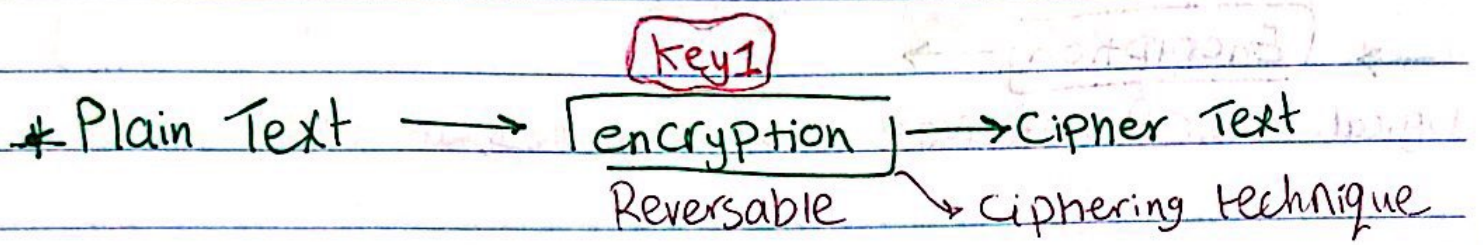
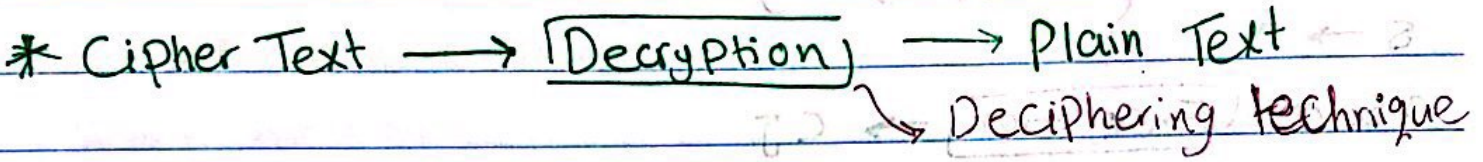
• Cryptology : encryption and decryption





• Cryptalysis : Try to guess the key or the plaintext based on the algorithm and the cipher text

*We always assume that the encryption and decryption algorithm is known to everybody
* The only secure thing is the key

• Symatric encryption ⇒ Key 1 = Key 2
• Asymatric encryption ⇒ Key 1 ≠ Key 2

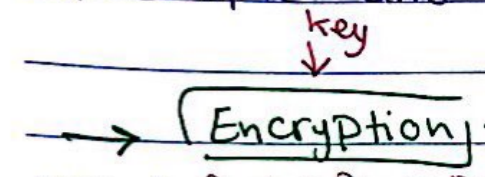
• Senario: Alice —Message→ BoB

Eve

• Active attack : Try to manipulate the data

• Passive attack : Will not destroy or change the data, only interrupt (sniff or collect) the data.

* Simple and naiive encryption techniques:



logical: And, OR, XOR, NOR ← key

Plain cipher
a → C
s → J
} Mapping key

as → [Encryption] → CJ

- Popular algorithms 2001

Symetric:
- DES
- 3DES
- AES

Asymetric:
- Diffic-Helma
- Elgamal
- Elleptic Curve
- RSA

it is a bad idea to design your own encryption technique

* Any algorithm should be secured against two things:
1) cryptanalysis
2) brute force attack: try every possible key or every possible plaintext to guess the key

- Problems with asymetric: They need high processing memory and time

* Hash Function:

Variable size text → [hash Function] → Fixed length hash value

→ Satisfies the data integrity

- Data integrity: make sure that the data was not manipulated or change
  - → by Encryption

- Data Confidentiality make sure that only legitimate users can use the data
  - → by hashing

- Authentication: make sure that you talk to the claimed side

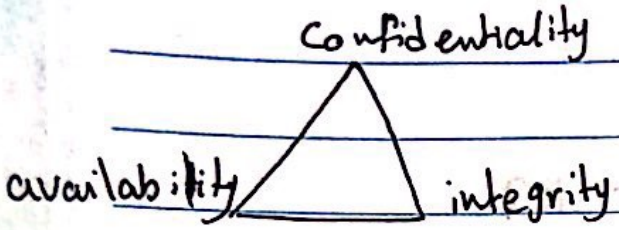
- Risk management
  - How possible a risk will happen?
  - What is the cost to prevent this risk?
  - What is the cost if the risk happened?
  - In case the risk happened, how it will take to recover from it? and what is the cost to recover
  - Cost-benifit-analysis

- Password: Design very strong password system
  - Password is sent encrypted
  - Minimum length is 15 characters
  - Small & capital letters
  - Special characters
  - Can't put 3 consecutive numbers or letters

- Detect trial of attack? check logs

- Physical security:
  - * Social engineering: based on social information about the victim, try to do some attacks

* Physhing: send an email to somebody with fake links to make the victim open malicious websites or files.



- Cryptographic: Solves Problems related to:
  - Confidentiality
  - Integrity
  - Availability
  - Non-Reputation: To prevent somebody from denying his actions.

- Accountability: know who is responsible about each action

- Anonymity: Nobody knows where I am by just seeing my buy order

---

* Cryptographic:

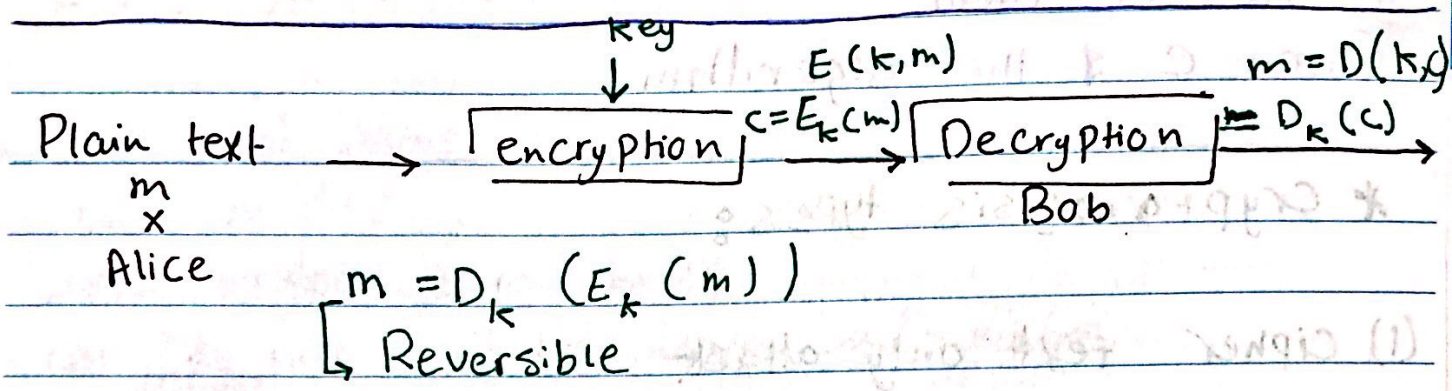
1- Substitution: Replace "a" by "C"
Replace "101" by "111"

