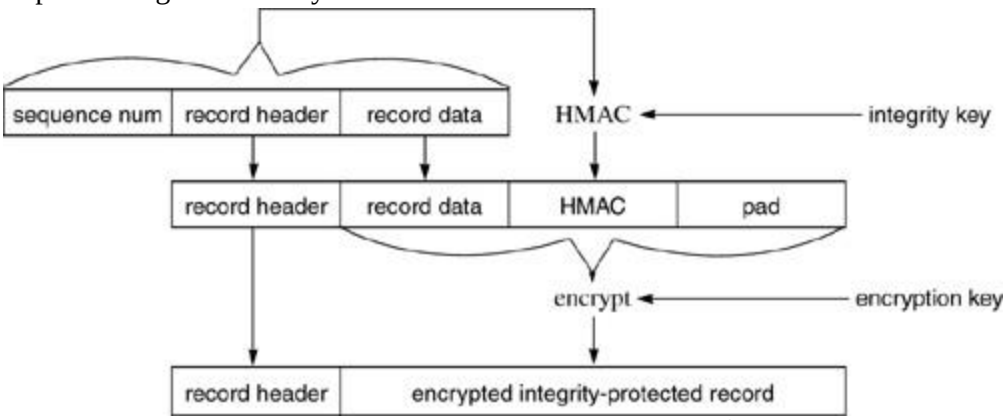


Figure: Cryptographically protected record format

For TLS, HMAC is computed as specified in RFC 2104. For SSLv3, the MAC is computed in a similar fashion, but is different both because the HMAC design was still evolving and they got an early draft and because a typographical error resulted in 40 octets of padding in a spot where there should have been 44. Neither of these differences is likely to affect the security, but they do complicate the documentation.

To summarize, what is transmitted is the record header, followed by an encrypted blob which, when decrypted, consists of the data followed by the HMAC, followed by padding including the padding length.

Handshake Messages

A handshake record (type 22) contains handshake messages. The specification allows someone to break a handshake message into pieces, and send each piece in a separate handshake record. It is also permissible for the handshake record to contain multiple handshake messages. For instance, the server might send a *ServerHello*, a *Certificate*, a *CertificateRequest*, and a *ServerHelloDone*, all within the same handshake record. The handshake messages are *ClientHello*, *ServerHello*, *ClientKeyExchange*, *Certificate*, *ServerHelloDone*, *HandshakeFinished*, *CertificateRequest*, *CertificateVerify*, and *ServerKeyExchange*.

ClientHello

# octets	
1	type=1
3	length
2	version number
32	random (R_{Alice})
1	length of session_id (or 0 if field absent)
Variable	session_id
2	length of cipher suite list (in octets)
variable	sequence of cipher suites, each a 2-octet type
1	length of compression list (in octets)
variable	sequence of compression methods, each a single octet

This message contains the optional session_id (to allow a session to be resumed), R_{Alice}, and Alice's list of cipher suites and compression methods.

ServerHello

# octets	
1	type=2
3	Length
2	version number
32	random (R_{Bob})
1	length of session_id (or 0 if field absent)
variable	session_id
2	chosen cipher
1	chosen compression method

This message contains R_{Bob}, an optional session_id (to allow the session to be resumable), and Bob's choice of cipher and compression method.

ServerHelloDone

# octets	
1	type=14
3	length=0

This indicates that the server is finished sending its handshake messages. It is the final piece of message 2 in Protocol. It just has a type octet (=14) and a length field (3 octets of 0).

ClientKeyExchange

# octets	
1	type=16
3	Length
2	length (missing in SSLv3, present in TLS)
variable	encrypted pre-master secret

This is the message in which Alice sends the pre-master secret encrypted with the server's public key. Note the extra length field. It serves no purpose with RSA or Diffie-Hellman. It was in the SSLv3 specification, but not in any implementations. TLS copied the SSLv3 spec, so left the extra length field in. As we said earlier, it is an artifact of the packet specification language they invented, and so difficult to read and visualize that it resulted not only in extra fields but implementation bugs.

Redundant length fields are a problem. At best it's extra work to set and look at them, and extra octets on the wire. And what are you supposed to do if they disagree? Also, note that the second length field is only 2 octets long and the first one is 3 octets long. If the length never needs to be larger than can be expressed in two octets, why is the other length field 3 octets?

