(S) ——M——→ (R) Is M unchanged.

- Can the receiver distinguish if M was changed to M'
- Or was M' itself sent by the sender.

(Unless there is only 1 message allowed to be sent)

→ Consider digital signatures $\langle m, \sigma_A \rangle$

Any change to either in the pair should indicate tampering.

- If $\langle m', \sigma_A' \rangle$ passes the test, then this breaks
- Further if $\langle m', \sigma_A \rangle$ passes then also breaks.

∴, No perfect scheme.

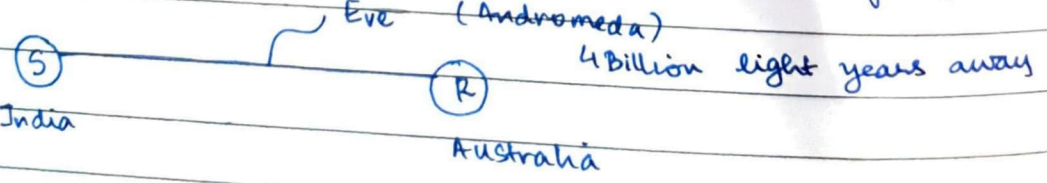
[Q]: How does one 'logically' circumvent a logical impossibility?

[A]: Bring in another impossibility (to destructively interfere with the said 'impossibility.')

[Metaphor: Jerry vs Tom impossible. Jerry vs Spike impossible. But All three ⇒ Jerry safe]

Then you get a fascinating solution to the problem.

- eg: - We want security for 4 Billion years.
  - Information can travel faster than speed of light] impossibilities



Let time value of message be 1 year

→ Secure communication not possible

→ Information cannot travel faster than light] destructively interfere

Simplest protocol in this case: Simply send message.

- Coding th
- Distributed Systems

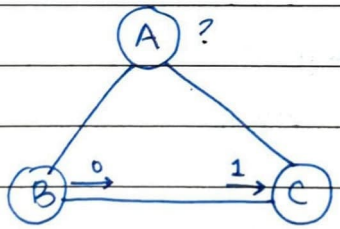
2 methods of impossibility
→ Natural limits
→ Computation hardness
→ Practical uncertainties

Eg:→ Universe is continuously expanding
Messages cannot travel faster than light.
Secure communication not possible.

Hamming distances. no. of mismatches occurring in 2 strings of the same position

Consider a coding theory problem
→ it says it is impossible to go beyond singleton bound
→ this is now any I.S problem.
→ Digitial signatures can be hashed & signed.
→ 2 impossibilities now destructively interfere.

Eg: Impossibility : If there are 3 people & one is a liar, it is impossible to localize that person
A, B, C ... 3 people



A claims ~~at~~ at least one of B/C is a liar

- Suppose digital signatures exists, this conundrom of A can be broken.
- Soln: sign the message & send.

Became famous when you had to build peer-to-peer systems
Gives way for decentralizing tasks itself

- Ways to list out impossibility:

1. Computation Hardness
2. Practical Uncertainity (Eg use UDP instead of TCP (User-datagram protocol) impossible to predict
Race conditions in OS.
3. Natural Limits. (Noone can predict whether schrodinger will be dead or alive
4. —

1-1-20

Intro to Modern Cryptography
- J. Katz & Y Lindell

→ Review :

- Problems :- logical No Go (fantastic)
- Solution :- Impossibility interference
- Science of IMPOSSIBLE ! ☺

- Caesar Cipher:

Follows a rule of shifting each character by 3

eg : a → D
b → E
etc.

crypto FUBSWR.

- Kerckhoffs Principle :

Security (system) does not depend on the obscurity of the algorithm but only on the secrecy of the key

[Passwords in machines are stored as hash.
Incorrect :- Noone knows the hash algorithm ⇒ secure
Correct :- No one knows the key/password]

Situation :- x is the password
$h(x)$: hash
h : function
] known.

Kerckhoff states that keeping h hidden is not sufficient for security.

Eg: $h(x) = 5x + 7$
$h(x_1) = 5x_1 + 7$
$h(x_2) = 5x_2 + 7$
⋮

Easy to see that
$h(x_1) - h(x_2) = 5k$

(1) Reverse engineering is easy for the algorithms.

Eg: Caesars Cipher :- $(x+3) \bmod 26$

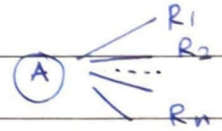
(2) Updation : complexity: If ever the key is revealed, recovery is quickly possible by just changing the key. However if it depends on the algo, the entire algo will have to be re written & re-compiled.

(3) Secure storage is costly : Every bit stored in secure memory should be worth the bit.

Eg: Storing a 128 bit key $\Rightarrow$ 2^{128} possibilities.
However storing 128 bit C program $\not\Rightarrow$ 2^{128} C programs
ie: Ineffective use of storage

→ Claiming that noone should know the hashing algorithm
→ Claiming that everyone knows the universal TM & password has 2 parts $U_{TM}(h, x) = h(x)$

(4) Multiple stakeholders becomes problematic
Different protocols for each routes will lead to immense storage.
(A) needs to remember all the protocols.
Easier way is having different keys.



ETHICAL HACKING:

Eg: CBI is running an ATM. Uses a proprietary algorithm for security.
https is secure if s is standard.

$x = 1 \rightarrow 5(1)+7 = 12$
$x = 2 \rightarrow 5(2)+7 = 17$

$17 - 12 \% 5 = 0$
$[h(x) - h(x_1)] \% 5 = 0$
$5(k) + ?$

classmate
Date
Page

→ **Shift Ciphers**

Overcoming the Kerchoffs problem in Caesars Ciphers.