2- Transposition: Replace letter with Position "x" by letter with position "y"

Key : Secret Part ⇝ Not easy task

[Random]

↳ For totally random secret each bit (b_i)
Should equal 0 or 1 with Probability $\frac{1}{2}$



$m = D_k(E_k(m))$

[↳ Reversible]

- The standard solutions are well tested and not subject to reverse engineering
- it is easier to share a key than an algorithm

*insider attack : Changing the key is much easier than changing the algorithm

*Any cryptographic algorithm must be secure. However it could be

→ only one example: one-time-Pad

(1) Unconditionally secure : whatever money, Processing, memory (resource) you have, you cant breat it

→ most of algos are :

(2) Computational secure : you can break it if you have enough resources. However the required resources are more expensive than the secret data

* The cryptanalysis: tries to find
m' or k' such that
m' = m
k' = k
or part of them
Given C & the algorithm

* Cryptanalysis types:

(1) Cipher text only attack

(2) Known Plain text: Pair of (ciphertext, algorithm, plaintext), try to guess the key or ~~subsequence~~ Subsequent messeges

(3) chosen Plain text: The attacker chooses the plaintext and asks one Party to encrypt using the shared key with the second Party
→ attacker knows the ciphertext of a chosen plaintext and the algorithm

Classic example: [redacted] Shift cipher

26 English letters
↳ 0 → 25

a = 0
b = 1
⋮
z = 25

Encryption algorithm = $(P_i + K) \bmod 26$
Decryption algorithm = $(C_i - K) \bmod 26$

Plain text = abc ⟶ Enc ⟶ cde
key = 2 (2 3 4)

Not secure against brute force attack and known plain text attack

---

Monoalphabetic Cipher

Substitute each letter by another letter

Plain	Cipher letter
a	x
b	z
c	e
d	f
e	j
⋮	⋮
z	

E(li) = cipher text letter (li)

c ← decryption — e cipher

b —encryption→ z

- is it secure against brute force attack?

Key size = 26! = 4×10^{26}

[redacted]

- IF you can do 10^6 decryption per second how long we need to break it on avarege?

$$\frac{4 * 10^{26}}{10^6} / 2$$

$= 2 \times 10^{20}$ seconds

$= 5.5 \times 10^{16}$ hours

$= 2.3 \times 10^{15}$ days

$= 6.3 \times 10^{12}$ years

* Frequency analysis :

For each letter in the cipher text, find its frequency

* Poly-alphabetic encryption: each letter may be represented by multiple letters
→ We need a way for reverting it

aa	cx
ab	cd
ac	⋮
⋮	⋮

• Diagrams: The Frequency of 2 letters are the same in Plain text & cipher text: ex: Th, is, io

• example of poly-alphabetic encryption:
Vigener cipher: another version of Shift cipher & Poly-alphabetic cipher.

Shift cipher
→ ~~encryption~~ : $(P_i + key) \bmod 26$
→ Vigener cipher : $(P_i + k_i) \bmod 26$, key size = messege size
(*) key word: repeat it as many times as you want

ex: Plain text length = 1000 letter
key word = key
→ key word word word ... 250 times

عدد أحرف word
1000 ÷ 4

- Assuming we know the key size → Frequency analysis

$(P_1, P_{10}, P_{19}, P_{28}, P_{37} \ldots)$
$P_2 \; P_{11} \; P_{20} \; P_{29} \; P_{38} \ldots$ Freq will be as in Plain text

$$\sum_{i=1}^{26} (P_i)^2 = 0.065$$

- For key size: 1 to 20
~~do freq~~
do Freq analysize on $\{(P_i * \text{key size})\}$

where i in $\frac{\text{Plain text size}}{\text{key size}}$

if $\sum (P_i)^2 \simeq 0.065$ return key size

- Assignment: 2 Parts encryption, decryption (5 marks)

Plain text.txt → Prog of encryption → cipher text using vigener

↓ ↑ Keyword

- bonus: ~~find~~ return at least letters (+2)

cipher text.txt → Hacking → size of key

- Defining Secrecy:

$P(m' = m \mid c) = P(m' = m)$ → cipher text gives you no information about the plaintext

where $\alpha = 0.000000001$ ⇒ not perfectly secure

if $\alpha = 0$ Perfect secure

Properties of one time Pad

- The key is truly random & of the same length of message

0 & 1: $P(K_i) = \frac{1}{2}$

$P(K_i = 1 \mid K_j) = \frac{1}{2} \quad \forall \; j < i$

If you know all previous bits of the key, probability of the next bit = 1 or 0 = $\frac{1}{2}$

- Encryption in OTP : $\boxed{C_i = m_i \oplus k_i}$

- Decryption in OTP : $m_i = c_i \oplus k_i$

$= (m_i \oplus k_i) \oplus k_i$

$= m_i \oplus (k_i \oplus k_i)$

$= m_i \oplus 0$

$= m_i$

- Pure random : $P(K_i = 0) = \frac{1}{2}$, $P(k_i = 1) = \frac{1}{2}$

$P(m_i = 0) = x$

$P(m_i = 1) = 1 - x$

m_i	$P(m_i)$	k_i	$P(k_i)$	c_i
0	x	0	0.5	0
0	x	1	0.5	1
1	1-x	0	0.5	1
1	1-x	1	0.5	0

- $P(E_1 \cup E_2) = P(E_1) + P(E_2) = (E_1 \cap E_2)$

- $P(E_1 \cap E_2)$ if E_1 & E_2 are independent $= P(E_1) \times P(E_2)$

- $P(c_i = 0) = P[(m_i = 0 \text{ \& } k_i = 0) \cup (m_i = 1 \text{ \& } k_i = 1)]$

$= P(m_i = 0 \text{ \& } k_i = 0) + P(m_i = 1 \text{ \& } k_i = 1)$

$= x * \frac{1}{2} + (1 - x)$

rand() → Random value between (0, 1)

For random value between $(0, 10^{19})$

(int) rand * 10^{19}

srand (seed) → time — 128 bit.