Although the second length field is redundant in RSA and Diffie-Hellman, there are key exchange algorithms such as FORTEZZA that require multiple variable length fields to express the encrypted pre-master secret. Also, future algorithms might require more than 2^{16} octets of information. The information following the 3-octet length field is algorithm dependent, so they could have left out the 2nd length field for RSA and Diffie-Hellman, but had the data for FORTEZZA consist of (length, value) pairs.

ServerKeyExchange

# octets	
1	type=12
3	length
2	length of modulus
Variable	modulus
2	length of exponent

variable	exponent
2	length of signature
variable	signature

This message is used in SSLv3 and TLS for export approval, in which the server signs a (short) ephemeral public key, signed with its long-term key. Instead of sending an X.509 certificate, it is just a signature on the new key. It is also used when the server's long-term public key can only be used for signing, for instance when the server's key is a DSS key.

CertificateRequest

# octets	
1	type=13
3	length
1	length of key type list
Variable	list of types of keys (each one octet long)
2	length of CA name list
2	length of 1st CA name
variable	1st CA name
2	length of 2nd CA name
variable	2nd CA name
2	length of 3rd CA name
variable	3rd CA name
	...
2	length of last CA name
variable	last CA name

This message is sent by the server to request that the client send a certificate and authenticate. There are several defined "key types", which not only specify the kind of key, but the kind of authentication method the user will use to authenticate. The most commonly used is type 1=RSA signing.

Certificate

# octets	
1	type=11
3	length
3	length (unnecessary field)
3	length of first certificate
Variable	first certificate
Variable	more (3-octet length, certificate) pairs, if more than one certificate is being sent

This message contains one or more certificates. Note this also contains a redundant length field. The second length field will always be 3 less than the first length field (since the first length field includes the second length field in the number of octets it is counting).

CertificateVerify

# octets	
1	type=15
3	length
2	length of signature
Variable	signature

This message is sent by the client to prove it knows its private key. The signature (in the case of RSA, which is really all that's deployed), is a modified PKCS #1 RSA-signed quantity consisting of both the MD5 of the handshake messages and the SHA of the handshake messages. The modification to PKCS #1 is that the ASN.1 encoded digest type is omitted and the MD5 and the SHA digests are concatenated.

HandshakeFinished

# octets	
1	type=20
3	length=36 or 12
36 or 12	digest

This message ensures integrity of the exchange as well as proving knowledge of the key. In SSLv3 the digest, strangely, is a 36-octet quantity consisting of a concatenation of a 16-octet keyed MD5 hash and 20-octet keyed SHA hash of information including the handshake messages. In TLS the digest is 12 octets computed with a complex combination of MD5 and SHA.

ChangeCipherSpec

# octets	
1	record type=20
2	version number
2	length (set to 1)
1	ChangeCipherSpecType (set to 1)

This isn't a handshake message, but is instead its own record type. It indicates that all records following this will be protected with the ciphers agreed upon as of this message. The final field (CHANGECIPHERSPECTYPE) is mysterious. Only one value (1) has been defined, which means *change to the cipher suite we've agreed to now*. It's unclear when you would want to say anything other than that. But, if someone thinks of a new thing to say, they can define a new value for that field.

