1. Explain the OSI Architecture.

OSI SECURITY ARCHITECTURE

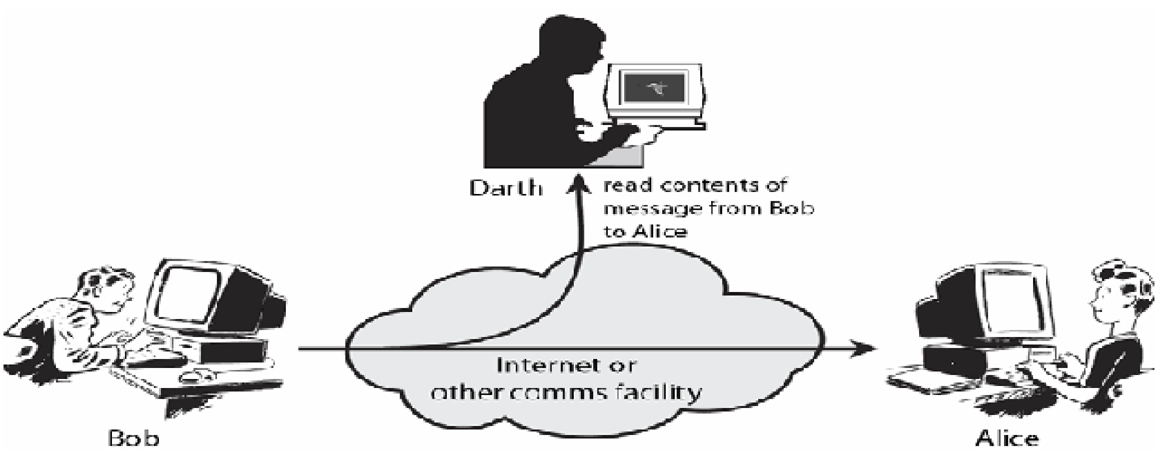
- Itu-t x.800 “security architecture for osi”
- Defines a systematic way of defining and providing security requirements
- For us it provides a useful, if abstract, overview of concepts we will study

Aspects of security

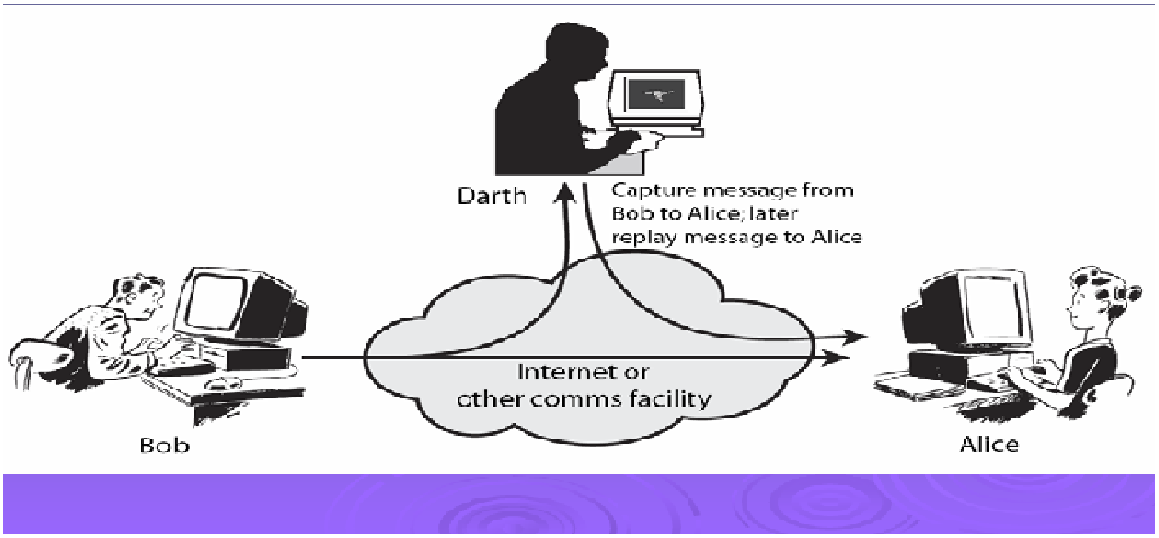
- Consider 3 aspects of information security:
  - **Security attack**
  - **Security mechanism**
  - **Security service**

Security attack

- Any action that compromises the security of information owned by an organization
- Information security is about how to prevent attacks, or failing that, to detect attacks on information-based systems
- Often *threat* & *attack* used to mean same thing
- Have a wide range of attacks
- Can focus of generic types of attacks
  - Passive
  - Active



Active attacks



SECURITY SERVICE

- Enhance security of data processing systems and information transfers of an organization
- Intended to counter security attacks
- Using one or more security mechanisms
- Often replicates functions normally associated with physical documents
  - Which, for example, have signatures, dates; need protection from disclosure, tampering, or destruction; be notarized or witnessed; be recorded or licensed

- X.800:

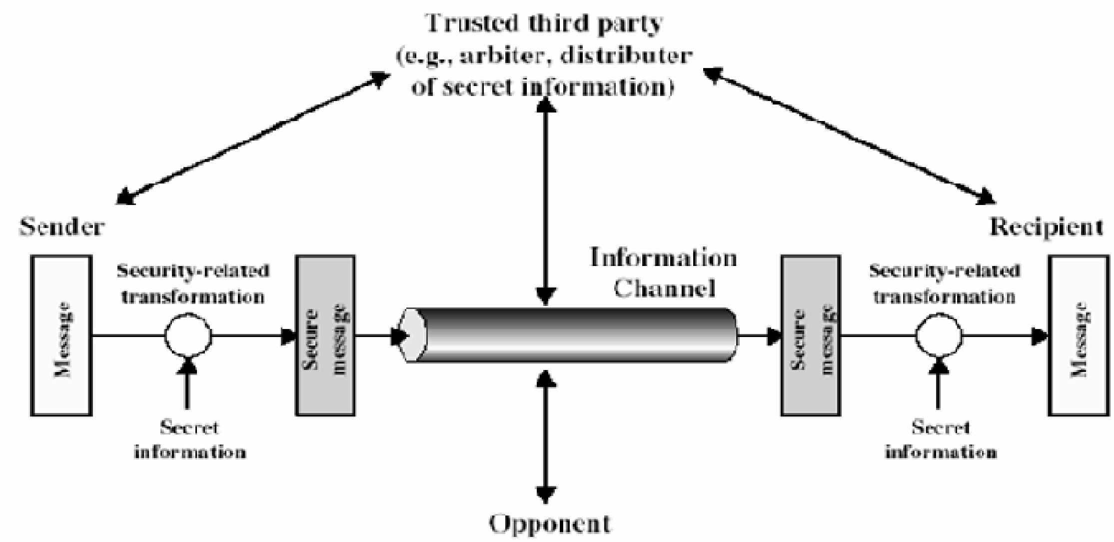
"a service provided by a protocol layer of communicating open systems, which ensures adequate security of the systems or of data transfers"

- Rfc 2828:

"a processing or communication service provided by a system to give a specific kind of protection to system resources"

- **Authentication** - assurance that the communicating entity is the one claimed
- **Access control** - prevention of the unauthorized use of a resource
- **Data confidentiality** –protection of data from unauthorized disclosure
- **Data integrity** - assurance that data received is as sent by an authorized entity
- **Non-repudiation** - protection against denial by one of the parties in a communication

MODEL FOR NETWORK SECURITY



MODEL FOR NETWORK SECURITY

- Using this model requires us to:
  1. Design a suitable algorithm for the security transformation
  2. Generate the secret information (keys) used by the algorithm
  3. Develop methods to distribute and share the secret information
  4. Specify a protocol enabling the principals to use the transformation and secret information for a security service

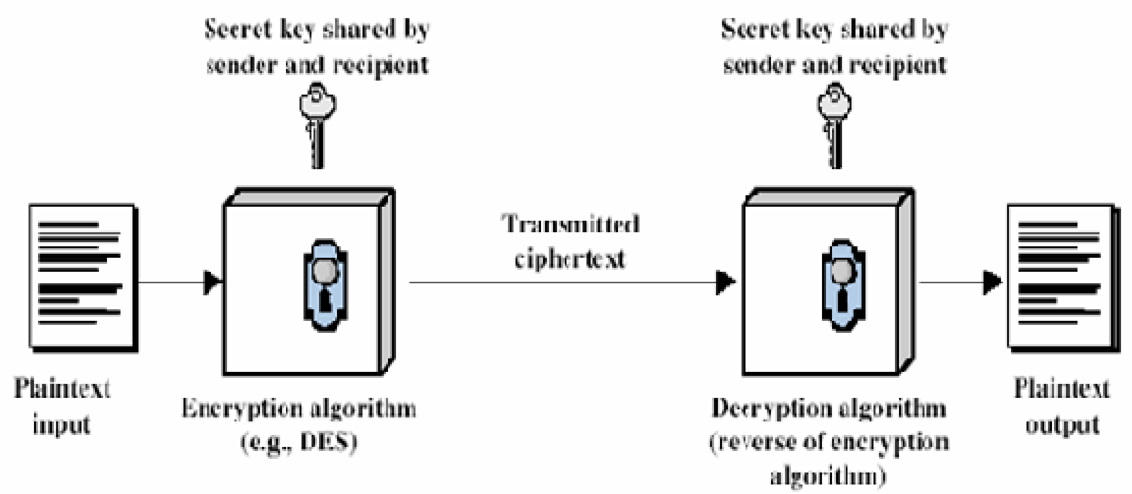
SYMMETRIC ENCRYPTION

- Sender and recipient share a common key
- All classical encryption algorithms are private-key
- Was only type prior to invention of public-key in 1970's
- And by far most widely used

SOME BASIC TERMINOLOGY

- **Plaintext** - original message
- **Ciphertext** - coded message
- **Cipher** - algorithm for transforming plaintext to ciphertext
- **Key** - info used in cipher known only to sender/receiver
- **Encipher (encrypt)** - converting plaintext to ciphertext
- **Decipher (decrypt)** - recovering ciphertext from plaintext
- **Cryptography** - study of encryption principles/methods
- **Cryptanalysis (codebreaking)** - study of principles/ methods of deciphering ciphertext *without* knowing key
- **Cryptology** - field of both cryptography and cryptanalysis

Symmetric cipher model



Requirements

- Two requirements for secure use of symmetric encryption:
    - A strong encryption algorithm
    - A secret key known only to sender / receiver
- Mathematically have:
  $y = \mathrm{e}k(x)$
  $x = \mathrm{d}k(y)$
- Assume encryption algorithm is known
- Implies a secure channel to distribute key

CRYPTOGRAPHY

- Characterize cryptographic system by:
    - Type of encryption operations used
        - Substitution / transposition / product
    - Number of keys used
        - Single-key or private / two-key or public
    - Way in which plaintext is processed
        - Block / stream

CRYPTANALYSIS

- Objective to recover key not just message
- General approaches:
    - Cryptanalytic attack
    - Brute-force attack

CRYPTANALYTIC ATTACKS

- **Ciphertext only**
    - Only know algorithm & ciphertext, is statistical, know or can identify plaintext

- **Known plaintext**
  - Know/suspect plaintext & ciphertext
- **Chosen plaintext**
  - Select plaintext and obtain ciphertext
- **Chosen ciphertext**
  - Select ciphertext and obtain plaintext
- **Chosen text**
  - Select plaintext or ciphertext to en/decrypt
- **Unconditional security**
  - No matter how much computer power or time is available, the cipher cannot be broken since the ciphertext provides insufficient information to uniquely determine the corresponding plaintext
- **Computational security**
  - Given limited computing resources (eg time needed for calculations is greater than age of universe), the cipher cannot be broken

BRUTE FORCE SEARCH

- Always possible to simply try every key
- Most basic attack, proportional to key size
- Assume either know / recognise plaintext

2. Explain Classical Encryption Techniques.

CLASSICAL SUBSTITUTION CIPHERS

- Where letters of plaintext are replaced by other letters or by numbers or symbols
- Or if plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns

CAESAR CIPHER

- Earliest known substitution cipher
- By julius caesar
- First attested use in military affairs
- Replaces each letter by 3rd letter on
- Example:

Meet me after the toga party
Phhw ph diwhu wkh wrjd sduwb

- Can define transformation as:

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

- Mathematically give each letter a number

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
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

- Then have caesar cipher as:

$C = e(p) = (p + k) \bmod (26)$

$P = d(c) = (c - k) \bmod (26)$

CRYPTANALYSIS OF CAESAR CIPHER

- Only have 26 possible ciphers
  - A maps to a,b,..z
- Could simply try each in turn
- A **brute force search**
- Given ciphertext, just try all shifts of letters
- Do need to recognize when have plaintext
- Eg. Break ciphertext "gcua vq dtgcm"

MONOALPHABETIC CIPHER

- Rather than just shifting the alphabet
- Could shuffle (jumble) the letters arbitrarily
- Each plaintext letter maps to a different random ciphertext letter
- Hence key is 26 letters long

Plain: abcdefghijklmnopqrstuvwxyz
Cipher: dkvqfibjwpescxhtmyauolrgzn

Plaintext: ifwewishtoreplaceletters
Ciphertext: wirfrwajuhyftsdvfsfuufya

LANGUAGE REDUNDANCY AND CRYPTANALYSIS

- Human languages are **redundant**
- Eg "th lrd s m shphrd shll nt wnt"
- Letters are not equally commonly used
- In english e is by far the most common letter
  - Followed by t,r,n,i,o,a,s
- Other letters like z,j,k,q,x are fairly rare
- Have tables of single, double & triple letter frequencies for various languages

USE IN CRYPTANALYSIS

- Key concept - monoalphabetic substitution ciphers do not change relative letter frequencies
- Discovered by arabian scientists in 9th century
- Calculate letter frequencies for ciphertext
- Compare counts/plots against known values
- If caesar cipher look for common peaks/troughs
  - Peaks at: a-e-i triple, no pair, rst triple
  - Troughs at: jk, x-z
- For monoalphabetic must identify each letter
  - Tables of common double/triple letters help

PLAYFAIR CIPHER

- Not even the large number of keys in a monoalphabetic cipher provides security
- One approach to improving security was to encrypt multiple letters
- The **playfair cipher** is an example
- Invented by charles wheatstone in 1854, but named after his friend baron playfair

ENCRYPTING AND DECRYPTING

- Plaintext is encrypted two letters at a time
    1. If a pair is a repeated letter, insert filler like 'x'
    2. If both letters fall in the same row, replace each with letter to right (wrapping back to start from end)
    3. If both letters fall in the same column, replace each with the letter below it (again wrapping to top from bottom)
    4. Otherwise each letter is replaced by the letter in the same row and in the column of the other letter of the pair

POLYALPHABETIC CIPHERS

- **POLYALPHABETIC SUBSTITUTION CIPHERS**
- Improve security using multiple cipher alphabets
- Make cryptanalysis harder with more alphabets to guess and flatter frequency distribution
- Use a key to select which alphabet is used for each letter of the message
- Use each alphabet in turn
- Repeat from start after end of key is reached

VIGENÈRE CIPHER

- Simplest polyalphabetic substitution cipher
- Effectively multiple caesar ciphers
- Key is multiple letters long k = k1 k2 ... Kd
- Ith letter specifies ith alphabet to use
- Use each alphabet in turn
- Repeat from start after d letters in message
- Decryption simply works in reverse

EXAMPLE OF VIGENÈRE CIPHER

- Write the plaintext out
- Write the keyword repeated above it
- Use each key letter as a caesar cipher key
- Encrypt the corresponding plaintext letter
- Eg using keyword *deceptive*

Key: deceptivedeceptivedeceptive
Plaintext: wearediscoveredsaveyourself
Ciphertext:zicvtwqngrzgvtwavzhcqyglmgj

AUTOKEY CIPHER

- Ideally want a key as long as the message
- Vigenère proposed the **autokey** cipher
- With keyword is prefixed to message as key
- Knowing keyword can recover the first few letters
- Use these in turn on the rest of the message
- But still have frequency characteristics to attack
- Eg. Given key *deceptive*

Key: deceptivewearediscoveredsav
Plaintext: wearediscoveredsaveyourself
Ciphertext:zicvtwqngkzeiigasxstslvvwla

ONE-TIME PAD

- If a truly random key as long as the message is used, the cipher will be secure
- Called a one-time pad
- Is unbreakable since cipher text bears no statistical relationship to the plaintext
- Since for **any plaintext** & **any cipher text** there exists a key mapping one to other
- Can only use the key **once** though
- Problems in generation & safe distribution of key

TRANSPOSITION CIPHERS

- Now consider classical **transposition** or **permutation** ciphers
- These hide the message by rearranging the letter order
- Without altering the actual letters used
- Can recognise these since have the same frequency distribution as the original text

RAIL FENCE CIPHER

- Write message letters out diagonally over a number of rows
- Then read off cipher row by row
- Eg. Write message out as:

M e m a t r h t g p r y
e t e f e t e o a a t

- Giving ciphertext

Mematrhtgpryetefeteoaat

ROW TRANSPOSITION CIPHERS

- A more complex transposition
- Write letters of message out in rows over a specified number of columns
- Then reorder the columns according to some key before reading off the rows

Key: 4 3 1 2 5 6 7
Plaintext: a t t a c k p
o s t p o n e

d u n t i l t
w o a m x y z

Ciphertext: ttnaaptmtsuoaodwcoixknlypetz

STEGANOGRAPHY

- An alternative to encryption
- Hides existence of message
  - Using only a subset of letters/words in a longer message marked in some way
  - Using invisible ink
  - Hiding in lsb in graphic image or sound file
- Has drawbacks
  - High overhead to hide relatively few info bits

MODERN BLOCK CIPHERS

- Now look at modern block ciphers
- One of the most widely used types of cryptographic algorithms
- Provide secrecy /authentication services
- Focus on des (data encryption standard)
- To illustrate block cipher design principles

BLOCK VS STREAM CIPHERS

- Block ciphers process messages in blocks, each of which is then en/decrypted
- Like a substitution on very big characters
  - 64-bits or more
- Stream ciphers process messages a bit or byte at a time when en/decrypting
- Many current ciphers are block ciphers
- Broader range of applications

3. Briefly explain design principles of block cipher

1.3 BLOCK CIPHER PRINCIPLES

- Most symmetric block ciphers are based on a **feistel cipher structure**
- Needed since must be able to **decrypt** ciphertext to recover messages efficiently
- Block ciphers look like an extremely large substitution
- Would need table of 264 entries for a 64-bit block
- Instead create from smaller building blocks
- Using idea of a product cipher

CLAUDE SHANNON AND SUBSTITUTION-PERMUTATION CIPHERS

- Claude shannon introduced idea of substitution-permutation (s-p) networks in 1949 paper
- Form basis of modern block ciphers

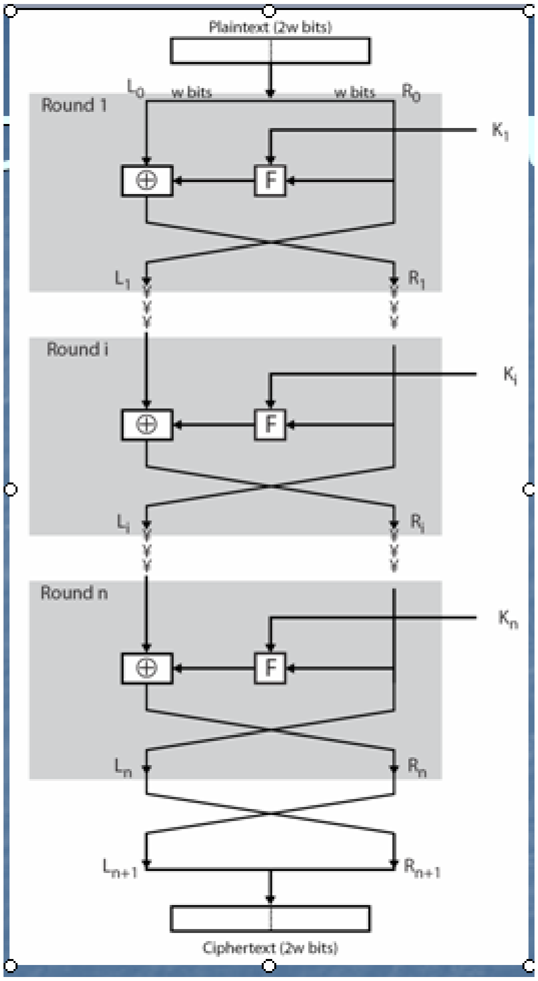
- S-p nets are based on the two primitive cryptographic operations seen before:
  - *Substitution* (s-box)
  - *Permutation* (p-box)
- Provide *confusion* & *diffusion* of message & key

CONFUSION AND DIFFUSION

- Cipher needs to completely obscure statistical properties of original message
- A one-time pad does this
- More practically shannon suggested combining s & p elements to obtain:
- Diffusion – dissipate statistical structure of plaintext over bulk of ciphertext
- Confusion – makes relationship between ciphertext and key as complex as possible

FEISTEL CIPHER STRUCTURE

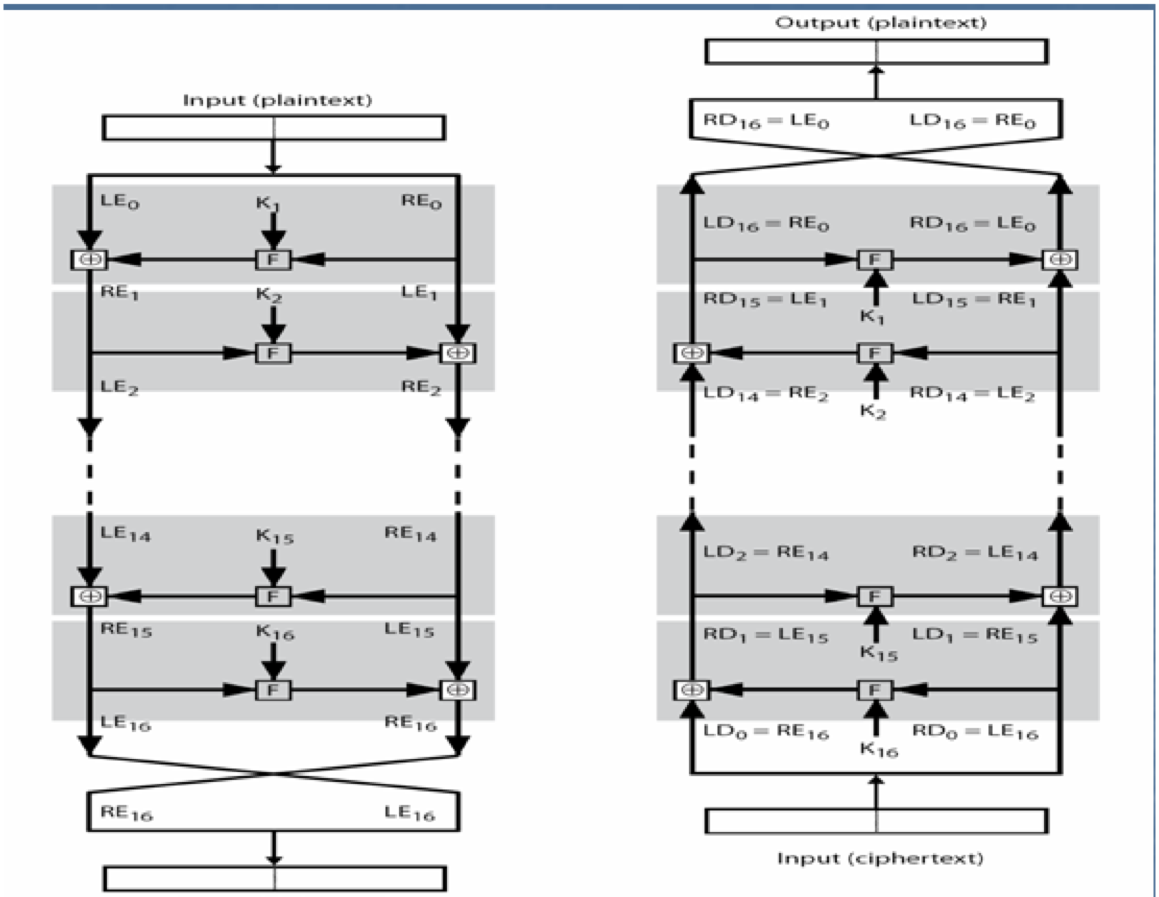
- Horst feistel devised the **feistel cipher**
  - Based on concept of invertible product cipher
- Partitions input block into two halves
  - Process through multiple rounds which
  - Perform a substitution on left data half
  - Based on round function of right half & subkey
  - Then have permutation swapping halves
- Implements shannon's s-p net concept