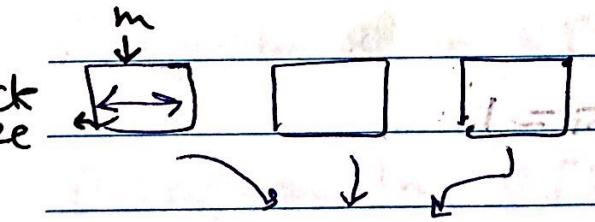
Alice & Bob agreed on the seed (initial key)

K_1 = Srand (seed) → 128 bits

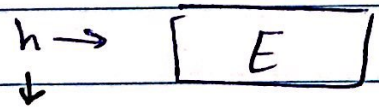
K_2 = Srand (K_1)

K_i = Srand (K_{i-1})

Block ciphers



Block size is determined by enc algorithm



large : To prevent brute force attack

64

128 bits

DES block size = 64 bits

DES key = 56 bits out of 64 bits

#Feistel network: A general architecture can be used by anybody to design his own encryption algorithm

*DES used Fiestel network
* Same block repeated many times (iteration)

Avalanche effect: Change in any bit of the inputs (key & message) will change approximately half of the output (cipher text)

* Repitition of some block → reduce code size that can be deployed on tiny devices and makes HW implementation easier

* Each round uses a key that is designed from original key

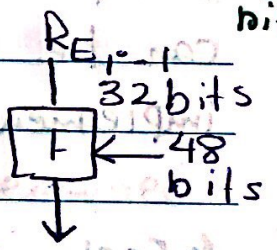
* $RE_i = LE_{i-1} \oplus F(k_i, RE_{i-1})$
$LE_i = RE_{i-1}$

* Parameters of Feistel:
. Key size, trade off between security and efficiency
. Round function: non-linear function & non piece-wise linear function
. # of rounds: trade of security & time
. Block size: trade of security & time
. Round key generation algorithm

(E) Same for encryption and decryption
no need for F^{-1} for decryption

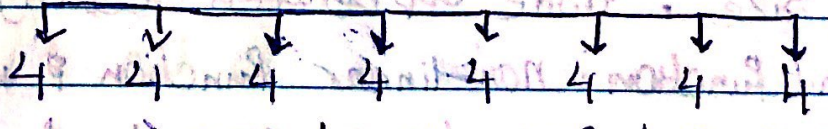
* Same code for encryption and decryption
* Other than the first permutation and last inverse permutation, DES has exactly Feistel structure

key (64 bits) → key expansion algorithm → 16 round keys, each of size 56 bits
- → Permutation of key
- → Circular shift left
- → Permutation 2 in each round. which is identified in all rounds



* Round fnction is composed of 4 steps:

1- Half block expansion: input is half block of size 32 bits → output (48 bits)



duplicate the LMB & RMB → 6 bits

48 bits

2) XOR 48 expanded word and the round key

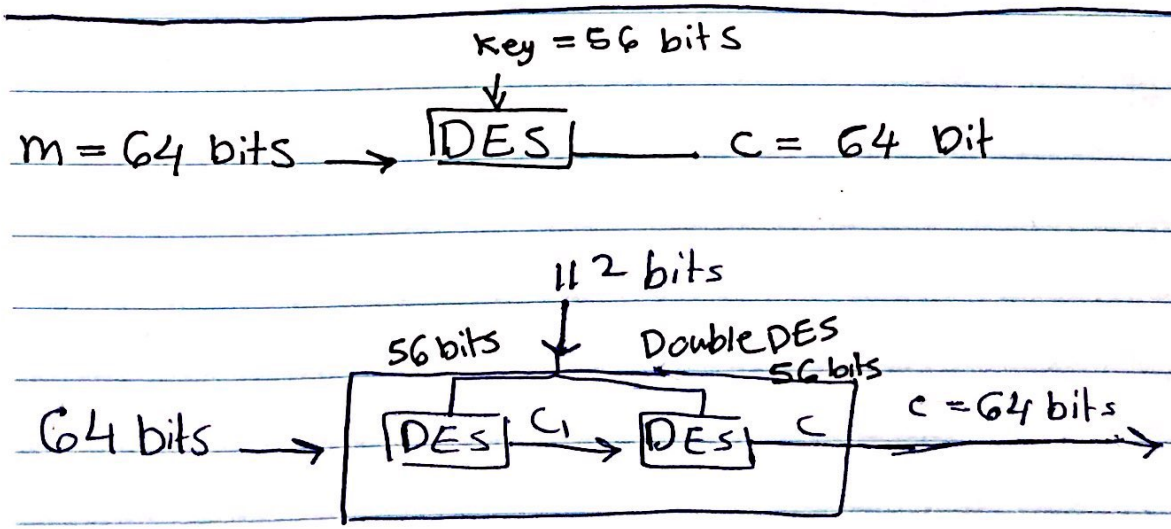
3) apply S-boxes on the generated 48 bits to generate 32 bits

S-Box: A Look up table (16 Columns)

makes round function non-linear

0000 0000 64 rows ⋮ 11111	4) Permutation 1101 1011

- Avalange effect
  - ↳ SAC :
  - ↳ BIC :