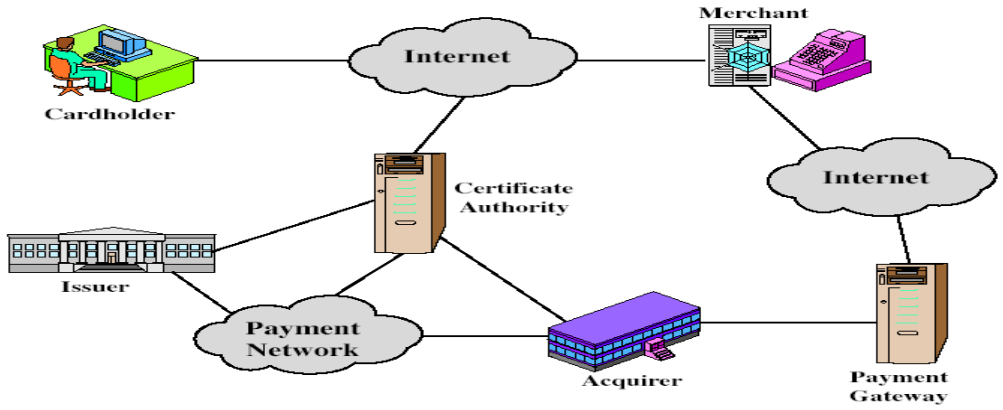
Alerts

An alert is sent to advise the other side of some condition. Most alerts are error messages, with a severity level of either 1=warning, or 2=fatal. The one defined alert which is not an error indication is the *closure* alert, in which one side notifies the other that it has no more data to send. This was added in SSLv3 because of a security vulnerability in SSLv2 known as the truncation attack. SSLv2 assumed the other side was finished sending data if the TCP connection closed. However, because TCP commands are not cryptographically protected, there is no way to know whether the TCP connection closed because the other side closed the TCP connection when it finished sending all its data, or an attacker sent the TCP command to close the connection.

10. Explain in detail about SET.

SET –Requirements

1. Provide confidentiality of payment and ordering information
2. Ensure the integrity of all transmitted data
3. Provide authentication that a cardholder is a legitimate user of a credit card account
4. Provide authentication that a merchant can accept credit card transaction through its relationship with a financial institution
5. Ensure the use of the best security practices
6. Create a protocol that neither depends on transport security mechanism nor prevents their use
7. Facilitate and encourage interoperability among software and network providers



Dual Signature

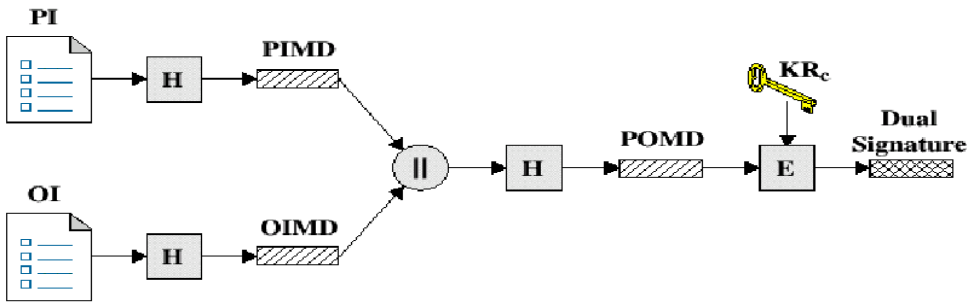
The operation for dual signature is as follows:

Take the hash (SHA-1) of the payment and order information.

These two hash values are concatenated [H(PI) || H(OI)] and then the result is hashed.

Customer encrypts the final hash with a private key creating the dual signature.

$$DS = E_{KRC} [H(H(PI) || H(OI))]$$



SET –Key features

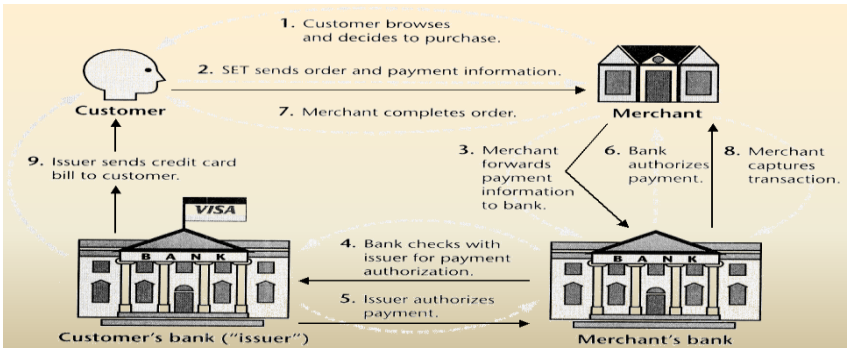
- Confidentiality of information
- Integrity of data
- Card holder account authentication
- Merchant authentication

SET participants

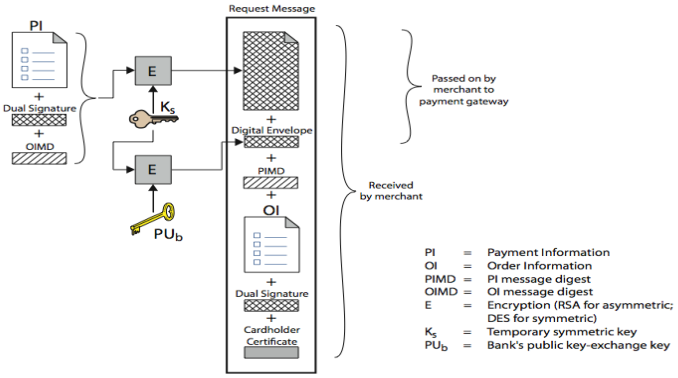
- Card holder
- Merchant
- Issuer
- Acquirer
- Payment gateway
- Certification Authority(CA)