FEISTEL CIPHER DESIGN ELEMENTS

- Block size
- Key size
- Number of rounds
- Subkey generation algorithm
- Round function
- Fast software en/decryption
- Ease of analysis

FEISTEL CIPHER DECRYPTION



4. Describe the working principle of DES with an example.

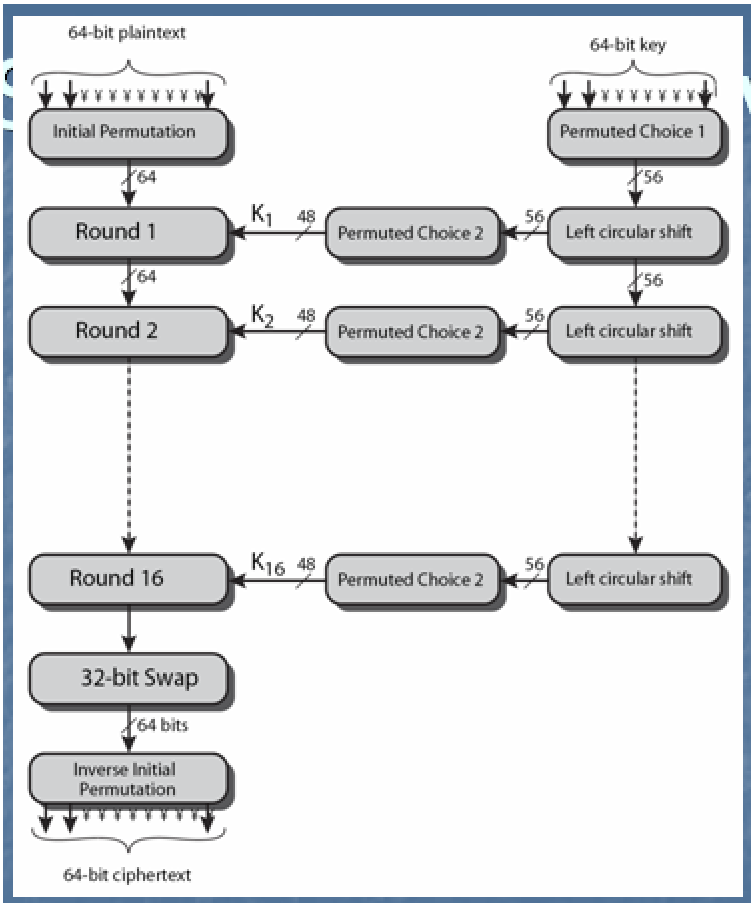
DATA ENCRYPTION STANDARD (DES)

- Most widely used block cipher in world
- Adopted in 1977 by nbs (now nist)
  - As fips pub 46
- Encrypts 64-bit data using 56-bit key
- Has widespread use
- Has been considerable controversy over its security

DES DESIGN CONTROVERSY

- Although des standard is public
- Was considerable controversy over design
  - In choice of 56-bit key (vs lucifer 128-bit)
  - And because design criteria were classified
- Subsequent events and public analysis show in fact design was appropriate
- Use of des has flourished
  - Especially in financial applications
  - Still standardised for legacy application use

DES ENCRYPTION OVERVIEW



INITIAL PERMUTATION IP

- First step of the data computation
- Ip reorders the input data bits
- Even bits to lh half, odd bits to rh half
- Quite regular in structure (easy in h/w)
- Example:

  ip(675a6967 5e5a6b5a) = (ffb2194d 004df6fb)

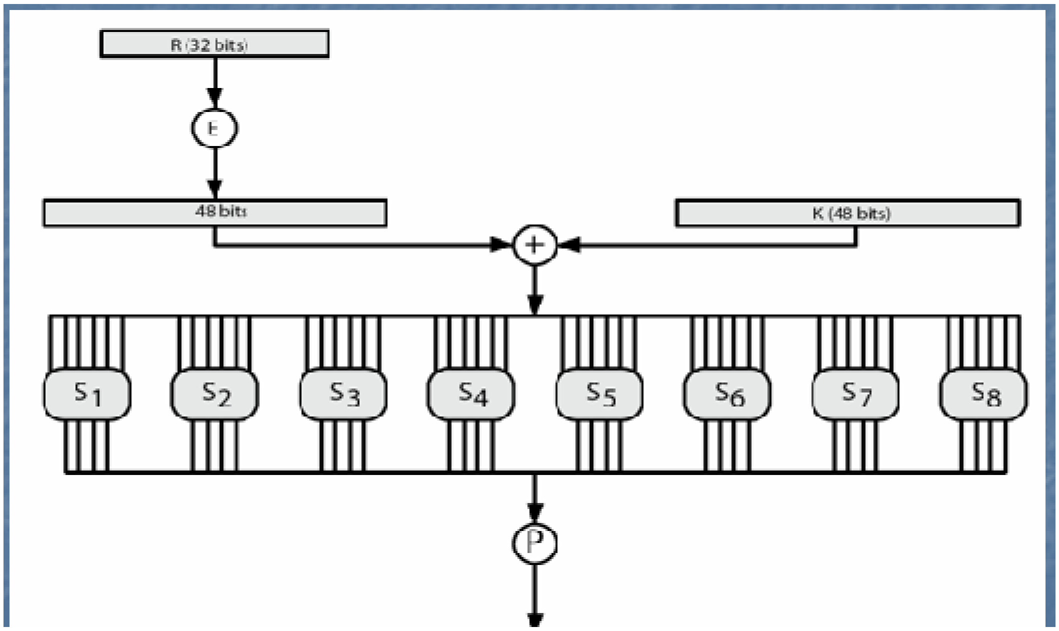
Des round structure

- Uses two 32-bit l & r halves
- As for any feistel cipher can describe as:

$Li = ri-1$

$Ri = li-1 \oplus f(ri-1, ki)$

- F takes 32-bit r half and 48-bit subkey:
    - Expands r to 48-bits using perm e
    - Adds to subkey using xor
    - Passes through 8 s-boxes to get 32-bit result
    - Finally permutes using 32-bit perm p



SUBSTITUTION BOXES S

- Have eight s-boxes which map 6 to 4 bits
- Each s-box is actually 4 little 4 bit boxes
    - Outer bits 1 & 6 (**row** bits) select one row of 4
    - Inner bits 2-5 (**col** bits) are substituted
    - Result is 8 lots of 4 bits, or 32 bits
- Row selection depends on both data & key
    - Feature known as autoclaving (autokeying)
- Example:
    - S(18 09 12 3d 11 17 38 39) = 5fd25e03

DES KEY SCHEDULE

- Forms subkeys used in each round
    - Initial permutation of the key (pc1) which selects 56-bits in two 28-bit halves
    - 16 stages consisting of:
        - Rotating **each half** separately either 1 or 2 places depending on the **key rotation schedule** k
        - Selecting 24-bits from each half & permuting them by pc2 for use in round function f
- Note practical use issues in h/w vs s/w

DES DECRYPTION

- Decrypt must unwind steps of data computation
- With feistel design, do encryption steps again using subkeys in reverse order (sk16 … sk1)
    - Ip undoes final fp step of encryption
    - 1st round with sk16 undoes 16th encrypt round

    - ….
    - 16th round with sk1 undoes 1st encrypt round
    - Then final fp undoes initial encryption ip
    - Thus recovering original data value

AVALANCHE EFFECT

- Key desirable property of encryption alg
- Where a change of **one** input or key bit results in changing approx **half** output bits
- Making attempts to "home-in" by guessing keys impossible
- Des exhibits strong avalanche

STRENGTH OF DES – KEY SIZE

- 56-bit keys have 256 = 7.2 x 1016 values
- Brute force search looks hard
- Recent advances have shown is possible
    - In 1997 on internet in a few months
    - In 1998 on dedicated h/w (eff) in a few days
    - In 1999 above combined in 22hrs!
- Still must be able to recognize plaintext
- Must now consider alternatives to des

STRENGTH OF DES – ANALYTIC ATTACKS

- Now have several analytic attacks on des
- These utilise some deep structure of the cipher
    - By gathering information about encryptions
    - Can eventually recover some/all of the sub-key bits
    - If necessary then exhaustively search for the rest
- Generally these are statistical attacks
- Include
    - Differential cryptanalysis
    - Linear cryptanalysis
    - Related key attacks

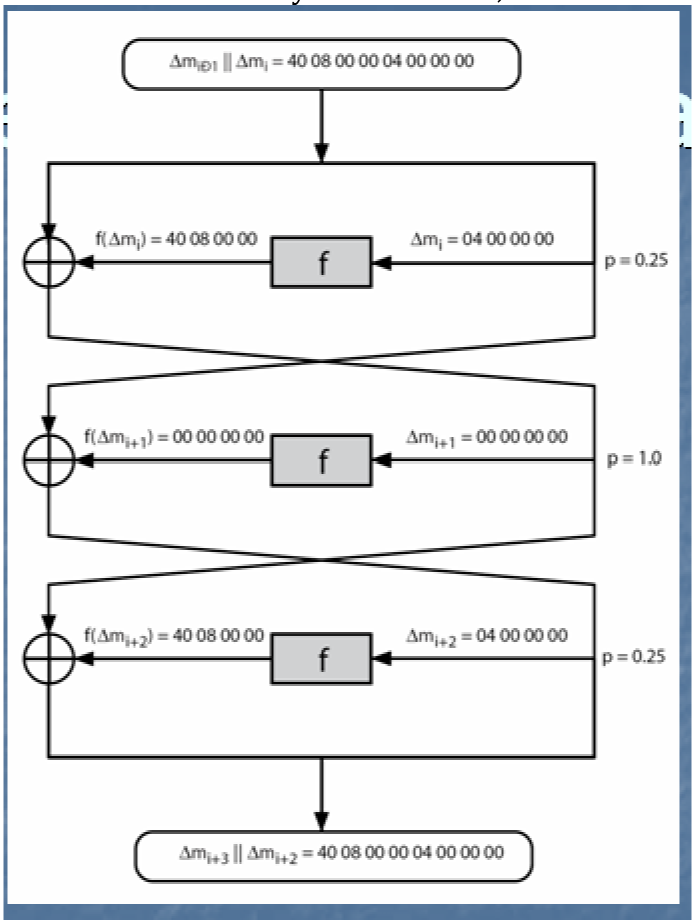
STRENGTH OF DES – TIMING ATTACKS

- Attacks actual implementation of cipher
- Use knowledge of consequences of implementation to derive information about some/all subkey bits
- Specifically use fact that calculations can take varying times depending on the value of the inputs to it
- Particularly problematic on smartcards

DIFFERENTIAL CRYPTANALYSIS

- One of the most significant recent (public) advances in cryptanalysis
- Known by nsa in 70's cf des design
- Murphy, biham & shamir published in 90's
- Powerful method to analyse block ciphers

- Used to analyse most current block ciphers with varying degrees of success
- Des reasonably resistant to it, cf lucifer



- Perform attack by repeatedly encrypting plaintext pairs with known input xor until obtain desired output xor
- When found
  - If intermediate rounds match required xor have a **right pair**
  - If not then have a **wrong pair**, relative ratio is s/n for attack
- Can then deduce keys values for the rounds
  - Right pairs suggest same key bits
  - Wrong pairs give random values
- For large numbers of rounds, probability is so low that more pairs are required than exist with 64-bit inputs
- Biham and shamir have shown how a 13-round iterated characteristic can break the full 16-round des

LINEAR CRYPTANALYSIS

- Another recent development
- Also a statistical method
- Must be iterated over rounds, with decreasing probabilities
- Developed by matsui et al in early 90's
- Based on finding linear approximations

- Can attack des with 243 known plaintexts, easier but still in practise infeasible
- Find linear approximations with prob p != ½

P[i1,i2,...,ia] ⊕ c[j1,j2,...,jb] = k[k1,k2,...,kc]
Where ia,jb,kc are bit locations in p,c,k

- Gives linear equation for key bits
- Get one key bit using max likelihood alg
- Using a large number of trial encryptions
- Effectiveness given by: |p–1/2|

5. Explain in detail the transformations take place in AES encryption procedure

ADVANCED ENCRYPTION STANDARD

Origins

- Clear a replacement for des was needed
  - Have theoretical attacks that can break it
  - Have demonstrated exhaustive key search attacks
- Can use triple-des – but slow with small blocks
- Us nist issued call for ciphers in 1997.
- 15 candidates accepted in jun 98.
- 5 were short listed in aug-99.
- Rijndael was selected as the aes in oct-2000.
- Issued as fips pub 197 standard in nov-2001.

Aes Evaluation Criteria

- Initial criteria:
  - Security – effort to practically cryptanalysis
  - Cost – computational
  - Algorithm & implementation characteristics
- Final criteria:
  - General security
  - Software & hardware implementation ease
  - Implementation attacks
  - Flexibility (in en/decrypt, keying, other factors)

The Aes Cipher - Rijndael

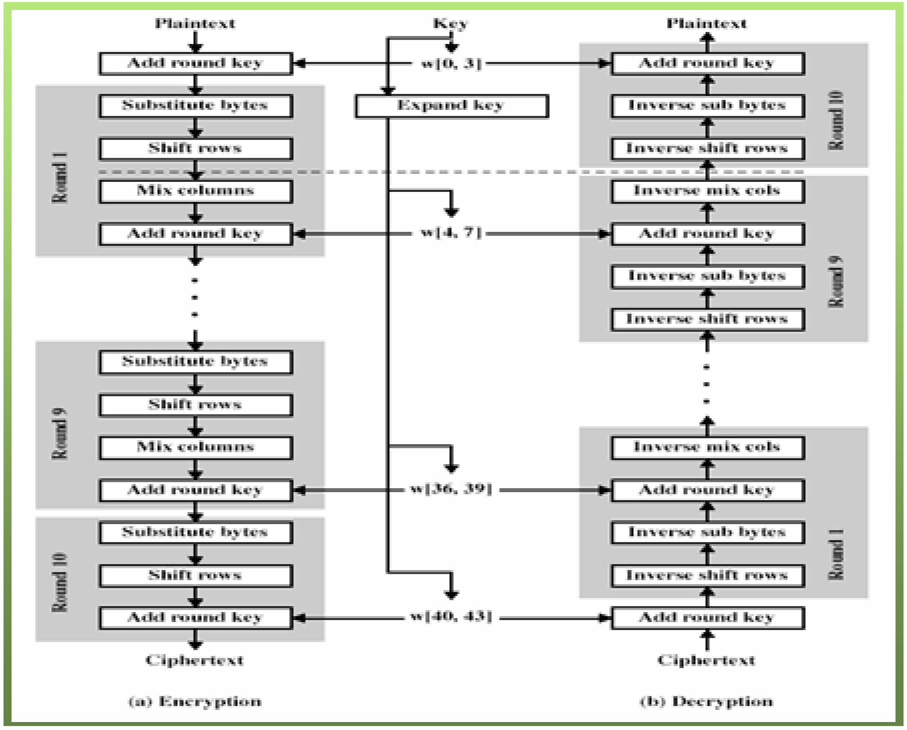
- Designed by rijmen-daemen in belgium
- Has 128/192/256 bit keys, 128 bit data
- An **iterative** rather than **feistel** cipher
  - Treats data in 4 groups of 4 bytes
  - Operates an entire block in every round
- Designed to be:
  - Resistant against known attacks
  - Speed and code compactness on many cpus
  - Design simplicity

Rijndael

- Processes data as 4 groups of 4 bytes (state)
- 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 multiply of groups)
- Add round key (xor state with key material)

- Initial xor key material & incomplete last round
- All operations can be combined into xor and table lookups - hence very fast & efficient



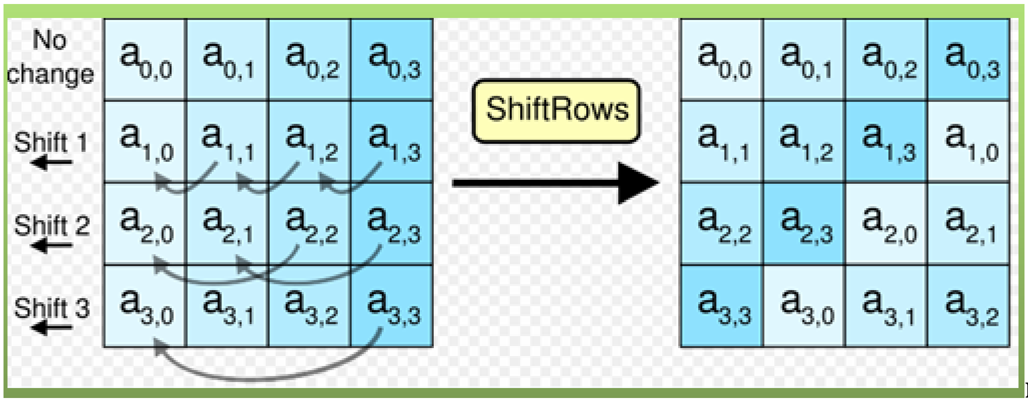
Byte Substitution

- A simple substitution of each byte
- Uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
- Each byte of state is replaced by byte in row (left 4-bits) & column (right 4-bits)
  - Eg. Byte {95} is replaced by row 9 col 5 byte
  - Which is the value {2a}
- S-box is constructed using a defined transformation of the values in gf(28)
- Designed to be resistant to all known attacks

Shift rows

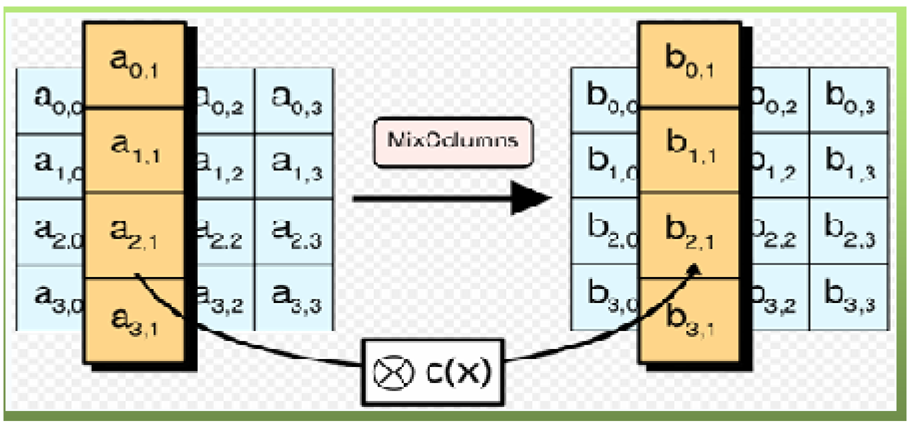
- Circular byte shift in each
  - 1st row is unchanged
  - 2nd row does 1 byte circular shift to left
  - 3rd row does 2 byte circular shift to left
  - 4th row does 3 byte circular shift to left
- Decrypt does shifts to right

- Since state is processed by columns, this step permutes bytes between the columns



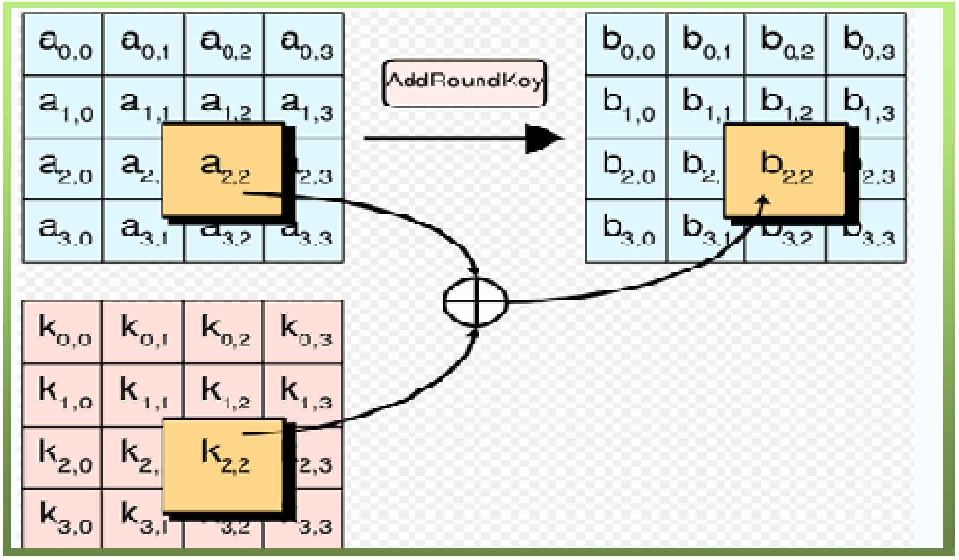
mix **columns**

- Each column is processed separately
- Each byte is replaced by a value dependent on all 4 bytes in the column
- Effectively a matrix multiplication in gf(28) using prime poly m(x) =x8+x4+x3+x+1



Add round key

- Xor state with 128-bits of the round key
- Again processed by column (though effectively a series of byte operations)
- Inverse for decryption is identical since xor is own inverse, just with correct round key
- Designed to be as simple as possible



Aes decryption

- Aes decryption is not identical to encryption since steps done in reverse
- But can define an equivalent inverse cipher with steps as for encryption
  - But using inverses of each step
  - With a different key schedule
- Works since result is unchanged when
  - Swap byte substitution & shift rows
  - Swap mix columns & add (tweaked) round key

Implementation aspects

- Can efficiently implement on 8-bit cpu
  - Byte substitution works on bytes using a table of 256 entries
  - Shift rows is simple byte shifting
  - Add round key works on byte xors
  - Mix columns requires matrix multiply in gf(28) which works on byte values, can be simplified to use a table lookup
- Can efficiently implement on 32-bit cpu
  - Redefine steps to use 32-bit words
  - Can precompute 4 tables of 256-words
  - Then each column in each round can be computed using 4 table lookups + 4 xors
  - At a cost of 16kb to store tables
- Designers believe this very efficient implementation was a key factor in its selection as the aes cipher

TRIPLE DES

- Clear a replacement for des was needed
  - Theoretical attacks that can break it
  - Demonstrated exhaustive key search attacks
- Aes is a new cipher alternative
- Prior to this alternative was to use multiple encryption with des implementations

- Triple-des is the chosen form

Triple-des with two-keys

- Hence must use 3 encryptions
  - Would seem to need 3 distinct keys
- But can use 2 keys with e-d-e sequence
  - C = ek1[dk2[ek1[p]]]
  - Nb encrypt & decrypt equivalent in security
  - If k1=k2 then can work with single des
- Standardized in ansi x9.17 & iso8732
- No current known practical attacks

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 = ek3[dk2[ek1[p]]]
- Has been adopted by some internet applications, eg pgp, s/mime

Blowfish

- A symmetric block cipher designed by bruce schneier in 1993/94
- Characteristics
  - Fast implementation on 32-bit cpus
  - Compact in use of memory
  - Simple structure eases analysis/implemention
  - Variable security by varying key size
- Has been implemented in various products
- Uses a 32 to 448 bit key
- Used to generate
  - 18 32-bit subkeys stored in k-array kj
  - Four 8x32 s-boxes stored in si,j
- Key schedule consists of:
  - Initialize p-array and then 4 s-boxes using pi
  - Xor p-array with key bits (reuse as needed)
  - Loop repeatedly encrypting data using current p & s and replace successive pairs of p then s values
  - Requires 521 encryptions, hence slow in rekeying

Blowfish encryption

- Uses two primitives: addition & xor
- Data is divided into two 32-bit halves *l0* & *r0*

For *i* = 1 to 16 do

Ri = *li-1* xor *pi*;

Li = f[*ri*] xor *ri-1*;

*L*17 = *r*16 xor *p*18;

*R*17 = *l*16 xor i17;

- Where

F[*a*,*b*,*c*,*d*] = ((s1,*a* + s2,b) xor s3,c) + s4,*a*

Rc5

- A proprietary cipher owned by rsadsi
- Designed by ronald rivest (of rsa fame)
- Used in various rsadsi products
- Can vary key size / data size / no rounds
- Very clean and simple design
- Easy implementation on various cpus
- Yet still regarded as secure
- Rc5 is a family of ciphers rc5-w/r/b
  - W = word size in bits (16/32/64) nb data=2w
  - R = number of rounds (0..255)
  - B = number of bytes in key (0..255)
- Nominal version is rc5-32/12/16
  - Ie 32-bit words so encrypts 64-bit data blocks
  - Using 12 rounds
  - With 16 bytes (128-bit) secret key

Rc5 encryption

- Split input into two halves a & b

$L0 = a + s[0];$

$R0 = b + s[1];$

For i = 1 to r do

$Li = ((li\text{-}1 \text{ xor } ri\text{-}1) \lll ri\text{-}1) + s[2 \text{ x } i];$

$Ri = ((ri\text{-}1 \text{ xor } li) \lll li) + s[2 \text{ x } i + 1];$

- Each round is like 2 des rounds
- Note rotation is main source of non-linearity
- Need reasonable number of rounds (eg 12-16)

Block cipher characteristics

- Features seen in modern block ciphers are:
  - Variable key length / block size / no rounds
  - Mixed operators, data/key dependent rotation
  - Key dependent s-boxes
  - More complex key scheduling
  - Operation of full data in each round
  - Varying non-linear functions

Stream ciphers

- Process the message bit by bit (as a stream)
- Typically have a (pseudo) random **stream key**
- Combined (xor) with plaintext bit by bit
- Randomness of **stream key** completely destroys any statistically properties in the message
  - Ci = mi xor streamkeyi
- What could be simpler!!!!