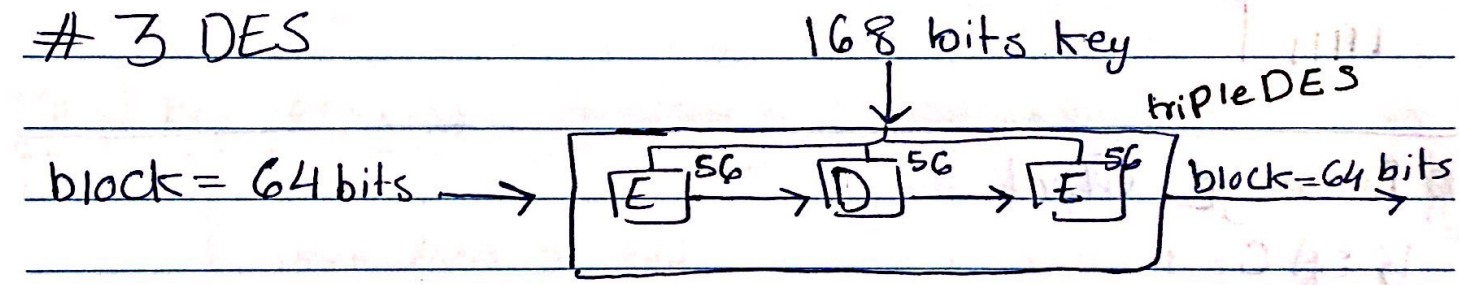


Theoritically = Brute force attack ⇒ on average we need 2^{111} decryption

- Meet in the middle attack
- if you have a single pair of plain text / cipher text I will build a table of 2^{56} rows, each row contains the enc of the plain text using the possible keys [2^{56} keys]

$E_{k_i}(P)$	2^{56} + on avg $\frac{2^{56}}{2}$ to break it

3 DES



168 → as efficient as 2^{167} as 112
- → not efficient, regarding key size
- → slow

Rijenal

↳ AES:

128 bits block size

128 bits cipher text

44 words

Key Size

round 0 +
[10 rounds] ↳ 128 bits (16 bytes) (4 words)

52 round 0 +
[12 rounds] ↳ 192 bits (24 bytes) (6 words)

round 0 +
[14 rounds] ↳ 256 bits (32 bytes) (8 words)

60 words

AES
128 bits

16 bytes

initial state ↓

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

*Several rounds

* Key expansion algorithm: takes as input the original key, and it expands it to several round keys

---

- Round 0 is just add round key
- Each round has 4 steps, except for the last round 3 steps
  - ↳ substitute bytes: Lookup table → 8 bits input, 8 bits output } none linearity
  - ↳ Shift rows
  - ↳ Mix Columns: Matrix multiplication $a_{ij} = \Sigma_{row_i} * Column_j$

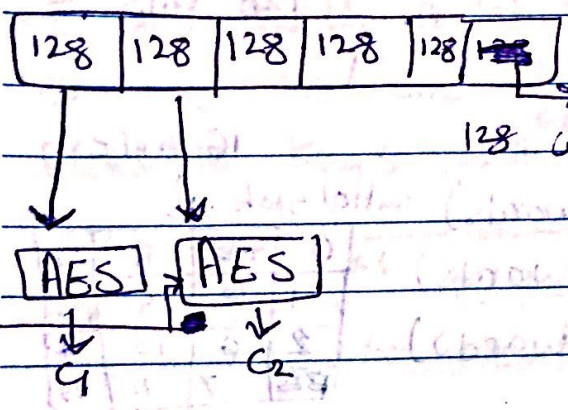
$$[a_{ij}]_{4 \times 4} = \begin{bmatrix} 32 & 11 \\ \end{bmatrix}$$ [(unreadable)

  - ↳ Add row keys: Bitwise XOR operation

- DES: Same code for enc & dec
- AES: not same code for enc & dec, you need inverse function

*Modes of encryption : when you have large text and you want to encrypt using block cipher algorithm, how you divide it

ECB



*Adv :

1) Parallel encryption & Dec

2) if one block was ~~damaged~~, lost you can retrieve the remain ~~Remain~~

(loss and noise tolerant~~e~~)

*Dis : similar blocks will be encrypted to the same cipher

Dont use ~~ECB~~ for long texts

→ Sol: if one block was repeated in the plaintext, it should give diffent cipher texts

- CBC: No Parallel sim (you cant encrypt block i before block i-1)
- Not tolerant to loss or noise
- The initialization vector should be kept secret & you may exchange it using ECB encryption

adv: you can use it to achieve data integrity

- You can ~~Choose~~ use any of the block cypher as stream cipher.

Cipher feedback mode [CFM]

adv: you can use it as stream cipher regardless of the plain text size
→ Same blocks will be ~~encoded~~ encrypted differently
→ Same code

Disadv: IV "Initialization vector"
↳ should be unpredictable
→ No parallelism
→ loss and noise effect
→ known plain text attack if same initialization vecto for multiple messeges

Output feedback mode [OFM]
nonce → A value that keeps changed
[Counter, timer, Date]

⊕ can be used as stream cipher
⊕ Pre-processing
⊕ loss and noise tolerance
⊕ Same code

⊖ keep changing nonce
⊖ Known Plaintext attack (same nonce) retrieve all subsequent messeges that are encrypted using same nonce

counter mode [CTR] the simplest
⊕ Hw and Sw efficent
⊕ Parallel
⊕ different cipher for same block
⊕ Same code for enc/dec
⊕ Preprocessing
⊖ Counter
⊖ moderate affect for noise or loss

• Messege authentication (2 solutions)
↳ Data integrity : the messege was not changed by unlegitite user
↳ Data origin authentication: the messege was created by the claimed Person

Solutions:
① Add MAC = Function (M, k)

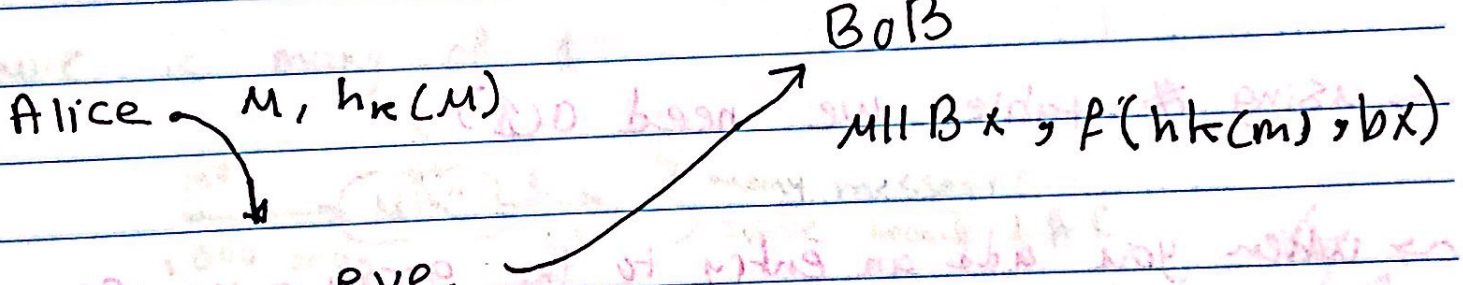
② Add digital signiture
M, MAC (M, k)