SET Transaction

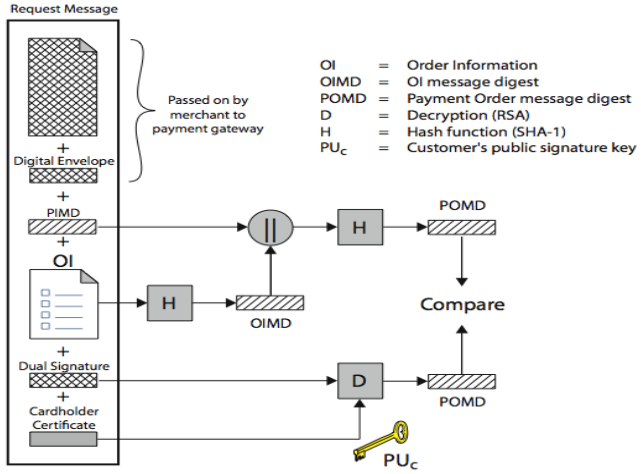
1. customer opens account
2. customer receives a certificate
3. merchants have their own certificates
4. customer places an order
5. merchant is verified
6. order and payment are sent
7. merchant requests payment authorization
8. merchant confirms order
9. merchant provides goods or service
10. merchant requests payment



Purchase Request – Customer



Purchase Request – Merchant



11. Explain PGP message generation and reception

The sending PGP entity performs the following steps

1. Signing the message:

a. PGP retrieves the sender's private key from the private-key ring using your userid as an index. If your userid was not provided in the command, the first private key on the ring is retrieved.

b. PGP prompts the user for the passphrase to recover the unencrypted private key.

c. The signature component of the message is constructed.

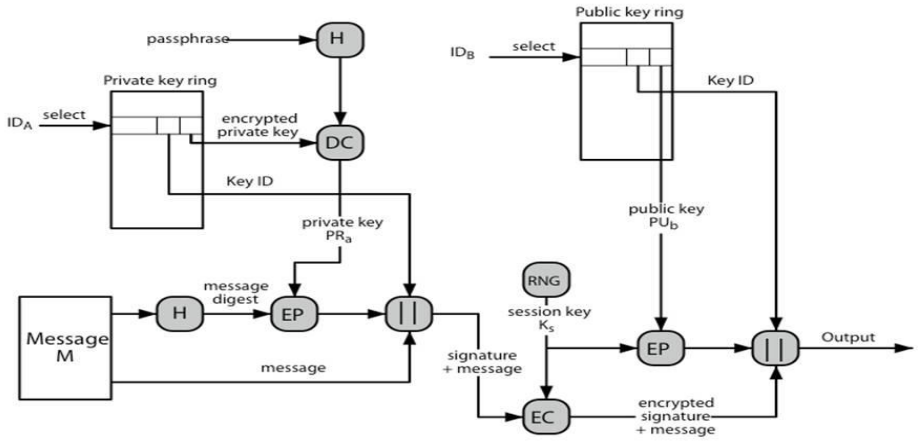
2. Encrypting the message:

a. PGP generates a session key and encrypts the message.

b. PGP retrieves the recipient's public key from the public-key ring using her_userid as an index.

c. The session key component of the message is constructed.

PGP Message Generation



The receiving PGP entity performs the following steps

1. Decrypting the message:

a. PGP retrieves the receiver's private key from the private-key ring using the Key ID field in the session key component of the message as an index.

b. PGP prompts the user for the passphrase to recover the unencrypted private key.

c. PGP then recovers the session key and decrypts the message

2. Authenticating the message:

a. PGP retrieves the sender's public key from the public-key ring using the Key ID field in the signature key component of the message as an index.

b. PGP recovers the transmitted message digest.

c. PGP computes the message digest for the received message and compares it to the transmitted message digest to authenticate

PGP Message Reception