- But must never reuse stream key
  - Otherwise can remove effect and recover messages

Rc4

- A proprietary cipher owned by rsa dsi
- Another ron rivest design, simple but effective
- Variable key size, byte-oriented stream cipher
- Widely used (web ssl/tls, wireless wep)
- Key forms random permutation of all 8-bit values
- Uses that permutation to scramble input info processed a byte at a time

Rc4 key schedule

- Starts with an array s of numbers: 0..255
- Use key to well and truly shuffle
- S forms internal state of the cipher
- Given a key k of length l bytes

For i = 0 to 255 do
S[i] = i
J = 0
For i = 0 to 255 do
J = (j + s[i] + k[i mod l]) (mod 256)
Swap (s[i], s[j])

Rc4 encryption

- Encryption continues shuffling array values
- Sum of shuffled pair selects "stream key" value
- Txor with next byte of message to en/decrypt

I = j = 0
For each message byte mi
I = (i + 1) (mod 256)
J = (j + s[i]) (mod 256)
Swap(s[i], s[j])
T = (s[i] + s[j]) (mod 256)
Ci = mi xor s[t]

Rc4 security

- Claimed secure against known attacks
  - Have some analyses, none practical
- Result is very non-linear
- Since rc4 is a stream cipher, must **never reuse a key**
- Have a concern with wep, but due to key handling rather than rc4 itself

UNIT II

BLOCK CIPHERS & PUBLIC KEY CRYPTOGRAPHY

1. Describe Euler's and Fermat's theorem.

Fermat's theorem

- **Fermat's little theorem** (not to be confused with fermat's last theorem) states that if *p* is a prime number, then for any integer *a*, *ap* − *a* will be evenly divisible by *p*. This can be expressed in the notation of modular arithmetic as follows:
- A variant of this theorem is stated in the following form: if *p* is a prime and *a* is an integer coprime to *p*, then *ap* − 1 − 1 will be evenly divisible by *p*. In the notation of modular arithmetic:
- Ap-1 = 1 (mod p)
  - Where p is prime and gcd(a,p)=1
- Also known as fermat's little theorem
- Also ap = p (mod p)
- Useful in public key and primality testing

Euler totient function ø(n)

- When doing arithmetic modulo n
- **Complete set of residues** is: 0..n-1
-
- **Reduced set of residues** is those numbers (residues) which are relatively prime to n
  - Eg for n=10,
  - Complete set of residues is {0,1,2,3,4,5,6,7,8,9}
  - Reduced set of residues is {1,3,7,9}
- Number of elements in reduced set of residues is called the **euler totient function ø(n)**
- To compute ø(n) need to count number of residues to be excluded
- In general need prime factorization, but
  - For p (p prime) ø(p) = p-1
  - For p.q (p,q prime) ø(pq) =(p-1)x(q-1)
- Eg.

Ø(37) = 36

Ø(21) = (3–1)x(7–1) = 2x6 = 12

Euler's theorem

- A generalisation of fermat's theorem
- Aø(n) = 1 (mod n)
  - For any a,n where gcd(a,n)=1
- Eg.

A=3;*n*=10; ø(10)=4;

hence 34 = 81 = 1 mod 10

A=2;*n*=11; ø(11)=10;

hence 210 = 1024 = 1 mod 11

Miller rabin algorithm

- A test based on fermat's theorem
- Algorithm is:

Test (*n*) is:

1. Find integers *k*, *q*, *k* > 0, *q* odd, so that (*n*–1)=2*kq*
2. Select a random integer *a*, 1<*a*<*n*–1
3. **If** *aq* mod *n* = 1 **then** return ("maybe prime");
4. **For** *j* = 0 **to** *k* – 1 **do**
    5. **If** (*a*2*jq* mod *n* = *n*-1)
        then return(" maybe prime ")
6. Return ("composite")

Prime distribution

- Prime number theorem states that primes occur roughly every (ln n) integers
- But can immediately ignore evens
- So in practice need only test 0.5 ln(n) numbers of size n to locate a prime
    - Note this is only the "average"
    - Sometimes primes are close together
    - Other times are quite far apart

Discrete logarithms

- The inverse problem to exponentiation is to find the **discrete logarithm** of a number modulo p
- That is to find x such that y = gx (mod p)
- This is written as x = logg y (mod p)
- If g is a primitive root then it always exists, otherwise it may not, eg.

X = log3 4 mod 13 has no answer

X = log2 3 mod 13 = 4 by trying successive powers

- Whilst exponentiation is relatively easy, finding discrete logarithms is generally a **hard** problem

2. Describe Public Key Cryptography.

Private key

- Traditional **private/secret/single key** cryptography uses **one** key
- Shared by both sender and receiver
- If this key is disclosed communications are compromised
- Also is **symmetric**, parties are equal
- Hence does not protect sender from receiver forging a message & claiming is sent by sender
- Probably most significant advance in the 3000 year history of cryptography

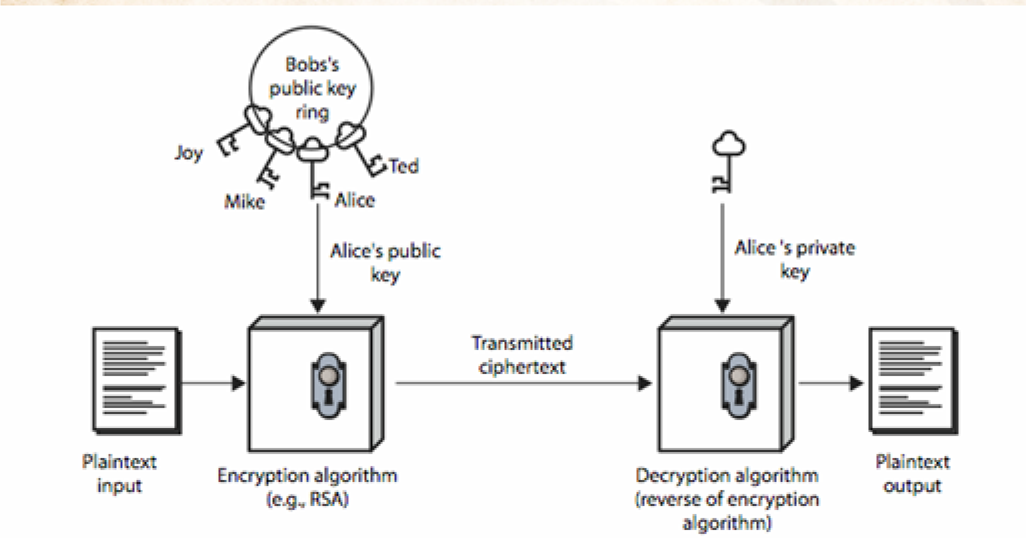
- Uses **two** keys – a public & a private key
- **Asymmetric** since parties are **not** equal
- Uses clever application of number theoretic concepts to function
- Complements **rather than** replaces private key crypto
- Developed to address two key issues:

    - Key distribution **– how to have secure communications in general without having to trust a kdc with your key**

    - Digital signatures **– how to verify a message comes intact from the claimed sender**
- Public invention due to whitfield diffie & martin hellman at stanford uni in 1976
    - **Known earlier in classified community**

- Public-key/two-key/asymmetric cryptography involves the use of two keys:
    - A public-key, which may be known by anybody, and can be used to encrypt messages, and verify signatures
    - A private-key, known only to the recipient, used to decrypt messages, and sign (create) signatures

- Is asymmetric because
    - Those who encrypt messages or verify signatures cannot decrypt messages or create signatures

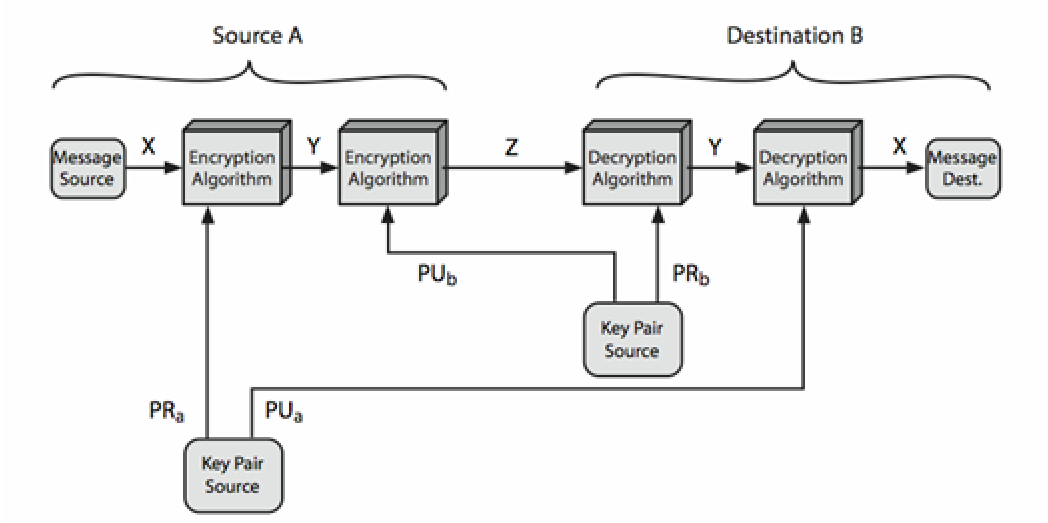


(a) Encryption

Public-key characteristics

- Public-key algorithms rely on two keys where:
    - It is computationally infeasible to find decryption key knowing only algorithm & encryption key
    - It is computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known

    - Either of the two related keys can be used for encryption, with the other used for decryption (for some algorithms)



Public-key applications

- Can classify uses into 3 categories:
    - Encryption/decryption (provide secrecy)
    - Digital signatures (provide authentication)
    - Key exchange (of session keys)
- Some algorithms are suitable for all uses, others are specific to one

3. Explain RSA method in detail.

RSA

- 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)

RSA key setup

- 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}

RSA works

- Because of euler's theorem:
  - Aø(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 :

  cd = me.d = m1+k.ø(n) = m1.(mø(n))k

  = m1.(1)k = m1 = m mod n

RSA example - key setup

1. Select primes: *p*=17 & *q*=11
2. Compute *n* = *pq* =17 x 11=187
3. Compute ø(*n*)=(*p*–1)(*q*-1)=16 x 10=160
4. Select e: gcd(e,160)=1; choose *e*=7
5. Determine d: *de*=1 mod 160 and *d* < 160 value is d=23 since 23x7=161= 10x160+1
6. Publish public key pu={7,187}
7. Keep secret private key pr={23,187}

RSA example - en/decryption

- Sample RSA encryption/decryption is:
- Given message m = 88 (nb. 88<187)
- Encryption:

C = 887 mod 187 = 11

- Decryption:

M = 1123 mod 187 = 88

RSA security

- Possible approaches to attacking rsa are:
  - Brute force key search (infeasible given size of numbers)
  - Mathematical attacks (based on difficulty of computing ø(n), by factoring modulus n)
  - Timing attacks (on running of decryption)
  - Chosen ciphertext attacks (given properties of rsa)

Factoring problem

- Mathematical approach takes 3 forms:
  - Factor n=p.q, hence compute ø(n) and then d
  - Determine ø(n) directly and compute d

    - Find d directly
- Currently believe all equivalent to factoring
    - Have seen slow improvements over the years
        - As of may-05 best is 200 decimal digits (663) bit with ls
    - Biggest improvement comes from improved algorithm
        - Cf qs to ghfs to ls
    - Currently assume 1024-2048 bit rsa is secure
        - Ensure p, q of similar size and matching other constraints

4. Describe public key management and cryptosystems

Key management

- Public-key encryption helps address key distribution problems

- Have two aspects of this:
    - Distribution of public keys
    - Use of public-key encryption to distribute **secret keys**

Distribution of public keys

- Can be considered as using one of:

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

Public announcement

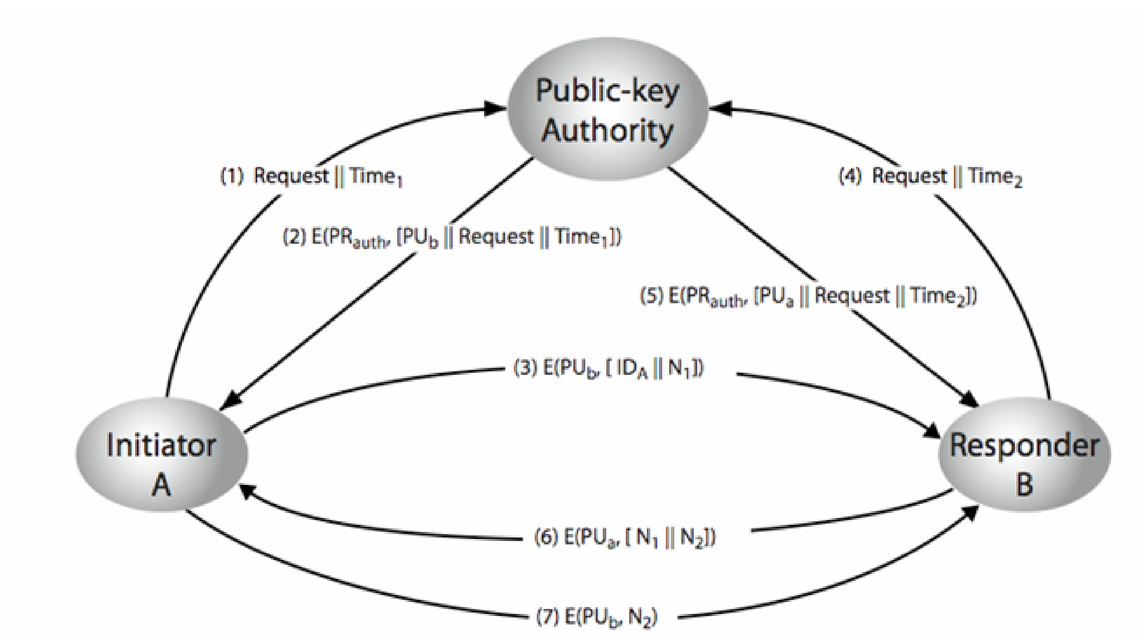
- Users distribute public keys to recipients or broadcast to community at large
    - Eg. Append pgp keys to email messages or post to news groups or email list
- Major weakness is forgery
    - Anyone can create a key claiming to be someone else and broadcast it
    - Until forgery is discovered can masquerade as claimed user

Publicly available directory

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

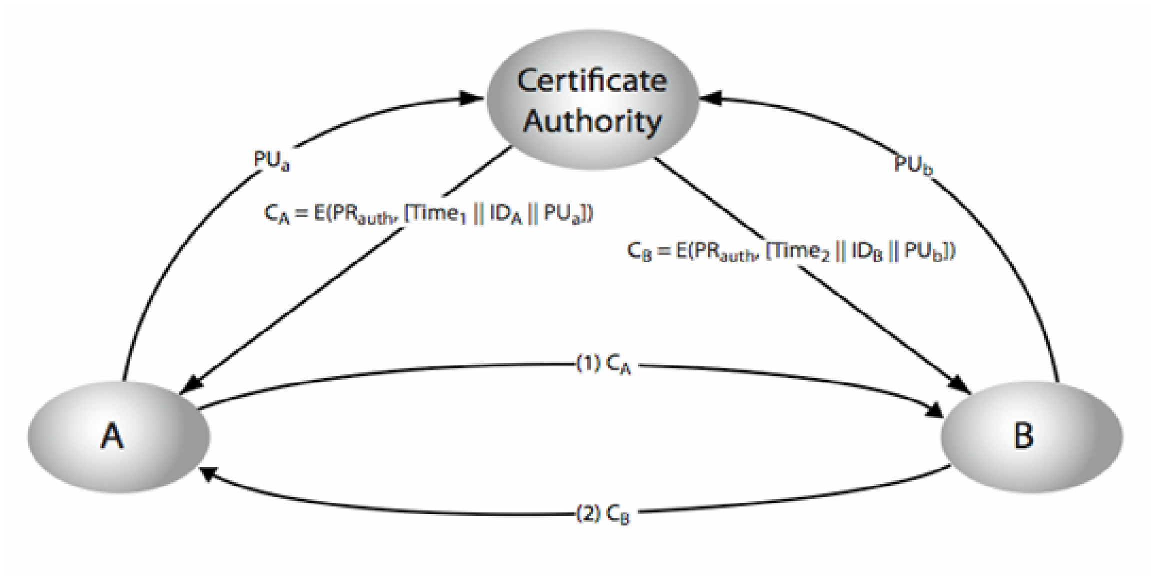
Public-key authority

- Improve security by tightening control over distribution of keys from directory.
- Has properties of directory.
- And requires users to know public key for the directory.
- Then users interact with directory to obtain any desired public key securely.
  - 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**
  - 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.

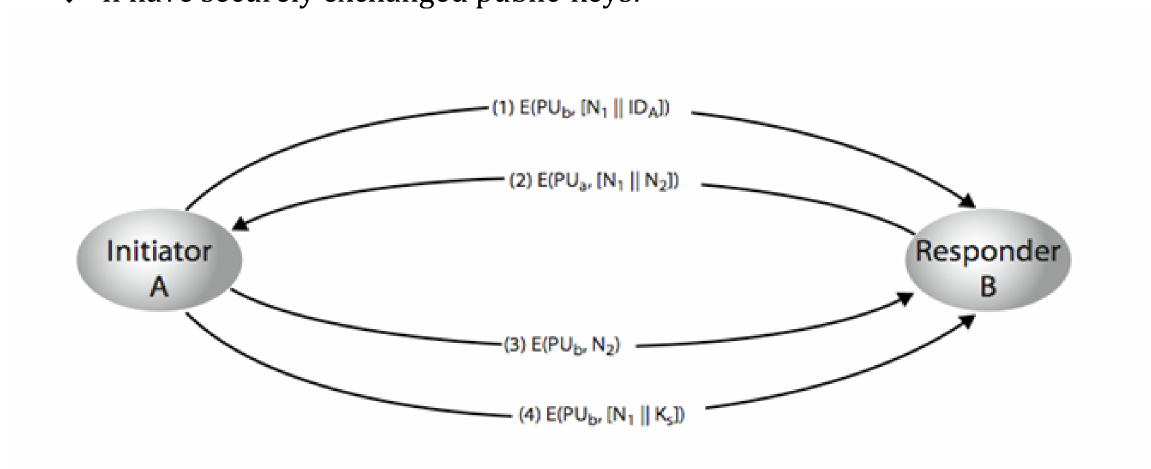


Public-key distribution of secret keys

- Use previous methods to obtain public-key
- Can use for secrecy or authentication
- But public-key algorithms are slow
- So usually want to use private-key encryption to protect message contents
- Hence need a session key
- Have several alternatives for negotiating a suitable session

Public-key distribution of secret keys

- If have securely exchanged public-keys:



Hybrid key distribution

- Retain use of private-key kdc
- Shares secret master key with each user

- Distributes session key using master key
- Public-key used to distribute master keys
  - Especially useful with widely distributed users
- Rationale
  - Performance
  - Backward compatibility

4. Briefly explain the Diffie-Hellman Key Exchange

DIFFIE-HELLMAN KEY EXCHANGE

- First public-key type scheme proposed
- By diffie & hellman in 1976 along with the exposition of public key concepts
  - Note: now know that williamson (uk cesg) secretly proposed the concept in 1970
- Is a practical method for public exchange of a secret key
- Used in a number of commercial products
- A public-key distribution scheme
  - Cannot be used to exchange an arbitrary message
  - Rather it can establish a common key
  - Known only to the two participants
- Value of key depends on the participants (and their private and public key information)
- Based on exponentiation in a finite (galois) field (modulo a prime or a polynomial) - easy
- Security relies on the difficulty of computing discrete logarithms (similar to factoring) – hard

Diffie-hellman setup

- All users agree on global parameters:
  - Large prime integer or polynomial q
  - A being a primitive root mod q
- Each user (eg. A) generates their key

- Chooses a secret key (number): xa < q
- Compute their **public key**: ya = axa mod q

◆ each user makes public that key ya

Diffie-hellman key exchange

◆ Shared session key for users a & b is kab:

Kab = axa.xb mod q
= yaxb mod q (which b can compute)
= ybxa mod q (which a can compute)

◆ Kab is used as session key in private-key encryption scheme between alice and bob
◆ If alice and bob subsequently communicate, they will have the same key as before, unless they choose new public-keys
◆ Attacker needs an x, must solve discrete log

Diffie-hellman example

◆ Users alice & bob who wish to swap keys:
◆ Agree on prime q=353 and a=3
◆ Select random secret keys:
  - A chooses xa=97, b chooses xb=233
◆ Compute respective public keys:
  - Ya=397 mod 353 = 40 (alice)
  - Yb=3233 mod 353 = 248 (bob)
◆ Compute shared session key as:
  - Kab= ybxa mod 353 = 24897 = 160 (alice)
  - Kab= yaxb mod 353 = 40233 = 160 (bob)

Key exchange protocols

◆ Users could create random private/public d-h keys each time they communicate
◆ Users could create a known private/public d-h key and publish in a directory, then consulted and used to securely communicate with them
◆ Both of these are vulnerable to a meet-in-the-middle attack
◆ Authentication of the keys is needed

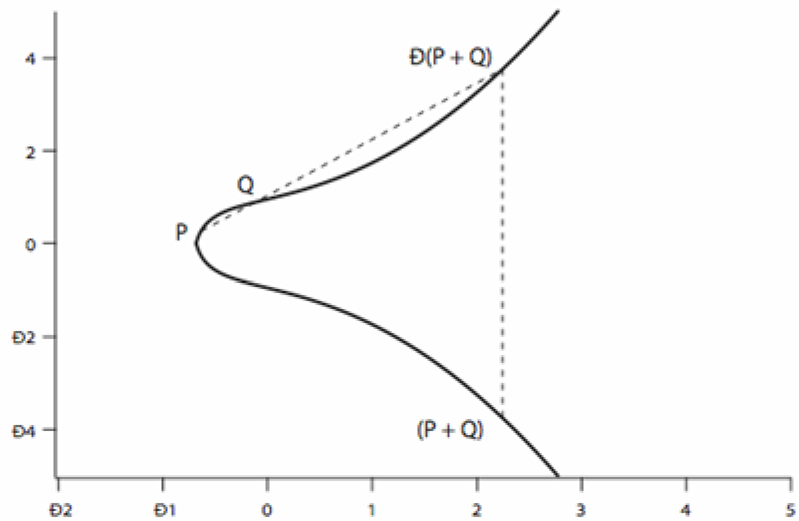
5. Briefly describe the idea behind Elliptic Curve Cryptosystems.