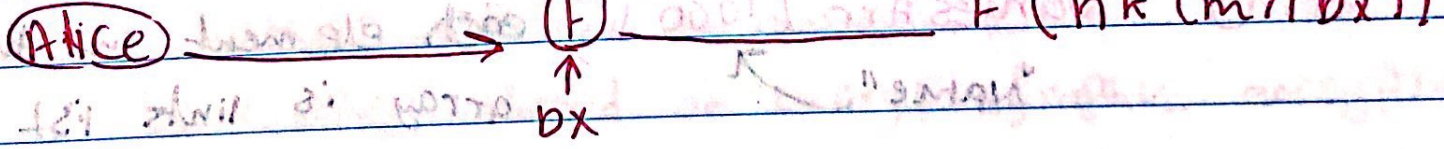
- Collision: To find 2 messeges m & m' such that $m = m'$, while $H(M) = H(m')$

Pre-Image → much more difficult

if hash value is of length (n) we need to check on average $\frac{2^n}{n}$ messages to finish Pre-Image

However, to find collision we need to check $2^{n/2}$ messages

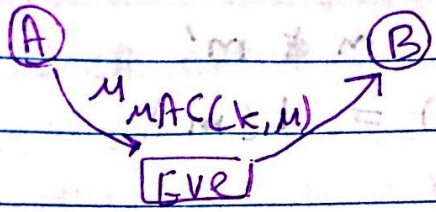




much faster

$k' = k$ // [zeros to generate one block]

$H(k // M)$



MAC(k, M')
Can eve generate this?
→ NO, except if she knows k

array of elements [1000] elements
↳ to find the index of an element, we need n/2
of average O(n), worst case
if the array is sorted, we need O(log n)
using binary search

→ using # table we need O(1)

→ when you add an entry to the array → you can calculate the index as function of array entry

String NamesArr [1000]; each element in this array is link list
"Name"

hash Function of element = Σ (ASCII codes of elements) mode 1000

Esraa → 213
Ali → 217
Khalid → 300

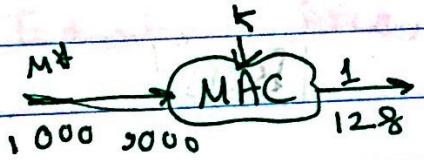
→ A good hash function, should be collision resistent
it is not easy to find 2 values
$(V_1 \neq V_2)$ with $H(V_1) = H(V_2)$

MAC ⇝ Uses mainly hash function

*One characteristic of hash functions that must be achieved to be one way

[irreversable] given H(M), you can't find M

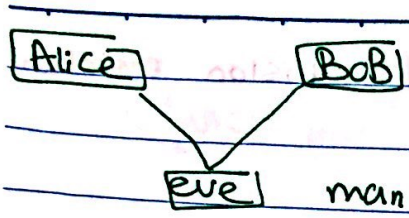
MAC is many of 1



$\infty \rightarrow 1$

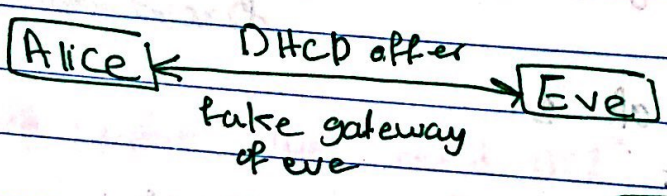
MAC → HMAC : Based on hash function
↳ CMAC : Based on encryptegraphic encryption

→ Pre-Image: if you have a message m & y = H(m) to find another messege (m') such that m' ≠ m while H(m') = y

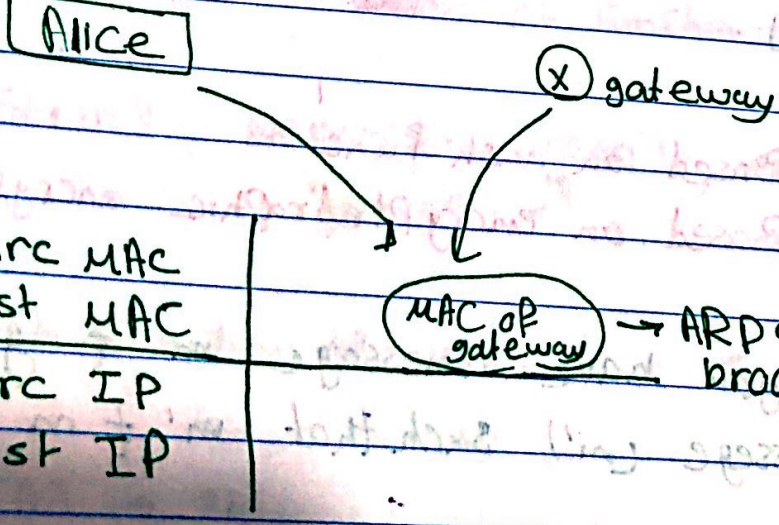


man in the middle attack
can be easily done

→ IP-address
→ Subnet mask
→ Local DNS IP address
→ Default gateway IP address



BoB



↓

Algo (n)

if complexity $= O(n^a)$ where a is constant

→ Feasible

→ can be solved in Ploynomial time

if complexity $= O(2^n)$ or (a^n)

→ NP Complete

→ None Polynomial time complex

Mike, Ted, Bob, Alice, John

EPRB (P) → (Cant be impimented because PRB not available

Plaintext
Alice ——————→ BoB

Alice knows :
- Plain text
- PR_A
- RU_A
- PU_M
- PU_T
- PU_J
- PU_B

EPU_A (P) → no benifit (useless) because Bob does not know the Privatekey of Alice
EPUB (P)
EPRA (P)

→ Bob can decrypt the msg because he knows PR_B (Confidentialy)
Bob can retrieve the msg Because he has the Public Key of Alice
→ Data integrity and authentication

→ Part 8 "Public key encryption"
197 → Diffie and Hellman → For key exchange

* Public key encryption : everyon maintain 2 keys
a Public Key (PU) and a Private Key (PR)

→ The Public key :
↳ known to everyone → you can Publish it on website

→ The Private key :
↳ Only known by the owner → kept loccaly no one should have access to it

(PU PR) They should have mathematical relationship
↳ encryption operation is a mathematical equation

any (int) number (n) can be represented by the form

$$a = P_1^{a_1} * P_2^{a_2} * \dots * P_{PL}^{a_L}$$

where $P_i < P_{i+1}$

• Ex: 48/2 → 24/2 → 12/2 → 6/2 → 3/3 → 1
$= 2^4 * 3$

→ The idea is based on finding large prime numbers
ex: (2048 or 4096 bit)

* There is no mathematical proof that a number is Prime or not → there is no such function that is efficient

Fun (A) → if prime → True
↳ if (not prime → False

(1/410)

* Miller-Rubin Algorithm:
↳ Takes as input integer (n) and returns:
in conlusine: no conclusion → probability prime →
→ if the int n is not prime → with probability $< \frac{1}{4}$
it returns inCondusine

• composite: Definitly not prime
it will be invoked several times → if I invoked it 10 times with different values of parameter "a" and it returned in 10 times inclusive →
wip $(1 - \frac{1}{410})$ the number is prime

ex: $n = 57$

↳ $n-1 = 2^k * q$

① $56 = 2^3 * 7$

② $1 < q < n-1$

③ if $3^7 \bmod 57 = 1$

return inconclusive

④ For $j = 0$ to $k = -1$ do

if $(3^{2^j * 7} \bmod 57 = 56)$

return ("incomplete")

return ("complete")

→ all public encryption protocols need trap-door one way function.

* one way function:-
it is easy to calculate $y = f(x)$ given x
it is infeasible to calculate $x = f^{-1}(y)$ given y

* trap door one way function:-
$y = f_K(x)$ easy given x & K.
↓
$X = f_K(y)$ infeasible given only y.
↓
$x = f_K(y)$ easy given y & k.

→ trap door one way function like factoring function.
if you have a & b.

is it easy to find $n = a * b$ ✓
given n is it easy to find a or b ✗
is it easy to find a given n & b.

→ RSA:- is based on factoring trap door one function.

EL-gamel & Diffie Helman they are based on discrete log trap door one way function.

e = 7 } Ex in slides.
n = 187
d = 23
m = 88.

Fermat thm:: if P is Prime then for all a $a^{P-1} = 1 \mod P$.

$a^{P-1} \mod P = 1$.

C.R.T $X = a \mod P$

$X = a \mod q$.

$X = a \mod (P * q)$.

→ for correctness of R.S.A

$M^{e*d} \mod n = M$. → for this to be correct

$e*d \mod \phi(n) = 1$ $e*d = k * \phi(n) + 1$.

eg. $e = 7$, $d = 11$, $\phi(n) = 19$

$e*d \mod \phi(n) = 1$.

↳ $e*d = k * \phi(n) + 1$

$7 * 11 = k * 19 + 1$

$k = 4$ #

$X \mod n = a$ → $X = K * n + a$ (K: constant.)

$X = 60$, $n = 8$, $a = 4$.

→ For correctness:-

$$M = (M^e)^d \bmod n.$$

$$M = M^{ed} \bmod n.$$

* To Calculate Keys in RSA :-

① choose p & q to be large prime numbers. [3000, 4000 bits]

② calculate $n = q * p$

③ $\phi(n) = (n-1)(q-1)$

④ Select e, with e is relatively prime to $\phi(n)$.

⑤ calculate $d = e^{-1} \bmod \phi(n)$ } there is algorithm to calculate d, Given e & $\phi(n)$.
$d * e \bmod \phi(n) = 1$

if $n = p * q$ (p, q: Prime numbers.)

$\phi(n) = (p-1) * (q-1)$.

$p = 3, q = 7$

$n = 3 * 7 = \boxed{21}$

$\phi(21) = 12$.

1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20

p, q, n, $\phi(n)$, e, d

Private. d: Private.

p, q, ϕn → only needed only @ Key calculation then they are discard

FIVE APPLE

RSA:- is a block cipher.

→ each block will be handled as an integer between 0 & (n-1), where n is typically 1096 bits

Euler Tutient function :-

$\phi(n)$:- number of integers < n that are relatively prime to n.

x & a are relatively prime, $\gcd(x,a) = 1$

18, 5 ✓
12, 7 ✓
18, 6 X.

By convention
$\phi(1) = 1$

$\phi(p) = p-1$, if p is Prime.

Enc in RSA :-

$C = M^e \bmod n$ (e, n) Public

Dec :-

$M = C^d \bmod n$ (d, n) Private.

*the only secure thing is d.

Public e, n P, q

Private d, n $n = P * q$.

$C = M^e \bmod n$ $\phi(n) = (P-1)*(q-1)$

$M = C^d \bmod n$ Find $\begin{bmatrix} e \\ d \end{bmatrix}$ s.t $gcd(e, \phi(n)) = 1$

$= e^{-1} \bmod \phi(n)$.

Pri.

e, d, P, q, n, $\phi(n)$

e → Pub n → Pub $\phi(n)$ → Pri

∴

e, n → d

→ n = P * q.

If you can factor n to P * q → you discover the private key.

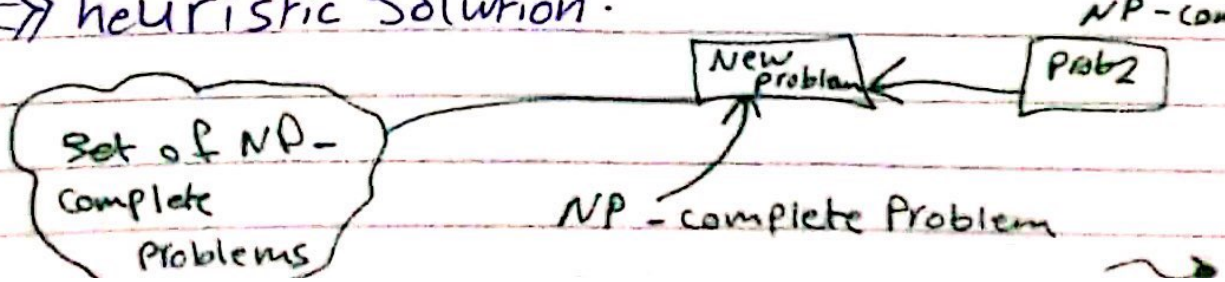
NP- complete Problem :-

P → complexity $O(n^a)$

NP → Complexity (a^n), (2^n)

→ once you prove that the problem is NP. problem → no optimal solution

⟹ heuristic Solution.