◆ Majority of public-key crypto (rsa, d-h) use either integer or polynomial arithmetic with very large numbers/polynomials
◆ Imposes a significant load in storing and processing keys and messages
◆ An alternative is to use elliptic curves
◆ Offers same security with smaller bit sizes
◆ Newer, but not as well analysed

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
  - $Y2 = x3 + ax + b$
  - 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

Real elliptic curve example



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

Finite elliptic curves

- Elliptic curve cryptography uses curves whose variables & coefficients are finite
- Have two families commonly used:
  - Prime curves ep(a,b) defined over zp
    - Use integers modulo a prime
    - Best in software
  - Binary curves e2m(a,b) defined over gf(2n)
    - Use polynomials with binary coefficients
    - Best in hardware

Elliptic curve cryptography

- Ecc addition is analog of modulo multiply
- Ecc repeated addition is analog of modulo exponentiation
- Need “hard” problem equiv to discrete log
  - 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
- Certicom example: e23(9,17)

Ecc diffie-hellman

- Can do key exchange analogous to d-h
- Users select a suitable curve ep(a,b)
- Select base point g=(x1,y1)
  - With large order n s.t. Ng=o
- A & b select private keys na<n, nb<n
- Compute public keys: pa=nag, pb=nbg
- Compute shared key: k=napb, k=nbpa
  - Same since k=nanbg

Ecc encryption/decryption

- Several alternatives, will consider simplest
- Must first encode any message m as a point on the elliptic curve pm
- Select suitable curve & point g as in d-h
- Each user chooses private key na<n
- And computes public key pa=nag
- To encrypt pm : cm={kg, pm+kpb}, k random
- Decrypt cm compute:

$Pm+kpb-nb(kg) = pm+k(nbg)-nb(kg) = pm$

Ecc security

- Relies on elliptic curve logarithm problem
- Fastest method is "pollard rho method"
- Compared to factoring, can use much smaller key sizes than with rsa etc
- For equivalent key lengths computations are roughly equivalent
- Hence for similar security ecc offers significant computational advantages

UNIT III
HASH FUNCTIONS AND DIGITAL SIGNATURES

1. Give a brief notes on message authentications and services.

Message authentication

- Message authentication is concerned with:
  - Protecting the integrity of a message
  - Validating identity of originator
  - Non-repudiation of origin (dispute resolution)
- Will consider the security requirements
- Then three alternative functions used:
  - Message encryption
  - Message authentication code (mac)
  - Hash function

Security requirements

- Disclosure
- Traffic analysis
- Masquerade
- Content modification
- Sequence modification
- Timing modification
- Source repudiation
- Destination repudiation

Message encryption

- Message encryption by itself also provides a measure of authentication

- If symmetric encryption is used then:
    - Receiver know sender must have created it
    - Since only sender and receiver now key used
    - Know content cannot of been altered
    - If message has suitable structure, redundancy or a checksum to detect any changes

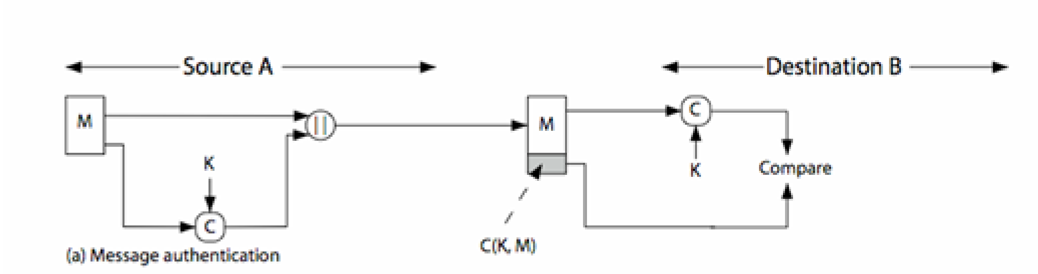
- If public-key encryption is used:
    - Encryption provides no confidence of sender
    - Since anyone potentially knows public-key
    - However if
        - Sender **signs** message using their private-key
        - Then encrypts with recipients public key
        - Have both secrecy and authentication
    - Again need to recognize corrupted messages
    - But at cost of two public-key uses on message

2. Briefly describe about MAC in detail.

MESSAGE AUTHENTICATION CODE (MAC)

- Generated by an algorithm that creates a small fixed-sized block
    - Depending on both message and some key
    - Like encryption though need not be reversible
- Appended to message as a **signature**
- Receiver performs same computation on message and checks it matches the mac
- Provides assurance that message is unaltered and comes from sender

Message authentication code



- As shown the mac provides authentication
- Can also use encryption for secrecy
  - Generally use separate keys for each
  - Can compute mac either before or after encryption
  - Is generally regarded as better done before
- Why use a mac?
  - Sometimes only authentication is needed
  - Sometimes need authentication to persist longer than the encryption (eg. Archival use)
- Note that a mac is not a digital signature

MAC properties

- A mac is a cryptographic checksum
  mac = ck(m)
  - Condenses a variable-length message m
  - Using a secret key k
  - To a fixed-sized authenticator
- Is a many-to-one function
  - Potentially many messages have same mac
  - But finding these needs to be very difficult

Requirements for MACS

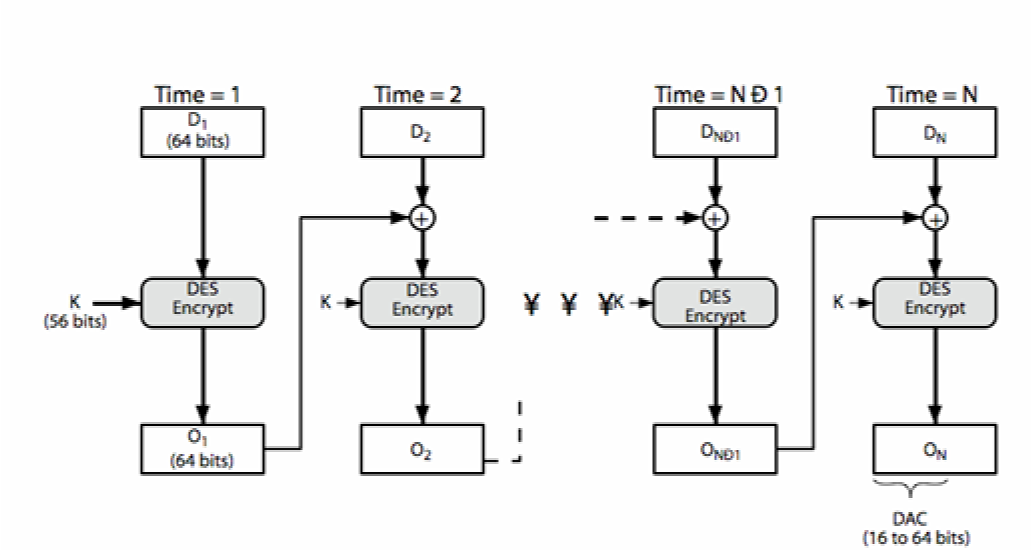
- Taking into account the types of attacks
- Need the mac to satisfy the following:
  1. Knowing a message and mac, is infeasible to find another message with same mac
  2. Macs should be uniformly distributed
  3. Mac should depend equally on all bits of the message

Using symmetric ciphers for macs

- Can use any block cipher chaining mode and use final block as a mac
- **Data authentication algorithm (daa)** is a widely used mac based on des-cbc
  - Using iv=0 and zero-pad of final block
  - Encrypt message using des in cbc mode
  - And send just the final block as the mac
    - Or the leftmost m bits ($16 \leq m \leq 64$) of final block

- But final mac is now too small for security

Data authentication algorithm



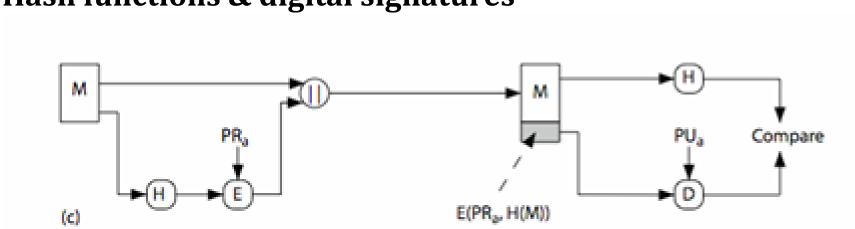
3. Write about the security hash functions in detail.

- Condenses arbitrary message to fixed size

H = h(m)

- Usually assume that the hash function is public and not keyed
  - Cf. Mac which is keyed
- Hash used to detect changes to message
- Can use in various ways with message
- Most often to create a digital signature

Hash functions & digital signatures



Requirements for hash functions

1. Can be applied to any sized message m
2. Produces fixed-length output h
3. Is easy to compute h=h(m) for any message m
4. Given h is infeasible to find x s.t. H(x)=h

- One-way property
5. Given x is infeasible to find y s.t. H(y)=h(x)
    - Weak collision resistance
6. Is infeasible to find any x,y s.t. H(y)=h(x)
    - Strong collision resistance

Simple hash functions

- Are several proposals for simple functions
- Based on xor of message blocks
- Not secure since can manipulate any message and either not change hash or change hash also
- Need a stronger cryptographic function (next chapter)

Birthday attacks

- Might think a 64-bit hash is secure
- But by **birthday paradox** is not
- **Birthday attack** works thus:
    - Opponent generates 2m/2 variations of a valid message all with essentially the same meaning
    - Opponent also generates 2m/2 variations of a desired fraudulent message
    - Two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)
    - Have user sign the valid message, then substitute the forgery which will have a valid signature
- Conclusion is that need to use larger mac/hash

Block ciphers as hash functions

- Can use block ciphers as hash functions
    - Using h0=0 and zero-pad of final block
    - Compute: hi = emi [hi-1]
    - And use final block as the hash value
    - Similar to cbc but without a key
- Resulting hash is too small (64-bit)
    - Both due to direct birthday attack
    - And to "meet-in-the-middle" attack
- Other variants also susceptible to attack

Hash functions & MAC security

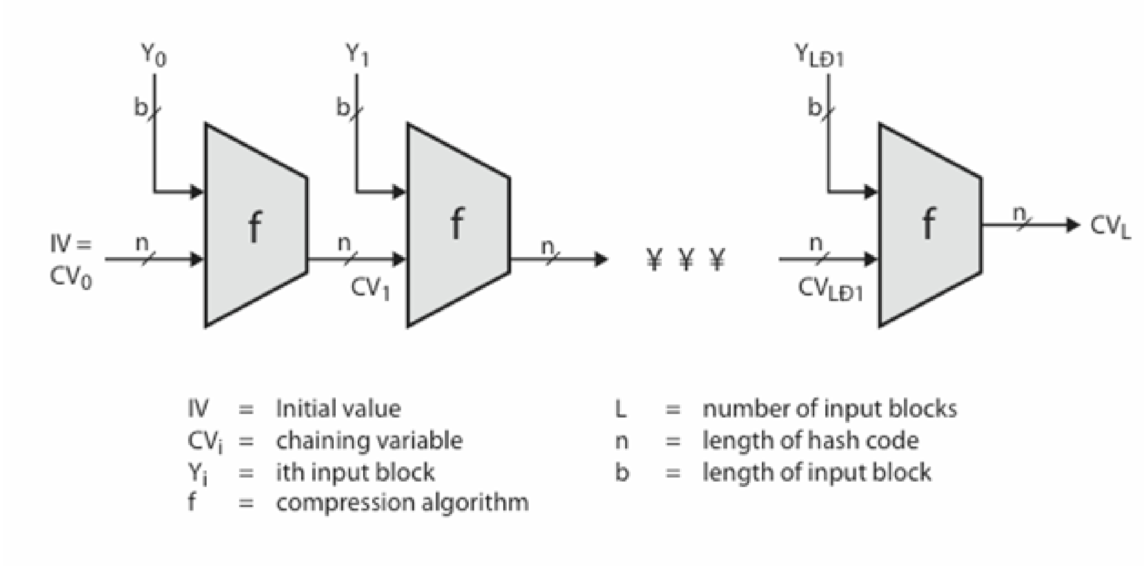
- Like block ciphers have:
- **Brute-force** attacks exploiting
    - Strong collision resistance hash have cost 2m/2
        - Have proposal for h/w md5 cracker
        - 128-bit hash looks vulnerable, 160-bits better
    - Macs with known message-mac pairs

- Can either attack keyspace (cf key search) or mac
    - At least 128-bit mac is needed for security
- **Cryptanalytic attacks** exploit structure
  - Like block ciphers want brute-force attacks to be the best alternative
- Have a number of analytic attacks on iterated hash functions
  - Cvi = f[cvi-1, mi]; h(m)=cvn
  - Typically focus on collisions in function f
  - Like block ciphers is often composed of rounds
  - Attacks exploit properties of round functions

Hash and mac algorithms

- Hash functions
  - Condense arbitrary size message to fixed size
  - By processing message in blocks
  - Through some compression function
  - Either custom or block cipher based
- Message authentication code (mac)
  - Fixed sized authenticator for some message
  - To provide authentication for message
  - By using block cipher mode or hash function

Hash algorithm structure



4. Illustrate Secure Hash algorithm in detail and classify its performance.

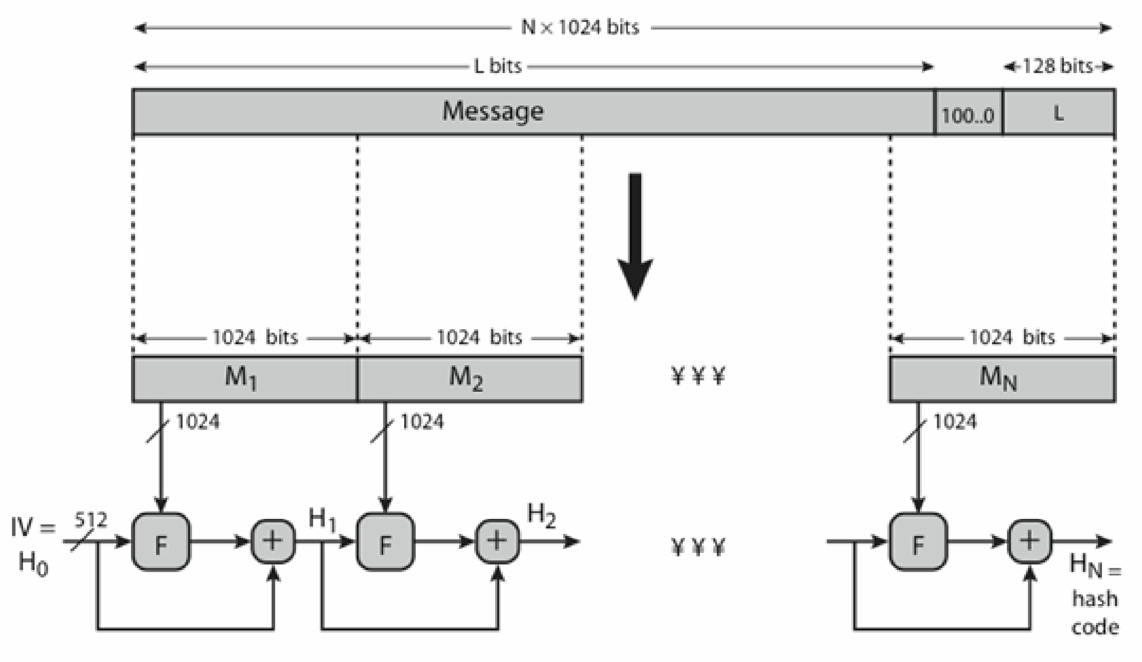
SECURE HASH ALGORITHM

- Sha originally designed by nist & nsa in 1993
- Was revised in 1995 as sha-1
- Us standard for use with dsa signature scheme
  - Standard is fips 180-1 1995, also internet rfc3174
  - Nb. The algorithm is sha, the standard is shs
- Based on design of md4 with key differences
- Produces 160-bit hash values
- Recent 2005 results on security of sha-1 have raised concerns on its use in future applications

Revised secure hash standard

- Nist issued revision fips 180-2 in 2002
- Adds 3 additional versions of sha
  - Sha-256, sha-384, sha-512
- Designed for compatibility with increased security provided by the aes cipher
- Structure & detail is similar to sha-1
- Hence analysis should be similar
- But security levels are rather higher

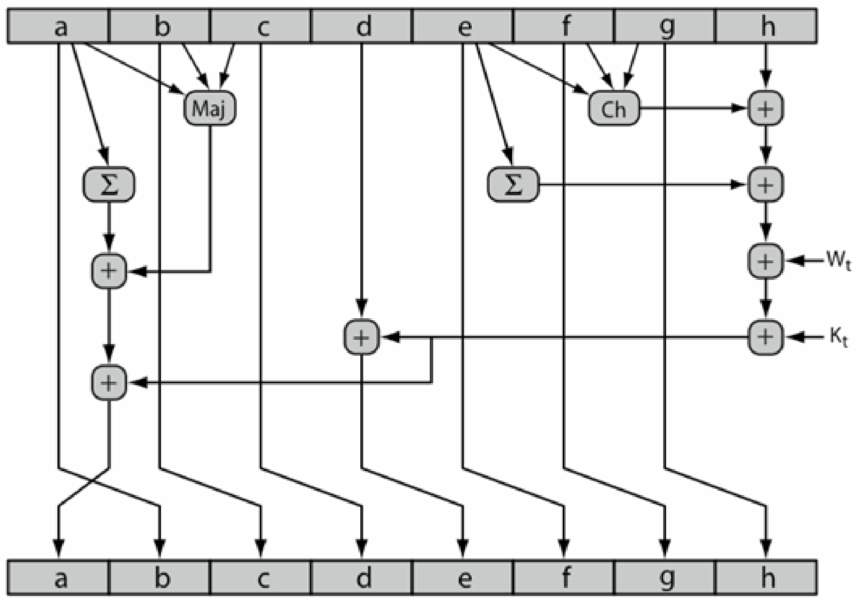
Sha-512 overview

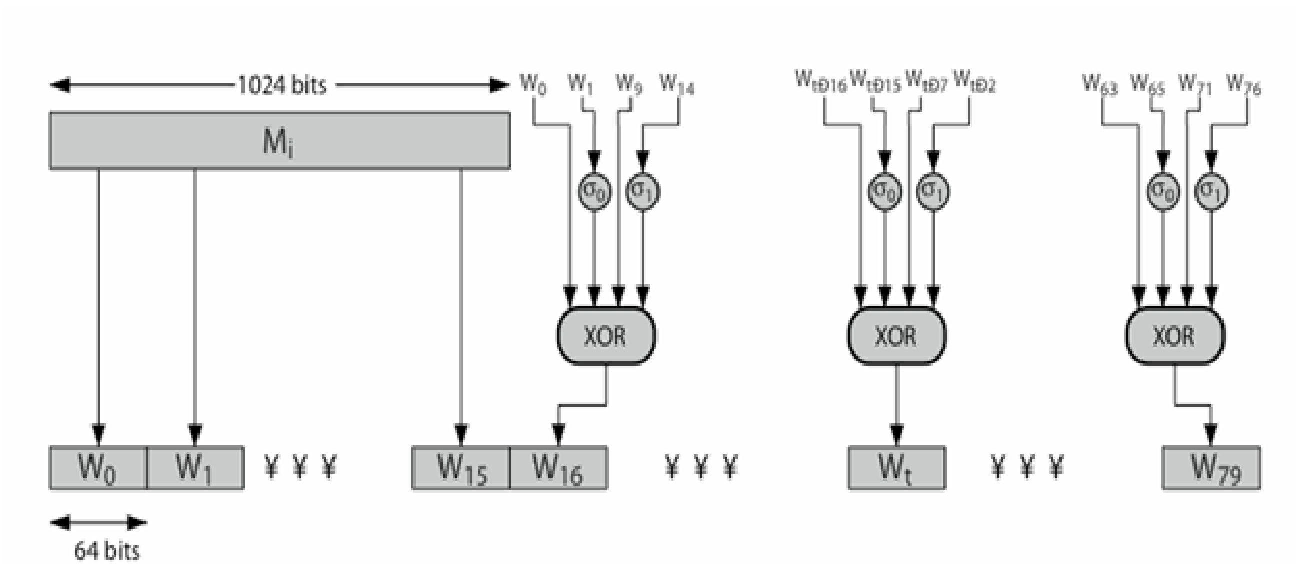


Sha-512 compression function

- Heart of the algorithm
- Processing message in 1024-bit blocks
- Consists of 80 rounds
  - Updating a 512-bit buffer
  - Using a 64-bit value wt derived from the current message block
  - And a round constant based on cube root of first 80 prime numbers

Sha-512 round function





KEYED HASH FUNCTIONS AS MACS

- Want a mac based on a hash function
  - Because hash functions are generally faster
  - Code for crypto hash functions widely available
- Hash includes a key along with message
- Original proposal:

  keyedhash = hash(key|message)
  - Some weaknesses were found with this
- Eventually led to development of hmac

HMAC

- Specified as internet standard rfc2104
- Uses hash function on the message:

Hmack = hash[(k+ xor opad) ||

hash[(k+ xor ipad)||m)]]

- Where k+ is the key padded out to size
- And opad, ipad are specified padding constants
- Overhead is just 3 more hash calculations than the message needs alone
- Any hash function can be used
  - Eg. Md5, sha-1, ripemd-160, whirlpool

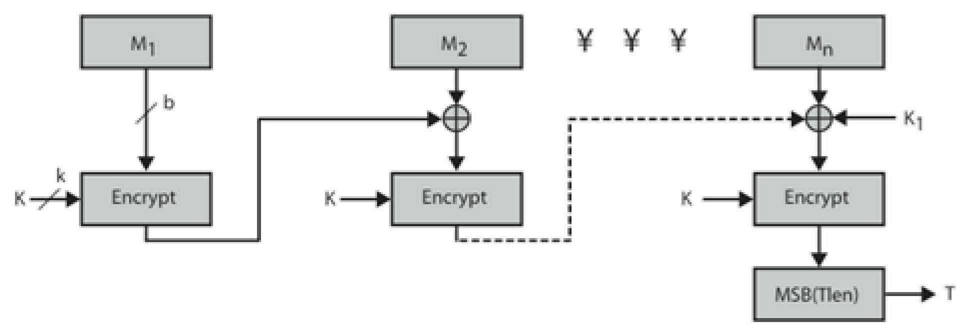
HMAC SECURITY

- Proved security of hmac relates to that of the underlying hash algorithm
- Attacking hmac requires either:
  - Brute force attack on key used
  - Birthday attack (but since keyed would need to observe a very large number of messages)
- Choose hash function used based on speed verses security constraints

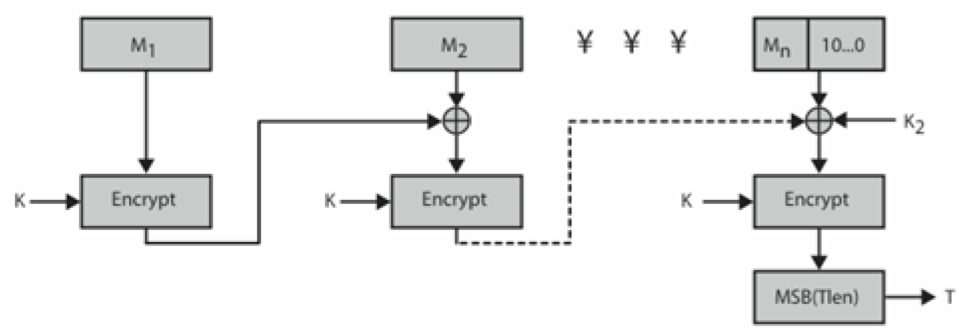
CMAC

- Previously saw the daa (cbc-mac)
- Widely used in govt & industry
- But has message size limitation
- Can overcome using 2 keys & padding
- Thus forming the cipher-based message authentication code (cmac)
- Adopted by nist sp800-38b

CMAC OVERVIEW



(a) Message length is integer multiple of block size



(b) Message length is not integer multiple of block size

RIPEMD-160

- ➢ Ripemd-160 was developed in europe as part of ripe project in 96
- ➢ By researchers involved in attacks on md4/5
- ➢ Initial proposal strengthen following analysis to become ripemd-160
- ➢ Somewhat similar to md5/sha
- ➢ Uses 2 parallel lines of 5 rounds of 16 steps
- ➢ Creates a 160-bit hash value
- ➢ Slower, but probably more secure, than sha

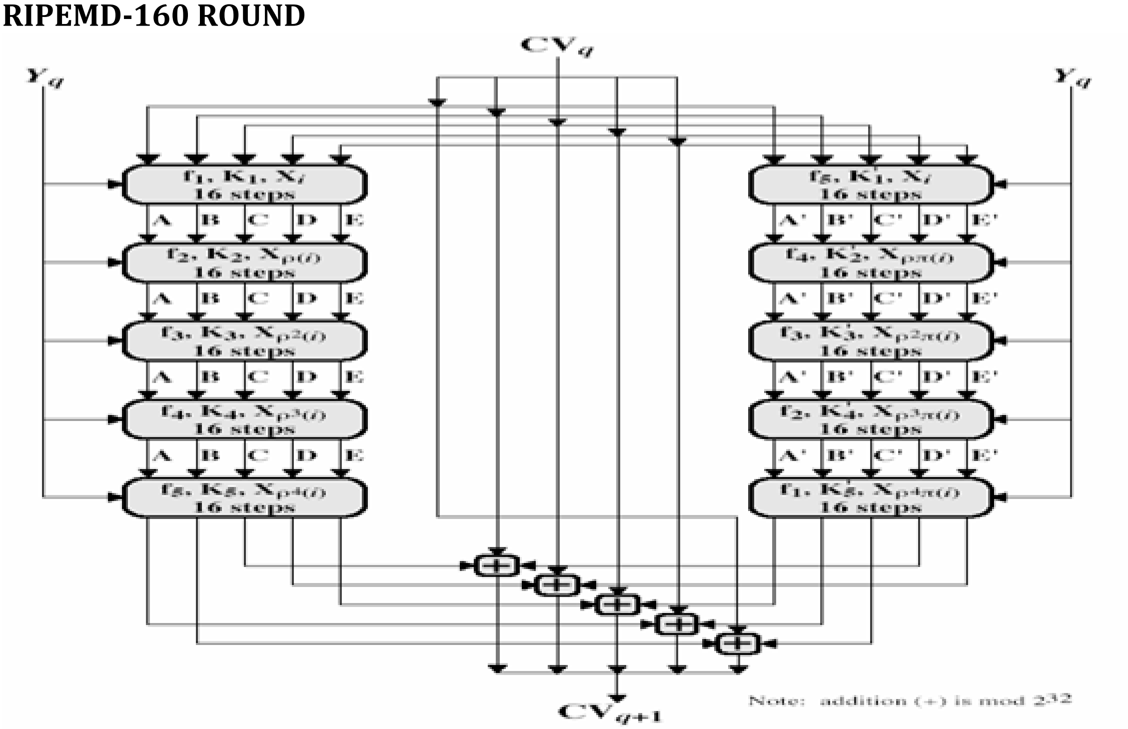
RIPEMD-160 OVERVIEW

1. Pad message so its length is 448 mod 512
2. Append a 64-bit length value to message
3. Initialise 5-word (160-bit) buffer (a,b,c,d,e) to

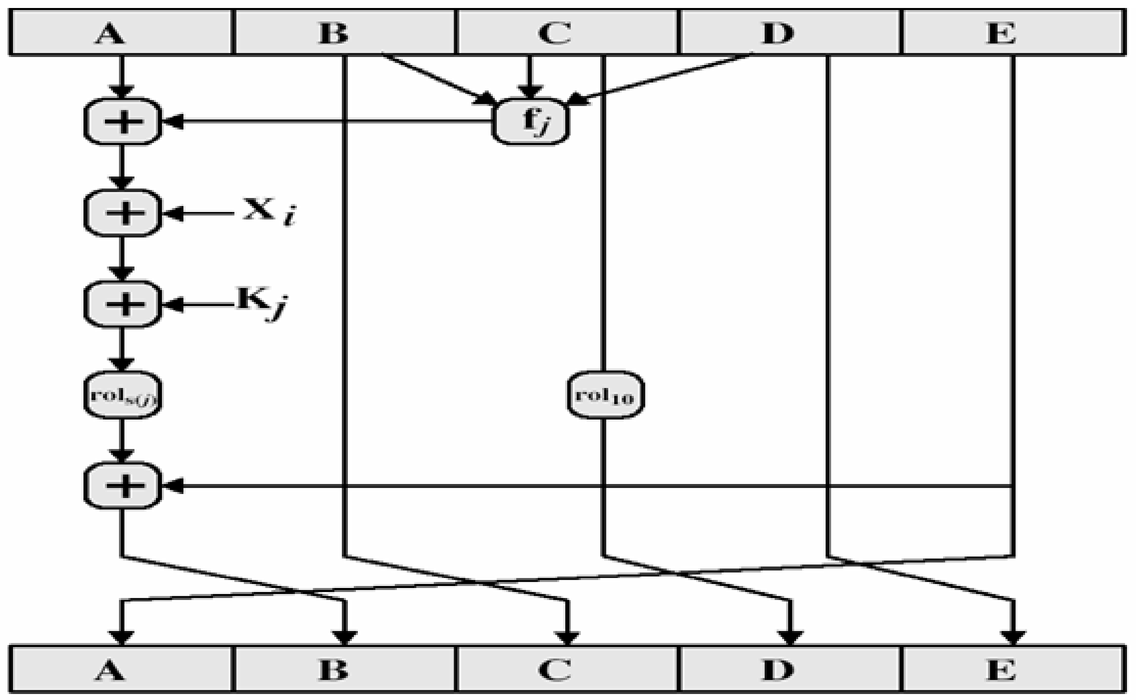
(67452301,efcdab89,98badcfe,10325476,c3d2e1f0)

1. Process message in 16-word (512-bit) chunks:
   - Use 10 rounds of 16 bit operations on message block & buffer – in 2 parallel lines of 5
   - Add output to input to form new buffer value
2. Output hash value is the final buffer value

RIPEMD-160 ROUND



RIPEMD-160 COMPRESSION FUNCTION



RIPEMD-160 design criteria

- Use 2 parallel lines of 5 rounds for increased complexity
- For simplicity the 2 lines are very similar
- Step operation very close to md5
- Permutation varies parts of message used
- Circular shifts designed for best results

RIPEMD-160 verses md5 & sha-1

- Brute force attack harder (160 like sha-1 vs 128 bits for md5)
- Not vulnerable to known attacks, like sha-1 though stronger (compared to md4/5)
- Slower than md5 (more steps)
- All designed as simple and compact
- Sha-1 optimised for big endian cpu's vs ripemd-160 & md5 optimised for little endian cpu's

5. Describe Digital Signature standard and authentication protocols.

Digital signatures

- Have looked at message authentication
  - But does not address issues of lack of trust
- Digital signatures provide the ability to:
  - Verify author, date & time of signature
  - Authenticate message contents
  - Be verified by third parties to resolve disputes
- Hence include authentication function with additional capabilities

Digital signature properties

- Must depend on the message signed
- Must use information unique to sender
  - To prevent both forgery and denial
- Must be relatively easy to produce
- Must be relatively easy to recognize & verify
- Be computationally infeasible to forge
  - With new message for existing digital signature
  - With fraudulent digital signature for given message
- Be practical save digital signature in storage

Direct digital signatures

- Involve only sender & receiver
- Assumed receiver has sender's public-key
- Digital signature made by sender signing entire message or hash with private-key
- Can encrypt using receivers public-key
- Important that sign first then encrypt message & signature
- Security depends on sender's private-key

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

<u>AUTHENTICATION PROTOCOLS</u>

- 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

Replay attacks

- Where a valid signed message is copied and later resent
    - Simple replay
    - Repetition that can be logged
    - Repetition that cannot be detected
    - Backward replay without modification
- Countermeasures include
    - Use of sequence numbers (generally impractical)
    - Timestamps (needs synchronized clocks)
    - Challenge/response (using unique nonce)

Using symmetric encryption

- As discussed previously can use a two-level hierarchy of keys
- Usually with a trusted key distribution center (kdc)
    - Each party shares own master key with kdc
    - Kdc generates session keys used for connections between parties
    - Master keys used to distribute these to them

Using public-key encryption

- Have a range of approaches based on the use of public-key encryption
- Need to ensure have correct public keys for other parties
- Using a central authentication server (as)
- Various protocols exist using timestamps or nonces

One-way authentication

- Required when sender & receiver are not in communications at same time (eg. Email)
- Have header in clear so can be delivered by email system
- May want contents of body protected & sender authenticated

Using symmetric encryption

- Can refine use of kdc but can't have final exchange of nonces, vis:

1. A->kdc: *ida* || *idb* || *n1*

2. Kdc -> a: eka[ks || *idb* || *n1* || e*kb*[*ks*||*ida*]]

3. A -> b: *ekb*[*ks*||*ida*] || eks[m]

- does not protect against replays
    - Could rely on timestamp in message, though email delays make this problematic

Public-key approaches

- Have seen some public-key approaches
- If confidentiality is major concern, can use:

A->b: epub[ks] || eks[m]

- Has encrypted session key, encrypted message
- If authentication needed use a digital signature with a digital certificate:

A->b: m || epra[h(m)] || epras[t||ida||pua]

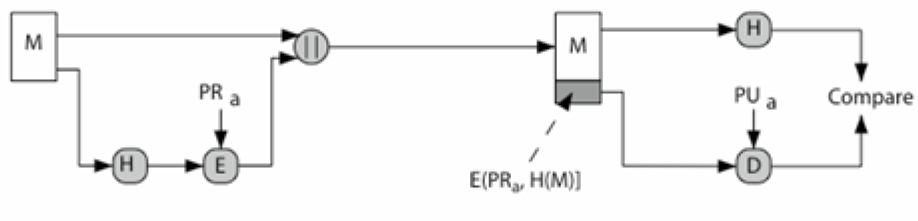
  - With message, signature, certificate

6. Briefly Explain about Digital signature algorithm

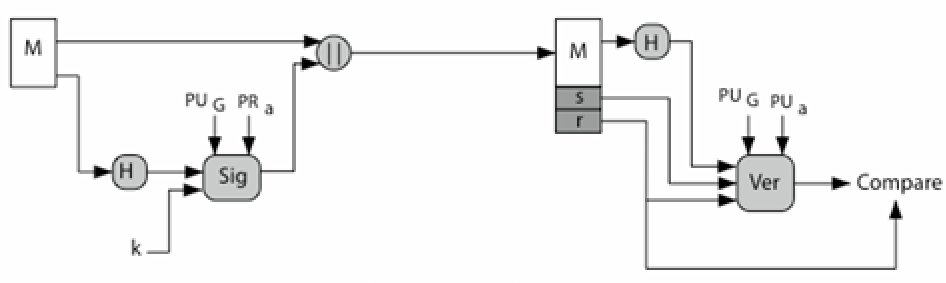
- Us govt approved signature scheme
- Designed by nist & nsa in early 90's
- Published as fips-186 in 1991
- Revised in 1993, 1996 & then 2000
- Uses the sha hash algorithm
- Dss is the standard, dsa is the algorithm
- Fips 186-2 (2000) includes alternative rsa & elliptic curve signature variants

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



(a) RSA Approach



(b) DSS Approach

DSA key generation

- Have shared global public key values (p,q,g):
  - Choose q, a 160 bit
  - Choose a large prime p = 2l
    - Where l= 512 to 1024 bits and is a multiple of 64
    - And q is a prime factor of (p-1)
  - Choose g = h(p-1)/q

- Where h<p-1, h(p-1)/q (mod p) > 1
- Users choose private & compute public key:
  - Choose x<q
  - Compute y = gx (mod p)

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 = (gk(mod p))(mod q)

S = (k-1.h(m)+ x.r)(mod q)

- Sends signature (r,s) with message m

DSA signature verification

- Having received m & signature (r,s)
- To **verify** a signature, recipient computes:

W = s-1(mod q)

U1= (h(m).w)(mod q)

U2= (r.w)(mod q)

V = (gu1.yu2(mod p)) (mod q)

- If v=r then signature is verified
- See book web site for details of proof why

UNIT IV
SECURITY PRACTICE & SYSTEM SECURITY

1. Elaborately explain Kerberos authentication mechanism with suitable diagrams.

KERBEROS

- Trusted key server system from mit
- Provides centralised private-key third-party authentication in a distributed network
  - Allows users access to services distributed through network
  - Without needing to trust all workstations
  - Rather all trust a central authentication server
- Two versions in use: 4 & 5

Kerberos requirements

- Its first report identified requirements as:
  - Secure
  - Reliable
  - Transparent
  - Scalable
- Implemented using an authentication protocol based on needham-schroeder

Simple authentication

- C->as : idc || pc || idv
- As->c : ticket
- C->v : idc||ticket
- Ticket=e(kv, [idc || adc|| idv])

- What adc plays here?

A more secure authentication

- Problem to be addressed
- 1. Repeated password requirement
- 2. Capture passwords, ie plain msg pwd.

- To solve this kerberos introduced tgs concept.

Kerberos v4

- 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

Kerberos v4 dialogue

1. Obtain ticket granting ticket from as
    - Once per session
2. Obtain service granting ticket from tgt
    - For each distinct service required
3. Client/server exchange to obtain service
    - On every service request

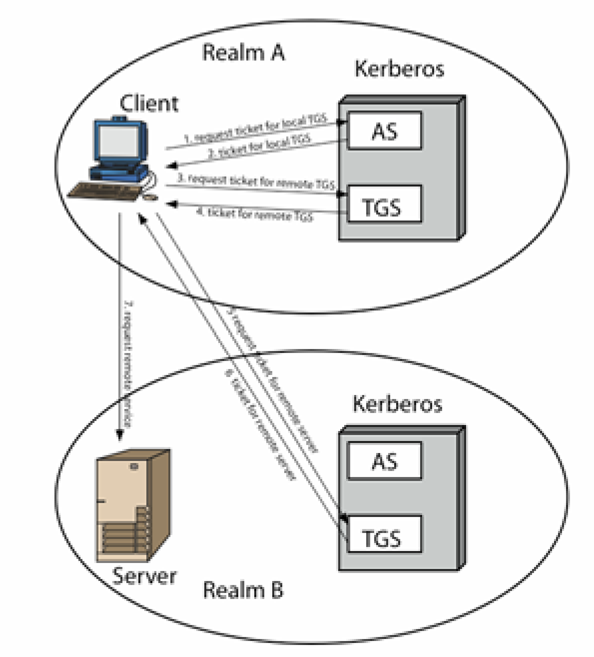
Dialogues

- C->as: idc || idtgs
- As->c: e(kc,ticket tgs)
- C->tgs: idc||idv||ticket tgs
- Tgs->c: ticket v
- C->v: idc||ticket v
- Ticket tgs =e(ktgs,[idc||adc||idtgs||ts1||lt1])
- Ticket v = e(kv,[idc||adc||idv||ts2||lt2])

Kerberos realms

- A kerberos environment consists of:
    - A kerberos server
    - A number of clients, all registered with server
    - Application servers, sharing keys with server

- This is termed a realm
    - Typically a single administrative domain
- If have multiple realms, their kerberos servers must share keys and trust



Kerberos version 5

- Developed in mid 1990's
- Specified as internet standard rfc 1510
- Provides improvements over v4
    - Addresses environmental shortcomings
        - Encryption alg, network protocol, byte order, ticket lifetime, authentication forwarding, interrealm auth
    - And technical deficiencies
        - Double encryption, non-std mode of use, session keys, password attacks

Environmental shortcomings

Encryption Algorithm:

1. Can use any algorithm.
2. V4 uses DES algorithm.

- Internet protocol dependence:

v4 uses IP address, ISO network address was not adopted. V5 uses any network address type.

Message byte ordering:

Msg byte ordering done by the sender, v5 uses ans.1 and ber ie no ambiguous byte ordering.

Ticket life time:

lt can be expressed in 8bit quantity of five minutes. Ie max 21 hrs can be expressed. v5 uses arbitrary lt.

Authentication fwd:
No credential fwds to others, v5 supports.

Inter-realm authentication:
Handled in v5 better than v4.

Comparing the dlg of k4 & k5

- Msg->1 :
- 1. Realm: indicates the realm of the user
  2. Options: used to request certain flags be set in the returned ticket.
  3. Times: used by the client to request the following times in the tickets.
  From: start time of validation of ticket
  Till: time period.
  Rtime: renew till time.
  4. Nonce: to stop replay attack.
- Msg 5/ 6:

1. Subkey: to protect this application session by using a specific key. If omitted then kc,v is assumed as session key.

2. Sequence number: optional field to specify the sequence number.

Some ticket flags

- **Renewable**
  - Long lived tickets are risky (may be stolen and the opponent use until the expiration time)
  - Short lived ones cause protocol overheads
    - For tgt, the user should enter password for each ticket
  - Solution: ticket originally has short lifetime, but can be periodically (and automatically) renewed
    - Until *renew-till* time specified in the ticket
    - Unless tgs or as refuses to renew it (if stolen)

- ***Proxiable / proxy***
  - If a tgt is *proxiable*, then tgs may issue *proxy* tickets that the ticket owner (say alice) may give some other servers that may act on behalf of alice
- ***Forwardable / forwarded***
  - More powerful than proxy
    - Proxy flag can be set only in server tickets
    - Forwarded flag can be set also in tgts
  - If a tgt bears a *forwardable* flag set, than tgs may issue *forwarded* tgts for a nearby realm
    - Nearby realm's tgs may either forward or issue a server ticket.

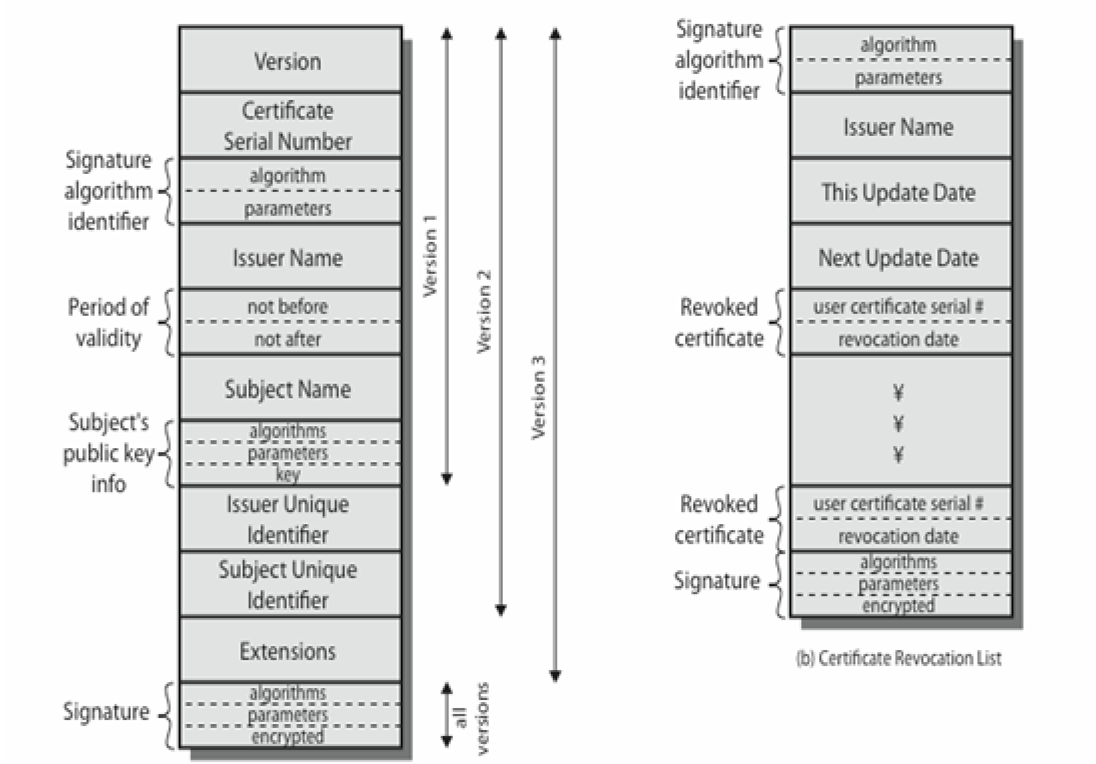
- In this way, realms can be connected

2. Give a brief notes on X.509 Authentication Service

- Part of ccitt x.500 directory service standards
    - Distributed servers maintaining user info database
- Defines framework for authentication services
    - Directory may store public-key certificates
    - With public key of user signed by certification authority
- Also defines authentication protocols
- Uses public-key crypto & digital signatures
    - Algorithms not standardised, but rsa recommended
- X.509 certificates are widely used

X.509 certificates

- Issued by a certification authority (ca), containing:
    - Version (1, 2, or 3)
    - Serial number (unique within ca) identifying certificate
    - Signature algorithm identifier
    - Issuer x.500 name (ca)
    - Period of validity (from - to dates)
    - Subject x.500 name (name of owner)
    - Subject public-key info (algorithm, parameters, key)
    - Issuer unique identifier (v2+)
    - Subject unique identifier (v2+)
    - Extension fields (v3)
    - Signature (of hash of all fields in certificate)
- Notation ca<<a>> denotes certificate for a signed by ca



(b) Certificate Revocation List

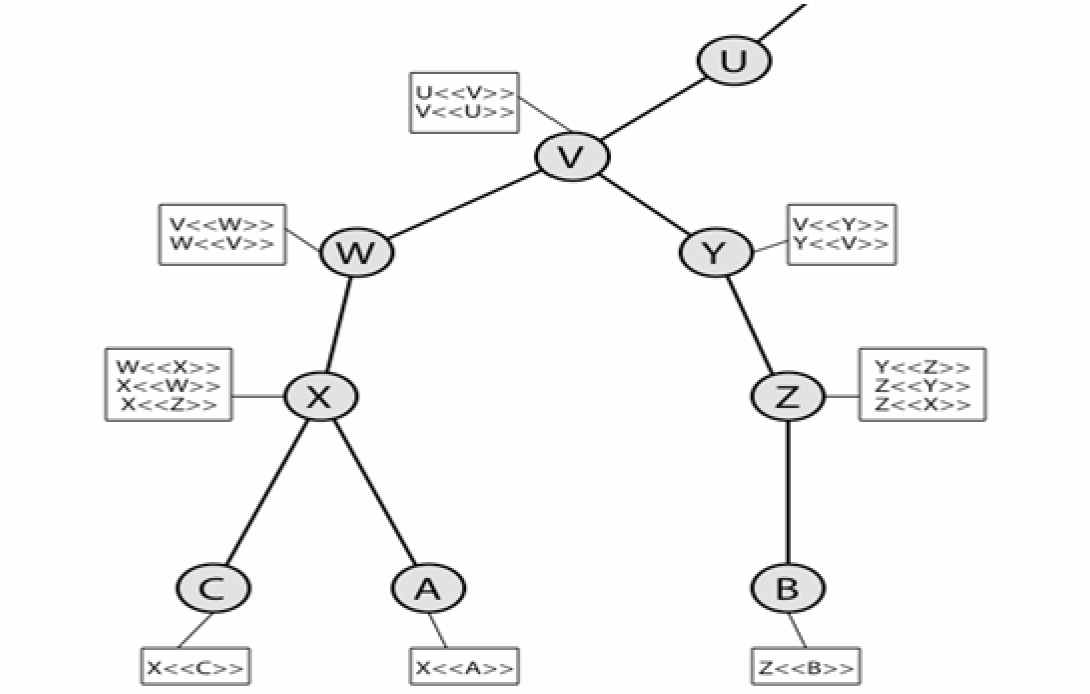
Obtaining a certificate

- Any user with access to ca can get any certificate from it
- Only the ca can modify a certificate
- Because cannot be forged, certificates can be placed in a public directory