128 bits key of AES almost equivalent to 2048 bits, key of RSA

↳ from Security perspective.

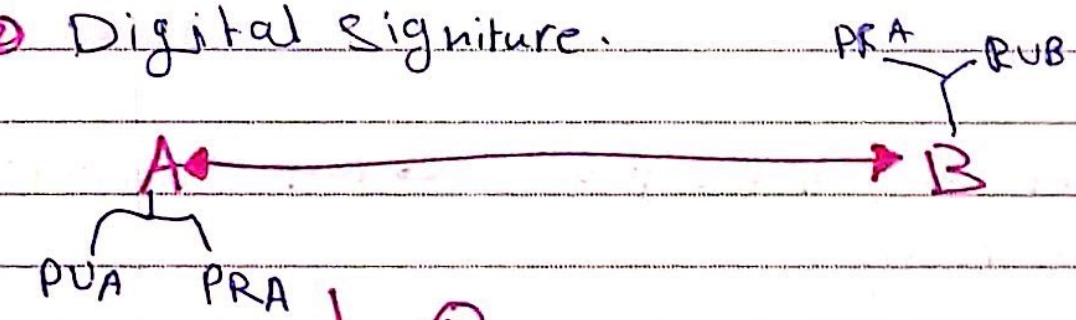
→ 4096 key size still much slower than AES one.

Public key Encryption is too slow to Enc data

Therefore we use them only :-

① Key exchange.

② Digital Signature.



No.

$M^{ed} = M^{k*(p-1)(q-1)+1}$

$\equiv M * M^{k(p-1)(q-1)}$

$\equiv M * [M^{(p-1)}]^{k(q-1)}$

$\equiv M * [1]$

$\equiv M \bmod p$.

$M^{ed} = M^{k(p-1)*(q-1)+1}$

$\equiv M * [M^{(q-1)}]^{k(p-1)}$

$\equiv M * 1^{k(p-1)} \bmod q$.

$M * 1 \bmod q$.

$M^{ed} = M \bmod q$.

$\rightarrow M^{ed} = M \bmod$ —

$M^{ed} = M \bmod n$, ##

؟ثبت ان ال DES برجعلي ال message.

$A^b \stackrel{?}{=} B^a$.

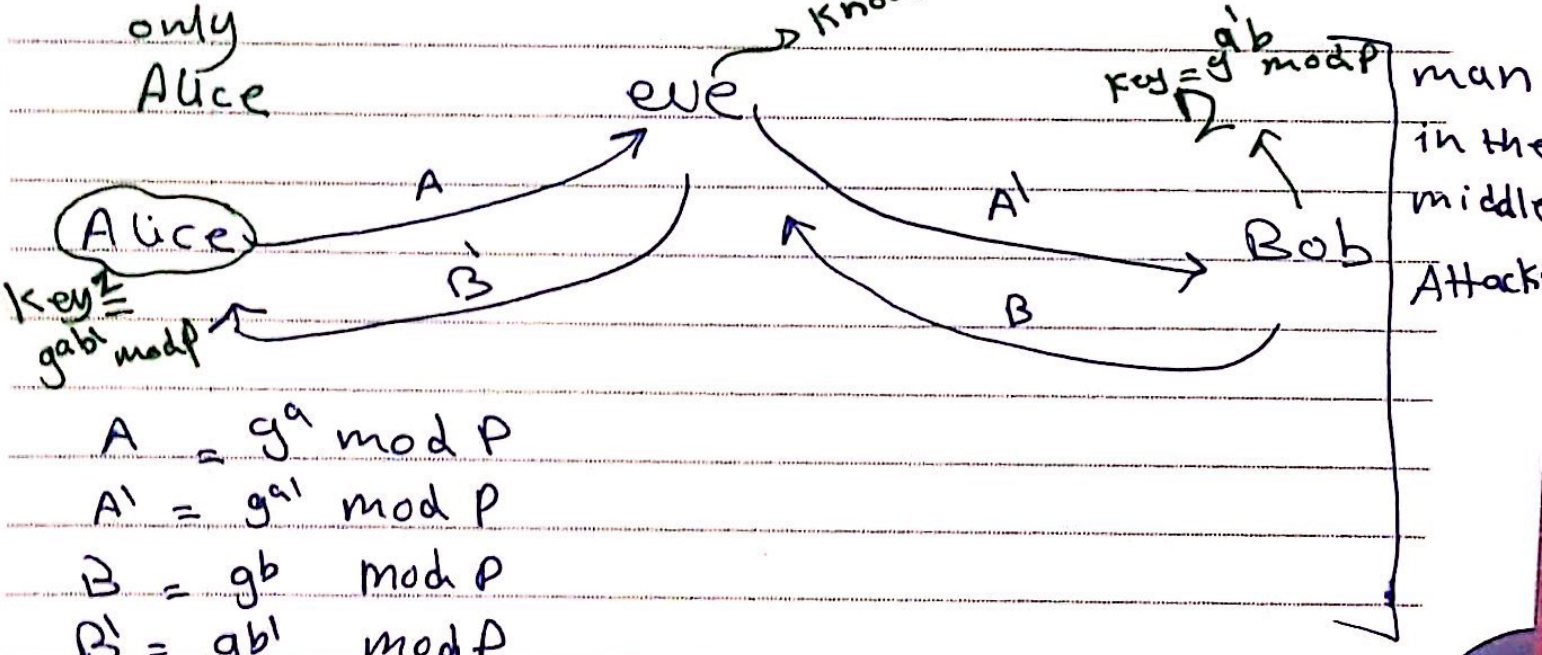
$(g^a \bmod p)^b \stackrel{?}{\equiv} (g^b \bmod p)^a$

$g^{axb} \bmod p \stackrel{?}{=} g^{axb} \bmod p$

→ any Body knows g, P, A, B but not a & b

only Alice

only Bob. knows K1, K2.



$A = g^a \bmod p$

$A' = g^{a'} \bmod p$

$B = g^b \bmod p$

$B' = g^{b'} \bmod p$

each message sent from Alice & Bob will be dec. using key1 and then enc using key2.

$Key_2 = g^{ab} \bmod p$.

→ to prove that a public belongs to me I need a digital certificate signed by the private key of the CA.