CA hierarchy

- If both users share a common ca then they are assumed to know its public key
- Otherwise ca's must form a hierarchy
- Use certificates linking members of hierarchy to validate other ca's
  - Each ca has certificates for clients (forward) and parent (backward)
- Each client trusts parents certificates
- Enable verification of any certificate from one ca by users of all other cas in hierarchy

CA hierarchy use



Certificate revocation

- Certificates have a period of validity
- May need to revoke before expiry, eg:
    1. User's private key is compromised
    2. User is no longer certified by this ca
    3. Ca's certificate is compromised
- Ca's maintain list of revoked certificates
    1. The certificate revocation list (crl)
- Users should check certificates with ca's crl

Authentication procedures

- X.509 includes three alternative authentication procedures:
- One-way authentication
- Two-way authentication
- Three-way authentication
- All use public-key signatures

One-way authentication

- 1 message (a->b) used to establish
    - The identity of a and that message is from a
    - Message was intended for b
    - Integrity & originality of message
- Message must include timestamp, nonce, b's identity and is signed by a
- May include additional info for b

        - Eg session key

Two-way authentication

- 2 messages (a->b, b->a) which also establishes in addition:
    - The identity of b and that reply is from b
    - That reply is intended for a
    - Integrity & originality of reply
- Reply includes original nonce from a, also timestamp and nonce from b
- May include additional info for a

Three-way authentication

- 3 messages (a->b, b->a, a->b) which enables above authentication without synchronized clocks
- Has reply from a back to b containing signed copy of nonce from b
- Means that timestamps need not be checked or relied upon

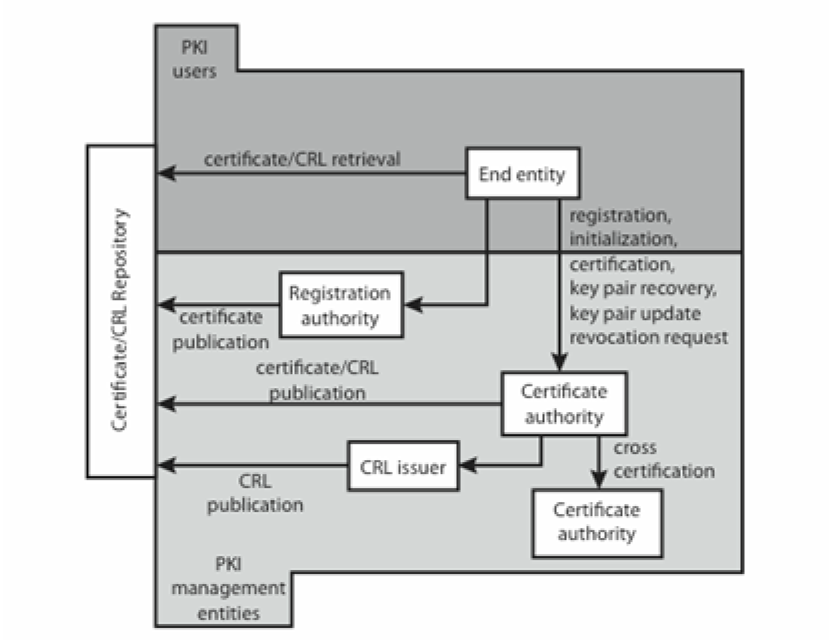
X.509 VERSION 3

- Has been recognised that additional information is needed in a certificate
    - Email/url, policy details, usage constraints
- Rather than explicitly naming new fields defined a general extension method
- Extensions consist of:
    - Extension identifier
    - Criticality indicator
    - Extension value

Certificate extensions

- Key and policy information
    - Convey info about subject & issuer keys, plus indicators of certificate policy
- Certificate subject and issuer attributes
    - Support alternative names, in alternative formats for certificate subject and/or issuer
- Certificate path constraints
    - Allow constraints on use of certificates by other ca’s

Public key infrastructure



ELECTRONIC MAIL SECURITY

- Email is one of the most widely used and regarded network services
- Currently message contents are not secure
  - May be inspected either in transit
  - Or by suitably privileged users on destination system

Email security enhancements

- Confidentiality
  - Protection from disclosure
- Authentication
  - Of sender of message
- Message integrity
  - Protection from modification
- Non-repudiation of origin
  - Protection from denial by sender

3. Explain Pretty Good Privacy in detail

PRETTY GOOD PRIVACY (PGP)

- Widely used de facto secure email
- Developed by phil zimmermann
- Selected best available crypto algs to use
- Integrated into a single program
- On unix, pc, macintosh and other systems
- Originally free, now also have commercial versions available

PGP operation – authentication

1. Sender creates message

2. Use sha-1 to generate 160-bit hash of message
3. Signed hash with rsa using sender's private key, and is attached to message
4. Receiver uses rsa with sender's public key to decrypt and recover hash code
5. Receiver verifies received message using hash of it and compares with decrypted hash code

PGP operation – confidentiality

1. Sender generates message and 128-bit random number as session key for it
2. Encrypt message using cast-128 / idea / 3des in cbc mode with session key
3. Session key encrypted using rsa with recipient's public key, & attached to msg
4. Receiver uses rsa with private key to decrypt and recover session key
5. Session key is used to decrypt message

PGP operation – confidentiality & authentication

- Can use both services on same message
  - Create signature & attach to message
  - Encrypt both message & signature
  - Attach rsa/elgamal encrypted session key

PGP operation – compression

- By default pgp compresses message after signing but before encrypting
  - So can store uncompressed message & signature for later verification
  - & because compression is non deterministic
- Uses zip compression algorithm

PGP operation – email compatibility

- When using pgp will have binary data to send (encrypted message etc)
- However email was designed only for text
- Hence pgp must encode raw binary data into printable ascii characters
- Uses radix-64 algorithm
  - Maps 3 bytes to 4 printable chars
  - Also appends a crc
- Pgp also segments messages if too big

PGP session keys

- Need a session key for each message
  - Of varying sizes: 56-bit des, 128-bit cast or idea, 168-bit triple-des
- Generated using ansi x12.17 mode
- Uses random inputs taken from previous uses and from keystroke timing of user

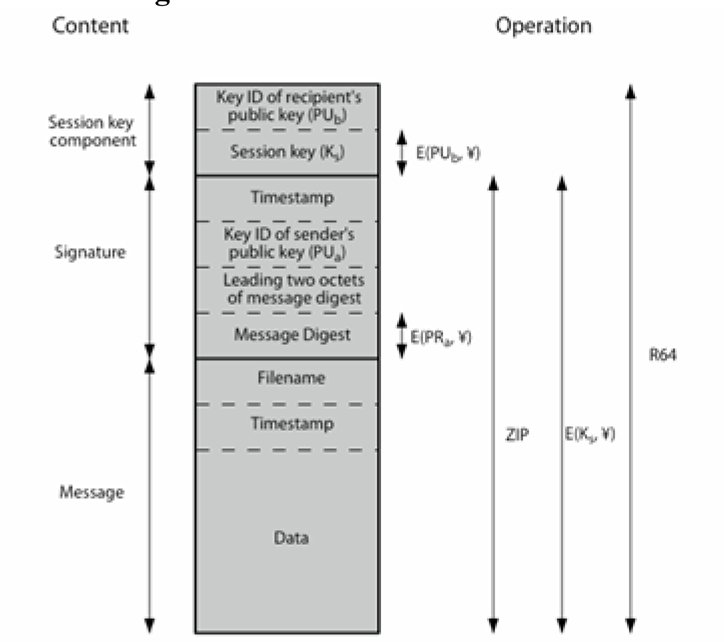
PGP public & private keys

- Since many public/private keys may be in use, need to identify which is actually used to encrypt session key in a message
  - Could send full public-key with every message
  - But this is inefficient
- Rather use a key identifier based on key

- Is least significant 64-bits of the key
- Will very likely be unique

➢ Also use key id in signatures

PGP message format

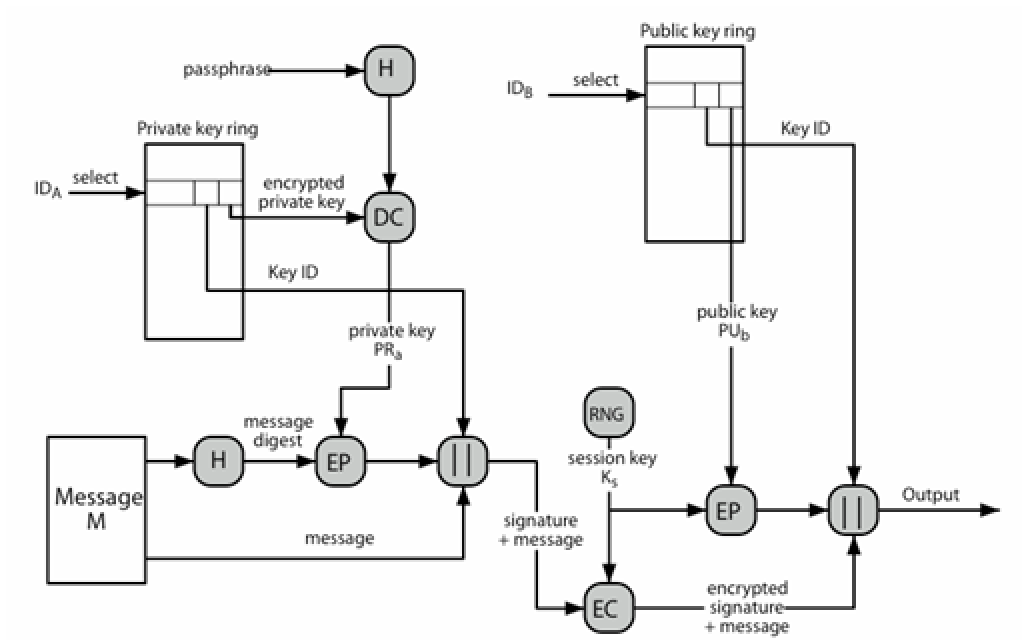


PGP key rings

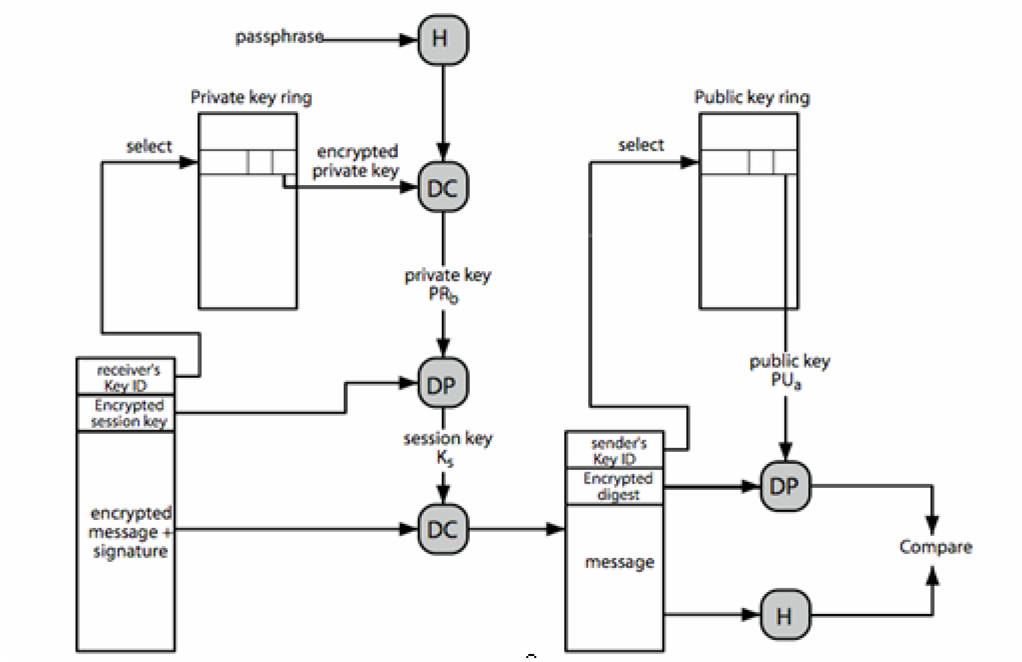
➢ Each pgp user has a pair of keyrings:
- Public-key ring contains all the public-keys of other pgp users known to this user, indexed by key id
- Private-key ring contains the public/private key pair(s) for this user, indexed by key id & encrypted keyed from a hashed passphrase

➢ Security of private keys thus depends on the pass-phrase security

PGP message generation



PGP message reception



PGP key management

- Rather than relying on certificate authorities
- In pgp every user is own ca
  - Can sign keys for users they know directly
- Forms a "web of trust"
  - Trust keys have signed
  - Can trust keys others have signed if have a chain of signatures to them
- Key ring includes trust indicators
- Users can also revoke their keys

4. Describe Secure Multi Purpose Internet Mail Extentions.

S/MIME (SECURE/MULTIPURPOSE INTERNET MAIL EXTENSIONS)

- Security enhancement to mime email
  - Original internet rfc822 email was text only
  - Mime provided support for varying content types and multi-part messages
  - With encoding of binary data to textual form
  - S/mime added security enhancements
- Have s/mime support in many mail agents
  - Eg ms outlook, mozilla, mac mail etc

S/MIME functions

- Enveloped data
  - Encrypted content and associated keys
- Signed data
  - Encoded message + signed digest
- Clear-signed data
  - Cleartext message + encoded signed digest
- Signed & enveloped data
  - Nesting of signed & encrypted entities

S/MIME cryptographic algorithms

- Digital signatures: dss & rsa
- Hash functions: sha-1 & md5
- Session key encryption: elgamal & rsa
- Message encryption: aes, triple-des, rc2/40 and others
- Mac: hmac with sha-1
- Have process to decide which algs to use

S/MIME messages

- S/mime secures a mime entity with a signature, encryption, or both
- Forming a mime wrapped pkcs object
- Have a range of content-types:
  - Enveloped data
  - Signed data
  - Clear-signed data
  - Registration request
  - Certificate only message

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

Certificate authorities

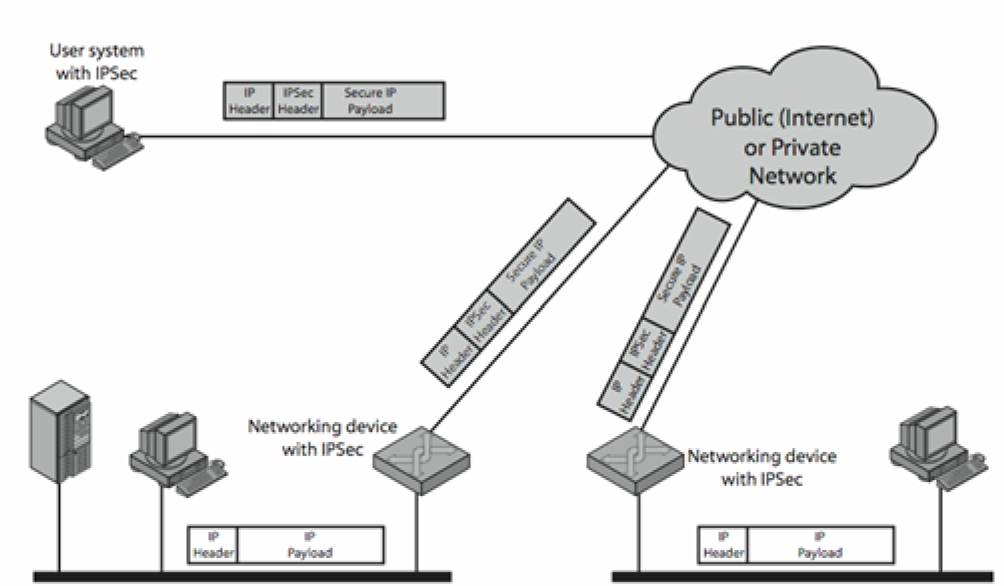
- Have several well-known ca's
- Verisign one of most widely used
- Verisign issues several types of digital ids
- Increasing levels of checks & hence trust

Class	**identity checks**	**usage**
1	name/email check	web browsing/email
2	enroll/addr check	email, subs, s/w validate
3	id documents	e-banking/service access

<u>IP SECURITY</u>

- Have a range of application specific security mechanisms
  - Eg. S/mime, pgp, kerberos, ssl/https
- However there are security concerns that cut across protocol layers
- Would like security implemented by the network for all applications
- General ip security mechanisms
- Provides
  - Authentication
  - Confidentiality
  - Key management
- Applicable to use over lans, across public & private wans, & for the internet

IPSEC uses



Benefits of IPSEC

- In a firewall/router provides strong security to all traffic crossing the perimeter
- In a firewall/router is resistant to bypass
- Is below transport layer, hence transparent to applications
- Can be transparent to end users
- Can provide security for individual users
- Secures routing architecture

IP security architecture

- Specification is quite complex
- Defined in numerous rfc's
  - Incl. Rfc 2401/2402/2406/2408
  - Many others, grouped by category
- Mandatory in ipv6, optional in ipv4
- Have two security header extensions:
  - Authentication header (ah)
  - Encapsulating security payload (esp)

IPSEC services

- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets
  - A form of partial sequence integrity
- Confidentiality (encryption)
- Limited traffic flow confidentiality

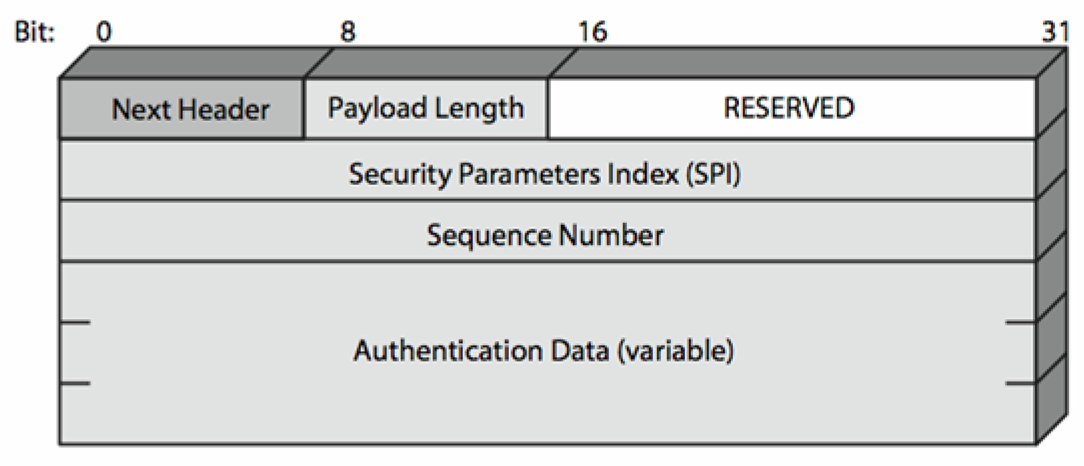
Security associations

- A one-way relationship between sender & receiver that affords security for traffic flow
- Defined by 3 parameters:
  - Security parameters index (spi)
  - Ip destination address
  - Security protocol identifier
- Has a number of other parameters
  - Seq no, ah & eh info, lifetime etc
- Have a database of security associations

Authentication header (AH)

- Provides support for data integrity & authentication of ip packets
  - End system/router can authenticate user/app
  - Prevents address spoofing attacks by tracking sequence numbers
- Based on use of a mac
  - Hmac-md5-96 or hmac-sha-1-96
- Parties must share a secret key

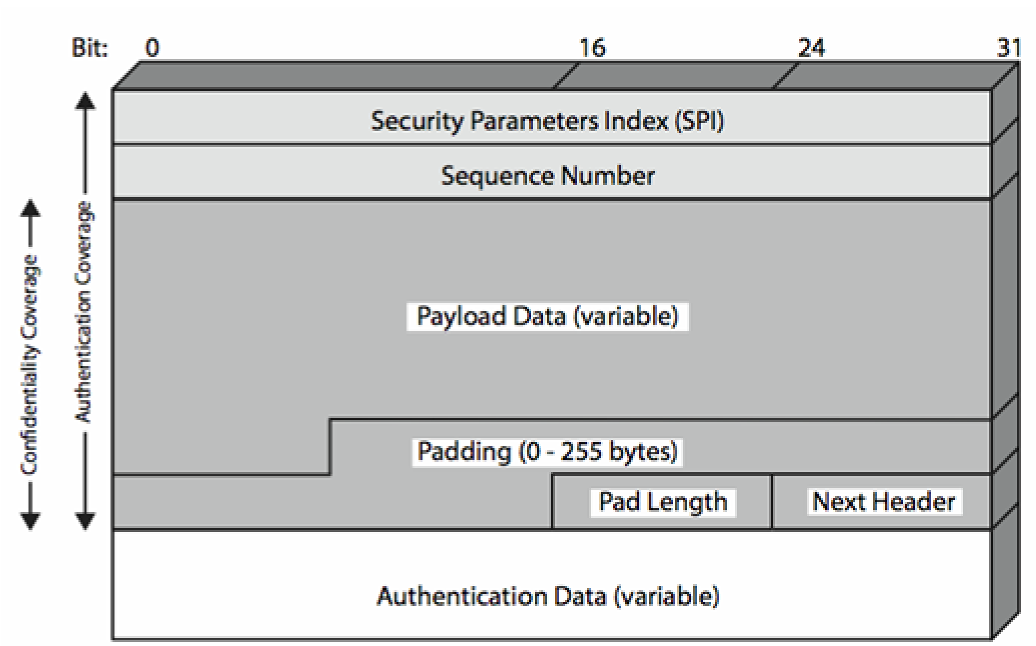
Authentication header



Encapsulating security payload (ESP)

- Provides message content confidentiality & limited traffic flow confidentiality
- Can optionally provide the same authentication services as ah
- Supports range of ciphers, modes, padding
  - Incl. Des, triple-des, rc5, idea, cast etc
  - Cbc & other modes
  - Padding needed to fill blocksize, fields, for traffic flow

Encapsulating security payload



Transport vs tunnel mode ESP

- Transport mode is used to encrypt & optionally authenticate ip data
  - Data protected but header left in clear
  - Can do traffic analysis but is efficient
  - Good for esp host to host traffic
- Tunnel mode encrypts entire ip packet
  - Add new header for next hop
  - Good for vpns, gateway to gateway security

Combining security associations

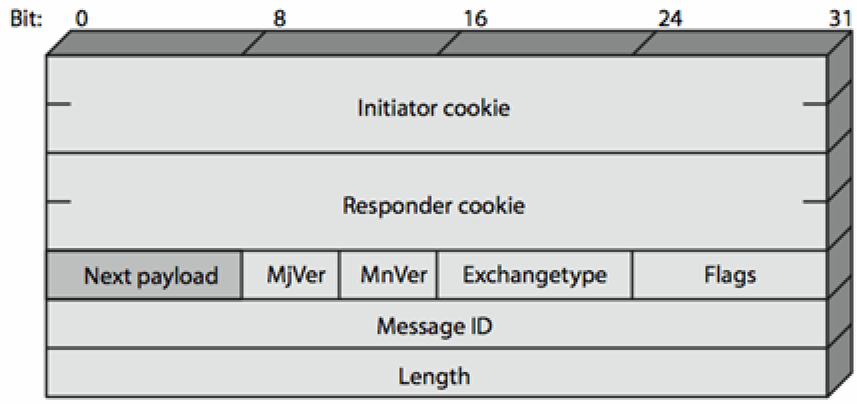
- Sa's can implement either ah or esp
- To implement both need to combine sa's
  - Form a security association bundle
  - May terminate at different or same endpoints
  - Combined by
    - Transport adjacency
    - Iterated tunneling
- Issue of authentication & encryption order