⁂ for any body to verify my signature, they need to know them public key of the C.A.

usually it is embedded in the OS.

Digital certificate has cost & expiration date.

Diffie AND Hellman.

⁂ No shared information between A & B,

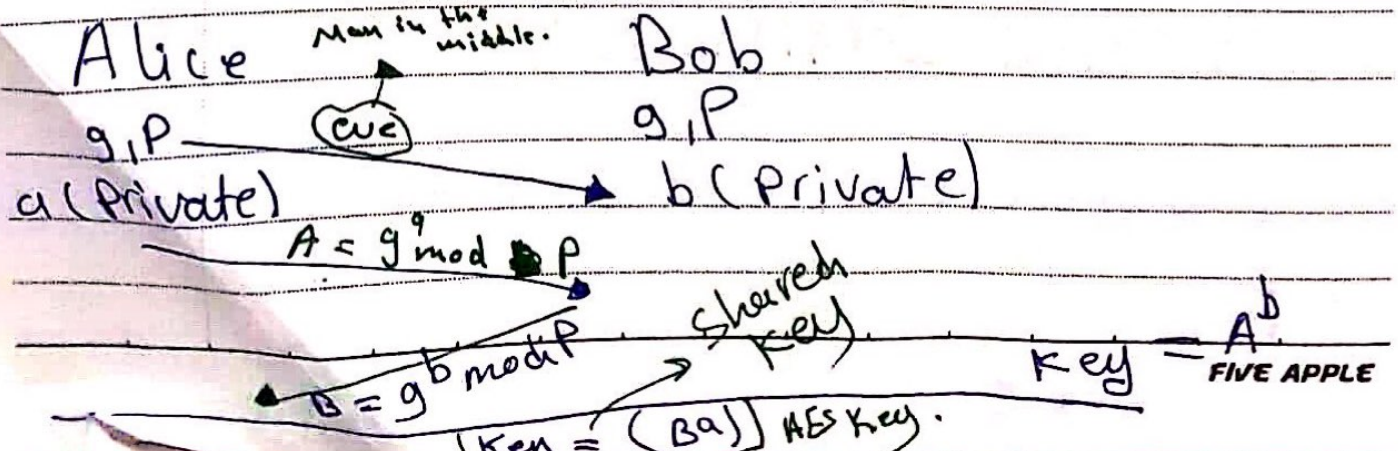
Alice | Bob

share a key →

- Diffie and helman is based on Discrete logarithm one way trap door function.

⁂ - $A = g^a \bmod P$.

it is infeasible to know a given A, g, P.
it is easy to calculate A given a, g, P.



→ If You can find any polynomial time solution to one of the NP comp problems →

You can Solve ~~NP complete~~ all of them.

[M, C]
↳ if they are very small , large.

Alice ——— $E_{PRA}(E_{PU}(key))$ ———→ Amazon
Amazon.
Random = key

Alice
PRA, PUA ——— E_{PRA} ———→ announ.

Eve
R_{AA}, PU_A ——— $E_{PP'A}$ ———→ Announ.

#Inced certificate for my Public Key

[Digital certificate].

$E_{PR_{CA}}$ [H(Alice, PUA, (Alice, PUA)]

↳ certificate Authority

FIVE APPLE

*Authentication

- Mutual authentication: The two parties don't know each other, and want to make sure of identity of each other

-Unidirectional authentication: A knows B, But not the inverse. Like log in to a website

- Sign-up: Identification to register yourself

-verification: Introduce something to be compared with the evidence you submitted at identification phase

* Types of authentication:
  - ~~shoulder surfing~~
  - ~~easy to install & use~~
  - ~~strong Password (need more restrict~~
  - ~~used many times~~
- Something you know : Password, Pin code, key
- " " are : Finger print, face, iris, DNA, voice
- " " have : Cards (ATM, Student), Dongles
- " " do : key stroke dynamics : type behavior, mouse movements

~~Password : shoulder surfing~~

know : ⊖ shoulder surfing, Sticky note, stored passwords

Strong Pass, can be guessed or ~~stolen~~ stolen, used many ti

⊕ easy to install & use

⊕ suitable for remote authentication

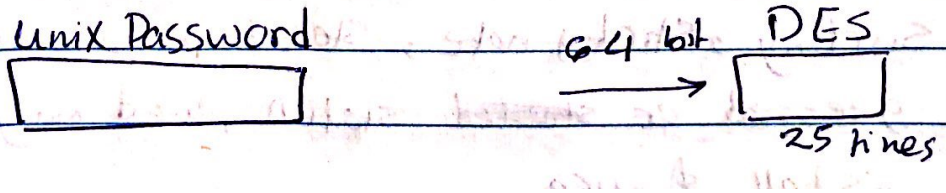
are: ⊕ Hard to guess or steal ⊖ Not convinent
⊖ False positive or negative
⊖ Need special HW (cost)

Have: ⊖ can be lost or stolen
⊖ Cost
⊖ Special HW

DO: ⊕ active authentication
⊕ No need for special HW
⊖ Pulse positive & negative

• Online attack:

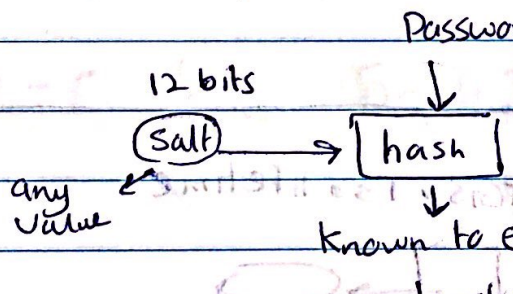
• Offline attack: you can brute force attack against a password file and compare them



- Syskey: you can encrypt the SAM file, and the key can be on a USB disk

- Dictionary attack: you download a dictionary
  - it may include all popular passwords
  - english words
  - you can add information about the user to it

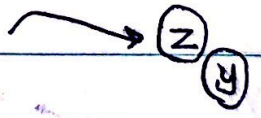
- Rainbow table: contains ~~of~~ hashes of all passwords



Rainbow table is much larger with 12 bits
⇒ 2^{12} * old rainbow table size

- One time password: Password that can be used only once

a = random
b = hash(a)
c = hash(b)
⋮
z = hash(y)



log in with z → z
login with y → x

- Two parties want to communicate with each other Securly. Mutual authentication
  - Public key
  - Trusted third Party (symetric key)