Oakley

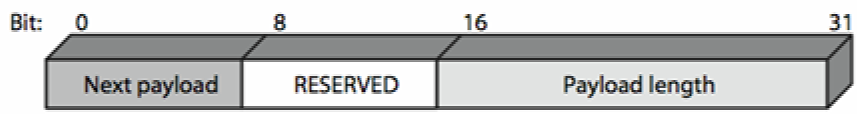
- A key exchange protocol
- Based on diffie-hellman key exchange
- Adds features to address weaknesses
  - Cookies, groups (global params), nonces, dh key exchange with authentication
- Can use arithmetic in prime fields or elliptic curve fields

ISAKMP

- Internet security association and key management protocol
- Provides framework for key management
- Defines procedures and packet formats to establish, negotiate, modify, & delete sas
- Independent of key exchange protocol, encryption alg, & authentication method



(a) ISAKMP Header



(b) Generic Payload Header

ISAKMP payloads & exchanges

- Have a number of isakmp payload types:
  - Security, proposal, transform, key, identification, certificate, certificate, hash, signature, nonce, notification, delete
- isakmp has framework for 5 types of message exchanges:
  - Base, identity protection, authentication only, aggressive, informational

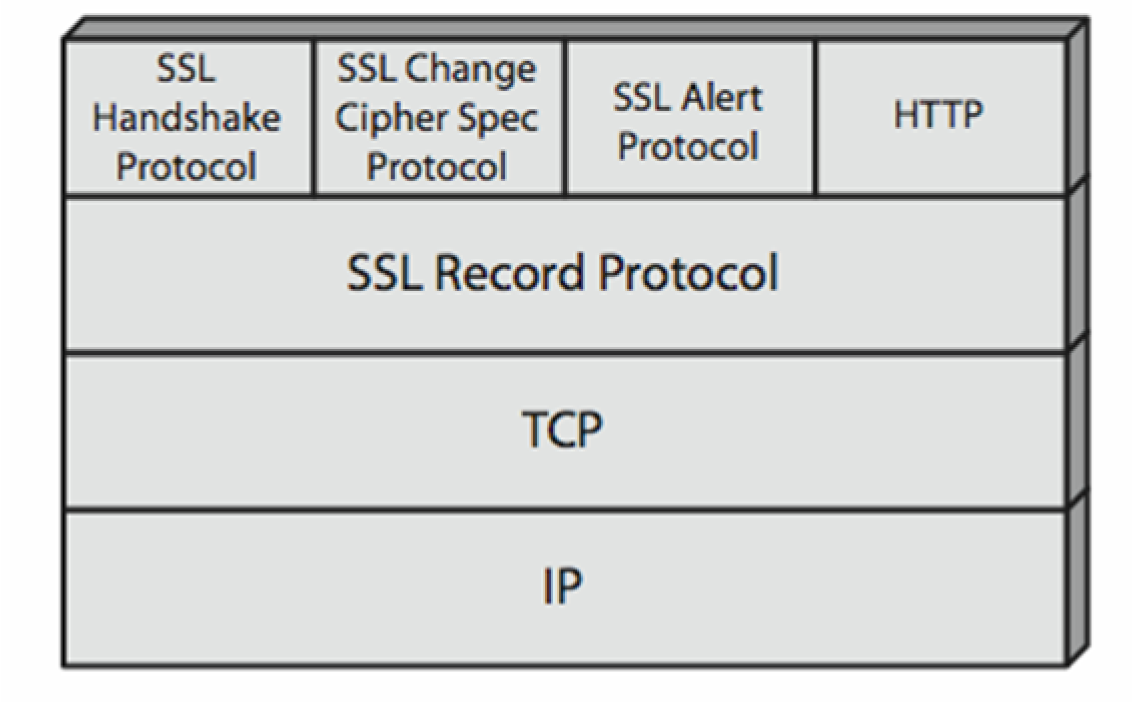
4.8WEB SECURITY

- Web now widely used by business, government, individuals
- But internet & web are vulnerable
- Have a variety of threats
  - Integrity
  - Confidentiality
  - Denial of service
  - Authentication
- Need added security mechanisms

SSL (SECURE SOCKET LAYER)

- Transport layer security service
- Originally developed by netscape
- Version 3 designed with public input
- Subsequently became internet standard known as tls (transport layer security)
- Uses tcp to provide a reliable end-to-end service
- Ssl has two layers of protocols

SSL architecture



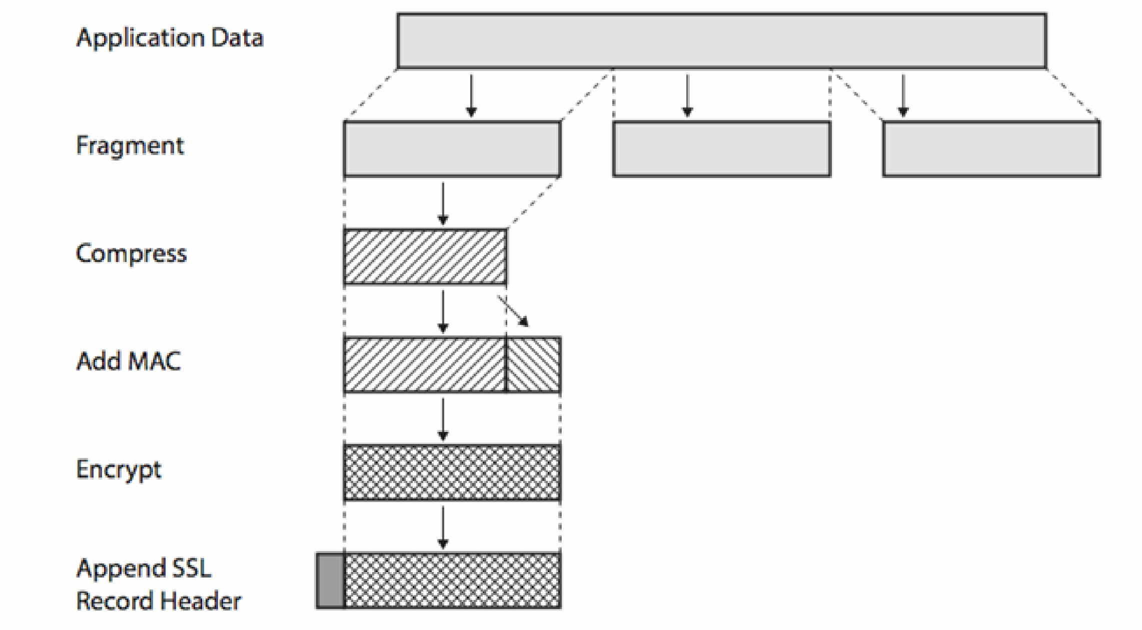
- **SSL Connection**
  - A transient, peer-to-peer, communications link
  - Associated with 1 ssl session
- **SSL session**
  - An association between client & server
  - Created by the handshake protocol
  - Define a set of cryptographic parameters
  - May be shared by multiple ssl connections

SSL record protocol services

- **Message integrity**
  - Using a mac with shared secret key
  - Similar to hmac but with different padding

- **Confidentiality**
  - Using symmetric encryption with a shared secret key defined by handshake protocol
  - Aes, idea, rc2-40, des-40, des, 3des, fortezza, rc4-40, rc4-128
  - Message is compressed before encryption

SSL record protocol operation

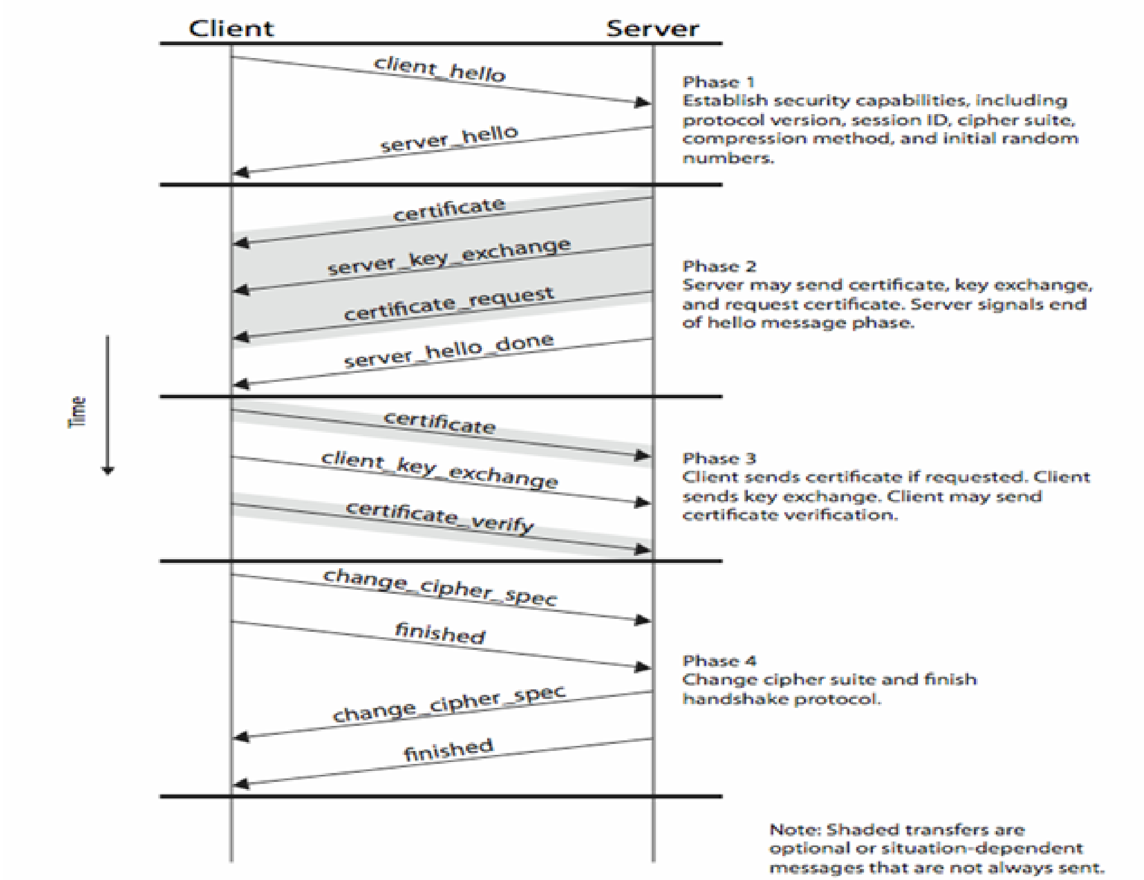


SSL alert protocol

- Conveys ssl-related alerts to peer entity
- Severity
    - Warning or fatal
- Specific alert
    - Fatal: unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter
    - Warning: close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown
- Compressed & encrypted like all ssl data

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



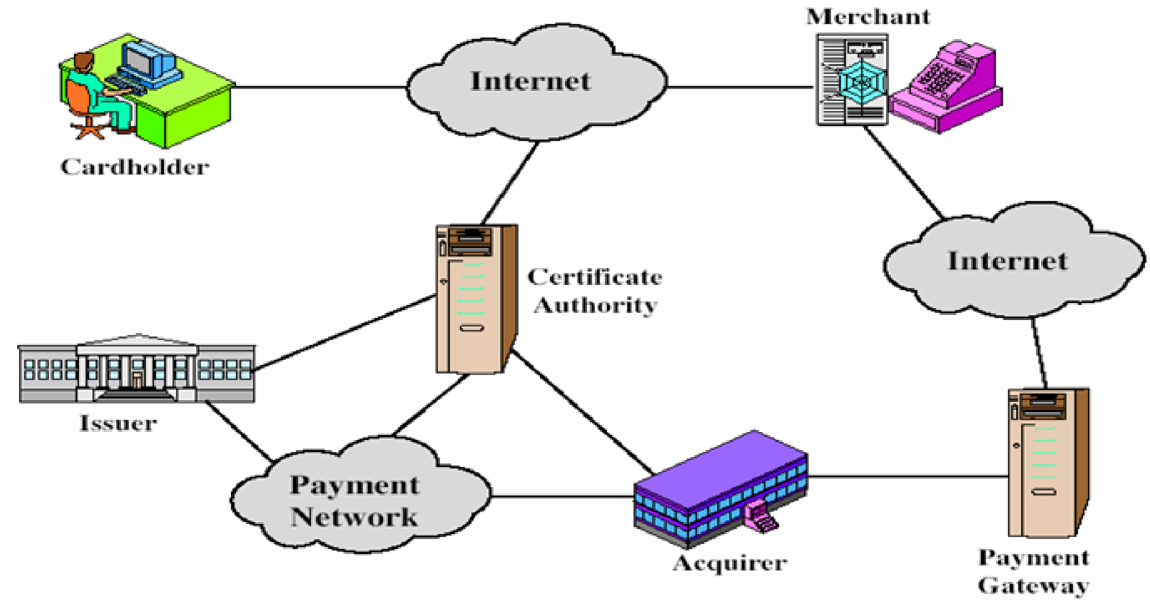
TLS (transport layer security)

- Ietf standard rfc 2246 similar to sslv3
- With minor differences
  - In record format version number
  - Uses hmac for mac
  - A pseudo-random function expands secrets
  - Has additional alert codes
  - Some changes in supported ciphers
  - Changes in certificate types & negotiations
  - Changes in crypto computations & padding

5. Briefly explain Secure Electronic Transactions.

- Open encryption & security specification
- To protect internet credit card transactions
- Developed in 1996 by mastercard, visa etc
- Not a payment system
- Rather a set of security protocols & formats
  - Secure communications amongst parties
  - Trust from use of x.509v3 certificates
  - Privacy by restricted info to those who need it

SET components



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

Dual signature

- Customer creates dual messages
  - Order information (oi) for merchant
  - Payment information (pi) for bank
- Neither party needs details of other
- But **must** know they are linked
- Use a dual signature for this
  - Signed concatenated hashes of oi & pi

Ds=e(prc, [h(h(pi)||h(oi))])

SET purchase request

- Set purchase request exchange consists of four messages
  1. Initiate request - get certificates
  2. Initiate response - signed response
  3. Purchase request - of oi & pi
  4. Purchase response - ack order

Purchase request – merchant

1. Verifies cardholder certificates using ca sigs
2. Verifies dual signature using customer's public signature key to ensure order has not been tampered with in transit & that it was signed using cardholder's private signature key
3. Processes order and forwards the payment information to the payment gateway for authorization (described later)
4. Sends a purchase response to cardholder

Payment gateway authorization

1. Verifies all certificates
2. Decrypts digital envelope of authorization block to obtain symmetric key & then decrypts authorization block
3. Verifies merchant's signature on authorization block
4. Decrypts digital envelope of payment block to obtain symmetric key & then decrypts payment block
5. Verifies dual signature on payment block
6. Verifies that transaction id received from merchant matches that in pi received (indirectly) from customer
7. Requests & receives an authorization from issuer
8. Sends authorization response back to merchant

Payment capture

- Merchant sends payment gateway a payment capture request
- Gateway checks request
- Then causes funds to be transferred to merchants account
- Notifies merchant using capture response

UNIT V
E-MAIL, IP & WEB SECURITY

1. Write short notes on Intrusion Detections.

- Significant issue for networked systems is hostile or unwanted access
- Either via network or local
- Can identify classes of intruders:
  - Masquerader
  - Misfeasor
  - Clandestine user
- Varying levels of competence
- Clearly a growing publicized problem
  - From "wily hacker" in 1986/87
  - To clearly escalating cert stats

- May seem benign, but still cost resources
- May use compromised system to launch other attacks
- Awareness of intruders has led to the development of certs

Intrusion techniques

- Aim to gain access and/or increase privileges on a system
- Basic attack methodology
    - Target acquisition and information gathering
    - Initial access
    - Privilege escalation
    - Covering tracks
- Key goal often is to acquire passwords
- So then exercise access rights of owner

Password guessing

- One of the most common attacks
- Attacker knows a login (from email/web page etc)
- Then attempts to guess password for it
    - Defaults, short passwords, common word searches
    - User info (variations on names, birthday, phone, common words/interests)
    - Exhaustively searching all possible passwords
- Check by login or against stolen password file
- Success depends on password chosen by user
- Surveys show many users choose poorly

Password capture

- Another attack involves **password capture**
    - Watching over shoulder as password is entered
    - Using a trojan horse program to collect
    - Monitoring an insecure network login
        - Eg. Telnet, ftp, web, email
    - Extracting recorded info after successful login (web history/cache, last number dialled etc)
- Using valid login/password can impersonate user
- Users need to be educated to use suitable precautions/countermeasures

Intrusion detection

- Inevitably will have security failures
- So need also to detect intrusions so can
    - Block if detected quickly
    - Act as deterrent
    - Collect info to improve security
- Assume intruder will behave differently to a legitimate user
    - But will have imperfect distinction between

Approaches to intrusion detection

- Statistical anomaly detection
    - Threshold
    - Profile based
- Rule-based detection
    - Anomaly
    - Penetration identification

Audit records

- Fundamental tool for intrusion detection
- Native audit records
    - Part of all common multi-user o/s
    - Already present for use
    - May not have info wanted in desired form
- Detection-specific audit records
    - Created specifically to collect wanted info
    - At cost of additional overhead on system

Statistical anomaly detection

- Threshold detection
    - Count occurrences of specific event over time
    - If exceed reasonable value assume intrusion
    - Alone is a crude & ineffective detector
- Profile based
    - Characterize past behavior of users
    - Detect significant deviations from this
    - Profile usually multi-parameter

Audit record analysis

- Foundation of statistical approaches
- Analyze records to get metrics over time
    - Counter, gauge, interval timer, resource use
- Use various tests on these to determine if current behavior is acceptable
    - Mean & standard deviation, multivariate, markov process, time series, operational
- Key advantage is no prior knowledge used

Rule-based intrusion detection

- Observe events on system & apply rules to decide if activity is suspicious or not
- Rule-based anomaly detection
    - Analyze historical audit records to identify usage patterns & auto-generate rules for them
    - Then observe current behavior & match against rules to see if conforms
    - Like statistical anomaly detection does not require prior knowledge of security flaws
- Rule-based penetration identification
    - Uses expert systems technology

- With rules identifying known penetration, weakness patterns, or suspicious behavior
- Compare audit records or states against rules
- Rules usually machine & o/s specific
- Rules are generated by experts who interview & codify knowledge of security admins
- Quality depends on how well this is done

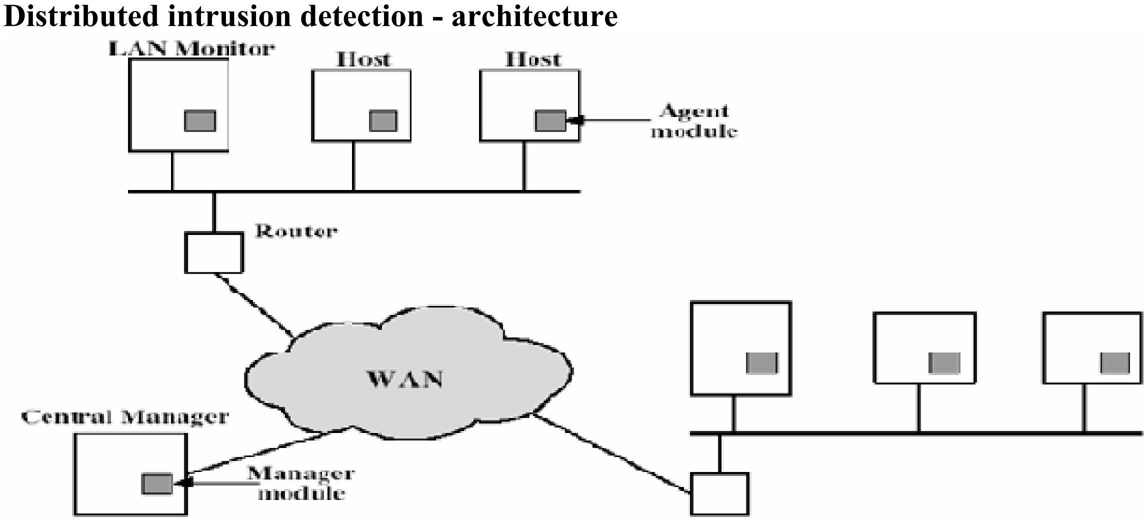
Base-rate fallacy

- Practically an intrusion detection system needs to detect a substantial percentage of intrusions with few false alarms
  - If too few intrusions detected -> false security
  - If too many false alarms -> ignore / waste time
- This is very hard to do
- Existing systems seem not to have a good record

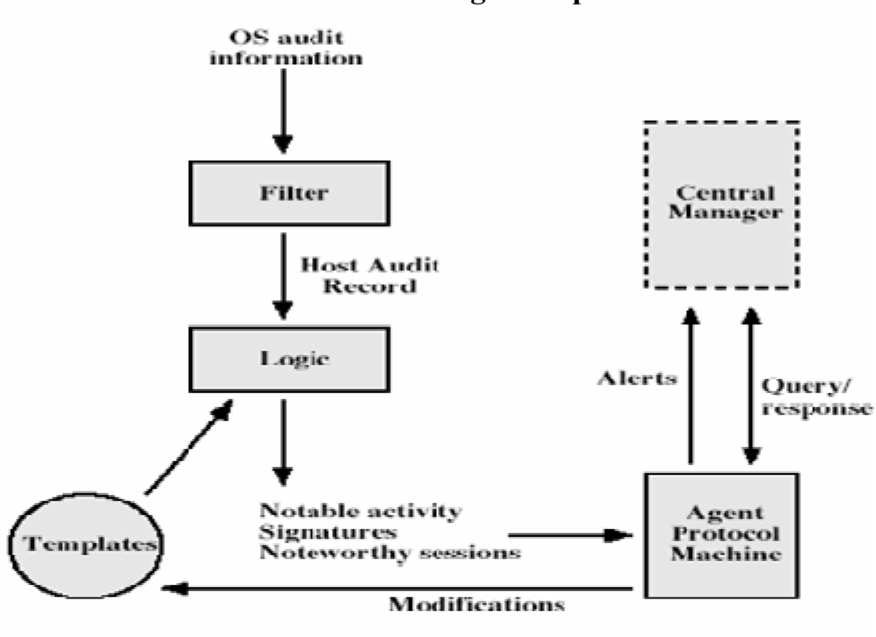
Distributed intrusion detection

- Traditional focus is on single systems
- But typically have networked systems
- More effective defense has these working together to detect intrusions
- Issues
  - Dealing with varying audit record formats
  - Integrity & confidentiality of networked data
  - Centralized or decentralized architecture

Distributed intrusion detection - architecture



Distributed intrusion detection – agent implementation



Honeypots

- Decoy systems to lure attackers
  - Away from accessing critical systems
  - To collect information of their activities
  - To encourage attacker to stay on system so administrator can respond
- Are filled with fabricated information
- Instrumented to collect detailed information on attackers activities
- Single or multiple networked systems
- Cf ietf intrusion detection wg standards

2. Briefly explain about Password Management

- Front-line defense against intruders
- Users supply both:
  - Login – determines privileges of that user
  - Password – to identify them
- Passwords often stored encrypted
  - Unix uses multiple des (variant with salt)
  - More recent systems use crypto hash function
- Should protect password file on system

Password studies

- Purdue 1992 - many short passwords
- Klein 1990 - many guessable passwords
- Conclusion is that users choose poor passwords too often

- Need some approach to counter this

Managing passwords - education

- Can use policies and good user education
- Educate on importance of good passwords
- Give guidelines for good passwords
  - Minimum length (>6)
  - Require a mix of upper & lower case letters, numbers, punctuation
  - Not dictionary words
- But likely to be ignored by many users

Managing passwords - computer generated

- Let computer create passwords
- If random likely not memorisable, so will be written down (sticky label syndrome)
- Even pronounceable not remembered
- Have history of poor user acceptance
- Fips pub 181 one of best generators
  - Has both description & sample code
  - Generates words from concatenating random pronounceable syllables

Managing passwords - reactive checking

- Reactively run password guessing tools
  - Note that good dictionaries exist for almost any language/interest group
- Cracked passwords are disabled
- But is resource intensive
- Bad passwords are vulnerable till found

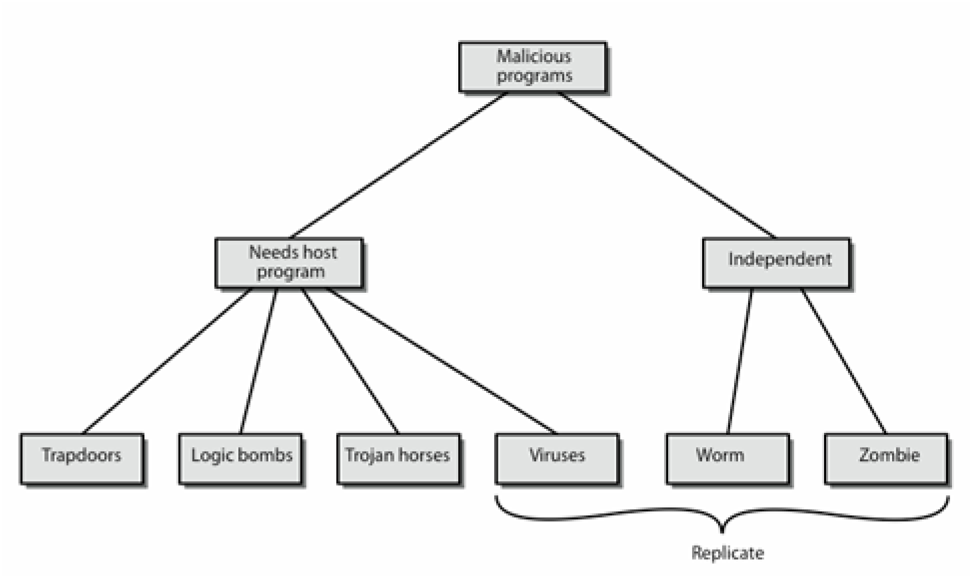
Managing passwords - proactive checking

- Most promising approach to improving password security
- Allow users to select own password
- But have system verify it is acceptable
  - Simple rule enforcement (see earlier slide)
  - Compare against dictionary of bad passwords
  - Use algorithmic (markov model or bloom filter) to detect poor choices

3. Define virus. Explain in detail.

- Computer viruses have got a lot of publicity
- One of a family of **malicious software**
- Effects usually obvious
- Have figured in news reports, fiction, movies (often exaggerated)
- Getting more attention than deserve
- Are a concern though

Malicious software



Backdoor or trapdoor

- Secret entry point into a program
- Allows those who know access bypassing usual security procedures
- Have been commonly used by developers
- A threat when left in production programs allowing exploited by attackers
- Very hard to block in o/s
- Requires good s/w development & update

Logic bomb

- One of oldest types of malicious software
- Code embedded in legitimate program
- Activated when specified conditions met
  - Eg presence/absence of some file
  - Particular date/time
  - Particular user
- When triggered typically damage system
  - Modify/delete files/disks, halt machine, etc

Trojan horse

- Program with hidden side-effects
- Which is usually superficially attractive
  - Eg game, s/w upgrade etc
- When run performs some additional tasks
  - Allows attacker to indirectly gain access they do not have directly
- Often used to propagate a virus/worm or install a backdoor
- Or simply to destroy data

Zombie

- Program which secretly takes over another networked computer
- Then uses it to indirectly launch attacks
- Often used to launch distributed denial of service (ddos) attacks

- Exploits known flaws in network systems

Viruses

- A piece of self-replicating code attached to some other code
    - Cf biological virus
- Both propagates itself & carries a payload
    - Carries code to make copies of itself
    - As well as code to perform some covert task

Virus operation

- Virus phases:
    - Dormant – waiting on trigger event
    - Propagation – replicating to programs/disks
    - Triggering – by event to execute payload
    - Execution – of payload
- Details usually machine/os specific
    - Exploiting features/weaknesses

Virus structure

```
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 condition holds}
        main: main-program :=       {infect-executable;
                                    if trigger-pulled then do-damage;
                                    goto next;}
        next:
}
```

4. Briefly explain the types of virus

- Can classify on basis of how they attack
- Parasitic virus
- Memory-resident virus
- Boot sector virus
- Stealth
- Polymorphic virus
- Metamorphic virus

Macro virus

- **Macro code** attached to some **data file**
- Interpreted by program using file
  - Eg word/excel macros
  - Esp. Using auto command & command macros
- Code is now platform independent
- Is a major source of new viral infections
- Blur distinction between data and program files
- Classic trade-off: "ease of use" vs "security"
- Have improving security in word etc
- Are no longer dominant virus threat

Email virus

- Spread using email with attachment containing a macro virus
  - Cf melissa
- Triggered when user opens attachment
- Or worse even when mail viewed by using scripting features in mail agent
- Hence propagate very quickly
- Usually targeted at microsoft outlook mail agent & word/excel documents
- Need better o/s & application security

Worms

- Replicating but not infecting program
- Typically spreads over a network
  - Cf morris internet worm in 1988
  - Led to creation of certs
- Using users distributed privileges or by exploiting system vulnerabilities
- Widely used by hackers to create **zombie pc's**, subsequently used for further attacks, esp dos
- Major issue is lack of security of permanently connected systems, esp pc's

Worm operation

- Worm phases like those of viruses:
  - Dormant
  - Propagation
    - Search for other systems to infect
    - Establish connection to target remote system
    - Replicate self onto remote system
  - Triggering
  - Execution

Morris worm

- Best known classic worm
- Released by robert morris in 1988
- Targeted unix systems
- Using several propagation techniques
    - Simple password cracking of local pw file
    - Exploit bug in finger daemon
    - Exploit debug trapdoor in sendmail daemon
- If any attack succeeds then replicated self

Recent worm attacks

- New spate of attacks from mid-2001
- Code red - used ms iis bug
    - Probes random ips for systems running iis
    - Had trigger time for denial-of-service attack
    - 2nd wave infected 360000 servers in 14 hours
- Code red 2 - installed backdoor
- Nimda - multiple infection mechanisms
- Sql slammer - attacked ms sql server
- Sobig.f - attacked open proxy servers
- Mydoom - mass email worm + backdoor

Worm techology

- Multiplatform
- Multiexploit
- Ultrafast spreading
- Polymorphic
- Metamorphic
- Transport vehicles
- Zero-day exploit

VIRUS COUNTERMEASURES

- Best countermeasure is prevention
- But in general not possible
- Hence need to do one or more of:
    - **Detection** - of viruses in infected system
    - **Identification** - of specific infecting virus
    - **Removeal** - restoring system to clean state

Anti-virus software

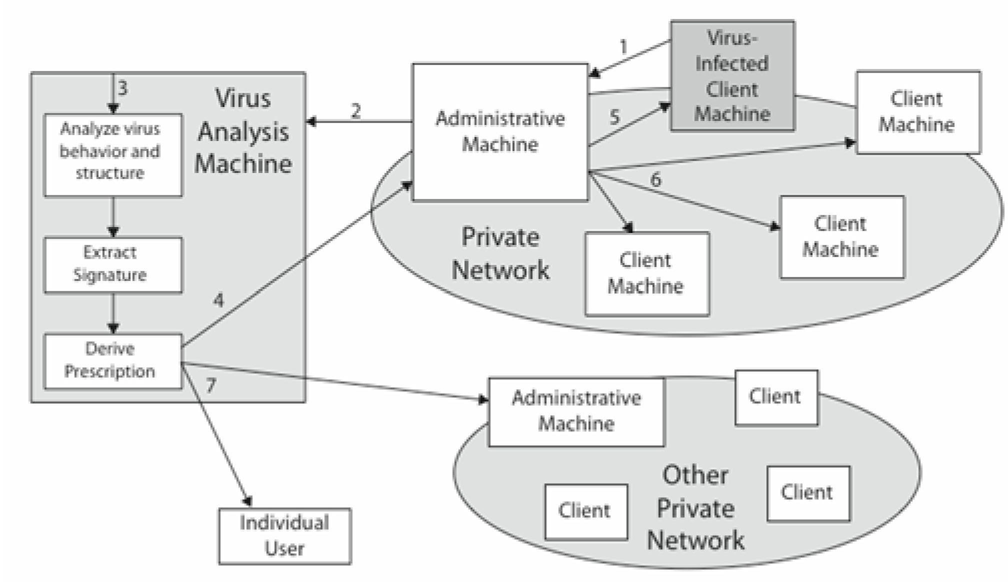
- **First-generation**

- Scanner uses virus signature to identify virus
- Or change in length of programs
- **Second-generation**
  - Uses heuristic rules to spot viral infection
  - Or uses crypto hash of program to spot changes
- **Third-generation**
  - Memory-resident programs identify virus by actions
- **Fourth-generation**
  - Packages with a variety of antivirus techniques
  - Eg scanning & activity traps, access-controls
- Arms race continues

Advanced anti-virus techniques

- Generic decryption
  - Use cpu simulator to check program signature & behavior before actually running it
- Digital immune system (ibm)
  - General purpose emulation & virus detection
  - Any virus entering org is captured, analyzed, detection/shielding created for it, removed

Digital immune system



Behavior-blocking software

- Integrated with host o/s
- Monitors program behavior in real-time
  - Eg file access, disk format, executable mods, system settings changes, network access
- For possibly malicious actions
  - If detected can block, terminate, or seek ok
- Has advantage over scanners
- But malicious code runs before detection

Distributed denial of service attacks (DDOS)

- Distributed denial of service (ddos) attacks form a significant security threat
- Making networked systems unavailable
- By flooding with useless traffic
- Using large numbers of "zombies"
- Growing sophistication of attacks
- Defense technologies struggling to cope

Contructing the ddos attack network

- Must infect large number of zombies
- Needs:

1. Software to implement the ddos attack
2. An unpatched vulnerability on many systems
3. Scanning strategy to find vulnerable systems
   - Random, hit-list, topological, local subnet

DDOS countermeasures

- Three broad lines of defense:
    1. Attack prevention & preemption (before)
    2. Attack detection & filtering (during)
    3. Attack source traceback & ident (after)
- Huge range of attack possibilities
- Hence evolving countermeasures

5. Explain the technical details of firewall and describe any three types of firewall with neat diagram .

Introduction

- Seen evolution of information systems
- Now everyone want to be on the internet
- And to interconnect networks
- Has persistent security concerns
    - Can't easily secure every system in org
- Typically use a **firewall**
- To provide **perimeter defence**
- As part of comprehensive security strategy

What is a firewall?

- A **choke point** of control and monitoring
- Interconnects networks with differing trust
- Imposes restrictions on network services
    - Only authorized traffic is allowed
- Auditing and controlling access
    - Can implement alarms for abnormal behavior
- Provide nat & usage monitoring
- Implement vpns using ipsec
- Must be immune to penetration

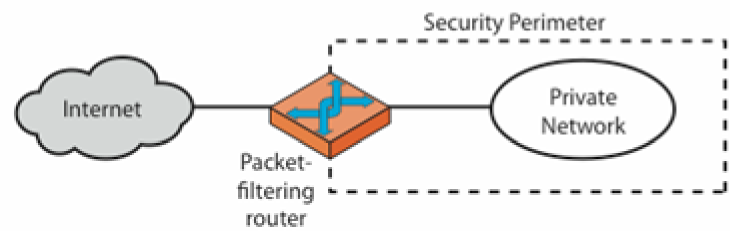
Firewall limitations

- Cannot protect from attacks bypassing it
    - Eg sneaker net, utility modems, trusted organisations, trusted services (eg ssl/ssh)
- Cannot protect against internal threats
    - Eg disgruntled or colluding employees
- Cannot protect against transfer of all virus infected programs or files
    - Because of huge range of o/s & file types

Firewalls – packet filters

- Simplest, fastest firewall component
- Foundation of any firewall system
- Examine each ip packet (no context) and permit or deny according to rules
- Hence restrict access to services (ports)
- Possible default policies

- That not expressly permitted is prohibited
- That not expressly prohibited is permitted



(a) Packet-filtering router

Attacks on packet filters

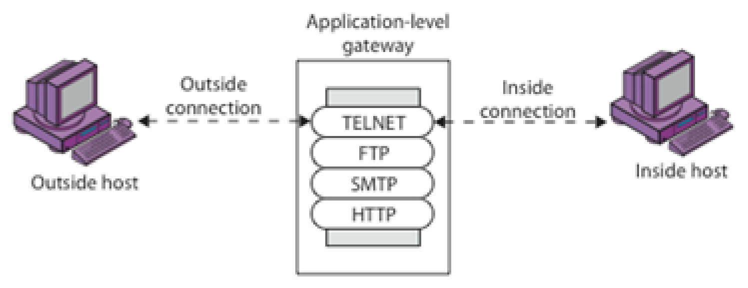
- Ip address spoofing
  - Fake source address to be trusted
  - Add filters on router to block
- Source routing attacks
  - Attacker sets a route other than default
  - Block source routed packets
- Tiny fragment attacks
  - Split header info over several tiny packets
  - Either discard or reassemble before check

Firewalls – stateful packet filters

- Traditional packet filters do not examine higher layer context
  - Ie matching return packets with outgoing flow
- Stateful packet filters address this need
- They examine each ip packet in context
  - Keep track of client-server sessions
  - Check each packet validly belongs to one
- Hence are better able to detect bogus packets out of context

Firewalls - application level gateway (or proxy)

- Have application specific gateway / proxy
- Has full access to protocol
  - User requests service from proxy
  - Proxy validates request as legal
  - Then actions request and returns result to user
  - Can log / audit traffic at application level
- Need separate proxies for each service
  - Some services naturally support proxying
  - Others are more problematic

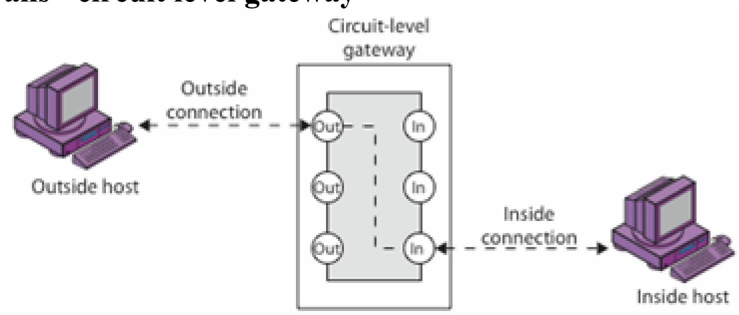


(b) Application-level gateway

Firewalls - circuit level gateway

- Relays two tcp connections
- Imposes security by limiting which such connections are allowed
- Once created usually relays traffic without examining contents
- Typically used when trust internal users by allowing general outbound connections
- Socks is commonly used

Firewalls - circuit level gateway

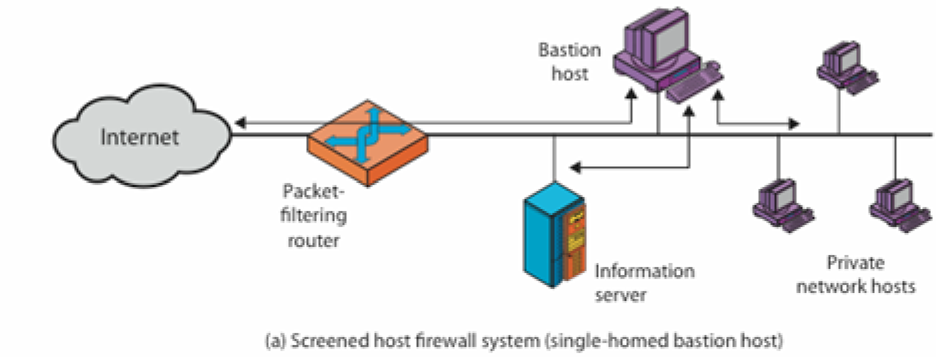


(c) Circuit-level gateway

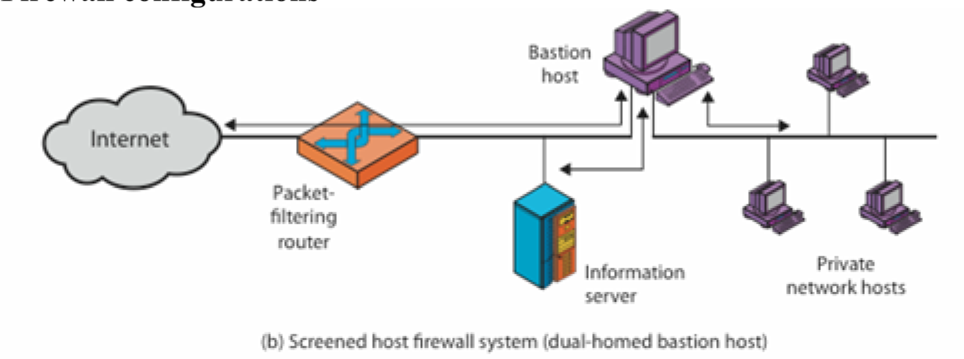
Bastion host

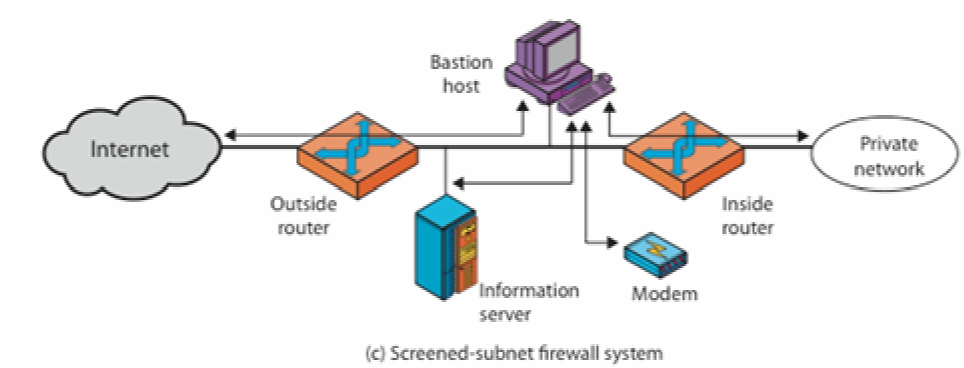
- Highly secure host system
- Runs circuit / application level gateways
- Or provides externally accessible services
- Potentially exposed to "hostile" elements
- Hence is secured to withstand this
  - Hardened o/s, essential services, extra auth
  - Proxies small, secure, independent, non-privileged
- May support 2 or more net connections
- May be trusted to enforce policy of trusted separation between these net connections

Firewall configurations



Firewall configurations





Access control

- Given system has identified a user
- Determine what resources they can access
- General model is that of access matrix with
  - **Subject** - active entity (user, process)
  - **Object** - passive entity (file or resource)
  - **Access right** – way object can be accessed
- Can decompose by
  - Columns as access control lists
  - Rows as capability tickets

Access control matrix

	Program1	. . .	SegmentA	SegmentB
Process1	Read Execute		Read Write	
Process2				Read
⋮				

(a) Access matrix

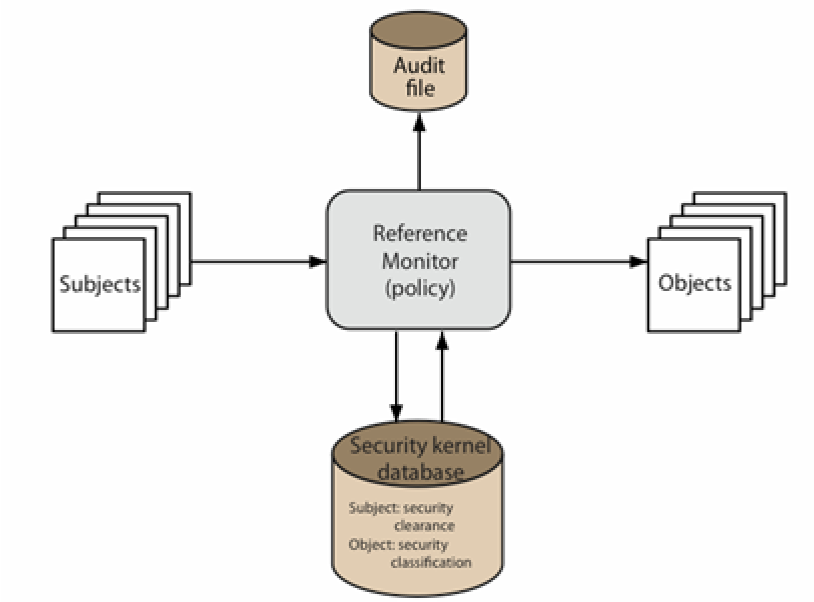
TRUSTED COMPUTER SYSTEMS

- Information security is increasingly important
- Have varying degrees of sensitivity of information
    - Cf military info classifications: confidential, secret etc
- Subjects (people or programs) have varying rights of access to objects (information)
- Known as multilevel security
    - Subjects have **maximum** & **current** security level
    - Objects have a fixed security level **classification**
- Want to consider ways of increasing confidence in systems to enforce these rights

Bell lapadula (blp) model

- One of the most famous security models
- Implemented as mandatory policies on system
- Has two key policies:
- **No read up** (simple security property)
    - A subject can only read/write an object if the current security level of the subject dominates (>=) the classification of the object
- **No write down** (*-property)
    - A subject can only append/write to an object if the current security level of the subject is dominated by (<=) the classification of the object

Reference monitor



Evaluated computer systems

- Governments can evaluate it systems
- Against a range of standards:
  - Tcsec, ipsec and now common criteria
- Define a number of "levels" of evaluation with increasingly stringent checking
- Have published lists of evaluated products
  - Though aimed at government/defense use
  - Can be useful in industry also

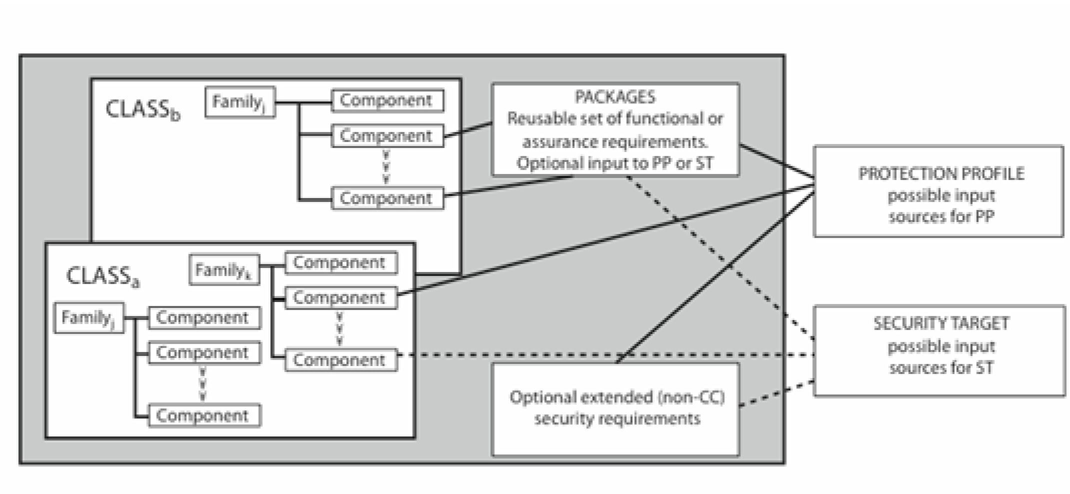
Common criteria

- International initiative specifying security requirements & defining evaluation criteria
- Incorporates earlier standards
  - Eg csec, itsec, ctcpec (canadian), federal (us)
- Specifies standards for
  - Evaluation criteria
  - Methodology for application of criteria
  - Administrative procedures for evaluation, certification and accreditation schemes

- Defines set of security requirements
- Have a target of evaluation (toe)
- Requirements fall in two categories
  - Functional
  - Assurance
- Both organised in classes of families & components

Common criteria requirements

- Functional requirements
  - Security audit, crypto support, communications, user data protection, identification & authentication, security management, privacy, protection of trusted security functions, resource utilization, toe access, trusted path
- Assurance requirements
  - Configuration management, delivery & operation, development, guidance documents, life cycle support, tests, vulnerability assessment, assurance maintenance



Common criteria