(A) $\xrightarrow{(x+k) \bmod 26}$ (B)
K K.

Where k is the key.
key size : $|K| = 26$

★ Brute Force Attack : works if key space is small.
[Principle of sufficiently large key space]

$\{a, \ldots, z\}$ $x+k \bmod 26$

Ciphers text Plain text

If a occurs k_1 times in PT, then some other char occurs k_1 times in CT.

Let p_i denote probability of occurance of i^{th} char.
& q_i frequency of occurance of CT.

$\therefore q_{i+k} \approx p_i$

Known fact : $\sum_{i=1}^{26} p_i^2 = 0.038.$ 0.065

$= \sum_{i=1}^{26} q_i^2 = \sum_{i=1}^{26} p_i^2 = \sum_{i=1}^{26} p_i q_{i+k}$ ✓ k, ✗ k ; 0.065, $\frac{1}{26}$

Any secure encryption scheme must have a key space not vulnerable to exhaustive search.

~ **Monoalphabetic Sub. ciphers**

(A) ——— (B)

Create a key that is a permutation of the english alphabet

a b c ... z
↓ ↓ ↓ ↓
x d f k.

key space 26!

Brute force attack not possible (easy)

Larger key space is a necessary condition for secure communication But not sufficient

Another attack stemmed up. (probability distribution of each letter in english alphabet is known beforehand)

Now with the same previous concept,

$p_i \approx q_j$ where $j = i + \text{some}(k)$.

$$\sum p_i^2 = \sum q_i^2 = 0.065$$

Arrange pis & qis in a sorting order

ie: p_5 p_{20} .... → desc.

q_7 . . . . → desc. (match)

★ Frequency based attack.

m a l l i k a

c a t c a t c

16 2 14 15 10 13 4

p c w o j n d

~ Vigenere Cipher:

Choose a keyword, say cat

And word to be encrypted is crypto

c r y p t o g

c a t c a t c

Add % 26

Now Brute force ✗ , Frequency ✗ [∵ not all chars of same type mapped to the same char]

→ No: of occurances of a char in plain text cannot be meaningfully mapped to some other char.

→ Frequency is garbled.

★ Break: < with length known / length unknown.

(i) Partition cipher text into periods of l (key length).

Eg:- C_0 C_3 C_6] essentially like a collection of unknown no: of shift ciphers.

Security of Vigenere cipher now depends on the length of the keyword.

(ii) If I guess length to be some l

C_0, C_l, C_{2l} If l is correct ∵ shift ciphers

∴, $\Sigma p_i^2 = \Sigma q_i^2$

Index of coincidence

If it close to $1/26$ → guess is wrong ✗

If close to 0.065 → guess is correct. → collection of shift ciphers → ✓ Broken

NOTE: 3 steps in Encryption that take place

① Gen: key generation algorithm probabilistic algo

② : Enc: Encrypts the plain text.

③ : Dec: Decrypts the encrypted cipher text.

Breaking the Vigenere Cipher without knowing the length of the keyword :-

Kasiski : If a string of alphabets appears repeatedly in a polyalphabetic ciphertext msg, it is possible that the distance b/w the reoccurring character is a multiple of the length of the keyword.

→ GCD of all distances b/w repeated sequences should yield the period length t.

→ A large ciphertext is required for determining the period if Kasiski method is used.

Enc: $\mathbb{M} \times \mathbb{K}$

classmate
Date
Page

- **Shannon's Approach:**

Claim: Patching & breaking schemes is indefinite process.
∴, He decided to approach this claim from the ∞.
Can we define a complete secrecy condition?

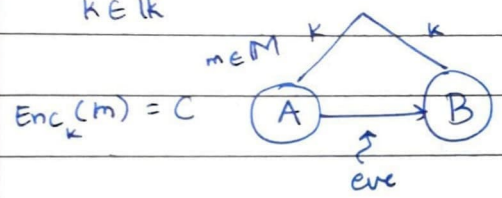
→ Vernam Cipher (One-time pad).
- Meets Shannon's defn
- Has some impracticality

~ **Shannons Defn for perfect Secrecy:**

Initial Model: 4-tuple $\langle Gen, Enc, Dec, \mathbb{M} \rangle$

$Key \leftarrow Gen(1^n)$ $\quad Dec_k(Enc_k(m)) = m$

$K \in \mathbb{K}$



→ $Enc: \mathbb{M} \times \mathbb{K} \longrightarrow \mathbb{C}$

Encryption algorithms map from $\mathbb{M} \times \mathbb{K} \rightarrow \mathbb{C}$
Once $\mathbb{K}$, $\mathbb{M}$ & Enc are fixed $\mathbb{C}$ also gets fixed.

→ No matter what you know a priori ≠ what you know before seeing the ciper text is what you know after seeing the cipher text.

'An encryption scheme $\langle Gen, Enc, Dec, \mathbb{M} \rangle$ is said to be perfectly secret if for all probability dist. over $\mathbb{M}$ for all $m \in \mathbb{M}$ for all $c \in \mathbb{C}$: (assume $P[C=c] > 0$)

length of ciphertext

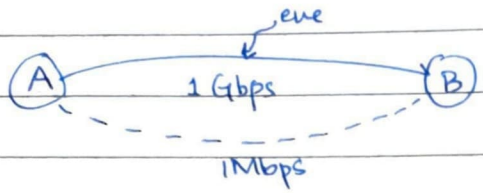
then

$$P[M = m \mid C = c] = P[M = m]$$

Assume K shared b/w

- **Vernam Cipher:**
  - Gen: $K \xleftarrow{R} \{0,1\}^n$: choose a random n bit key $\left(\frac{1}{2^n}\right)$
  - Enc: $c = m \oplus k$ (Bitwise XOR)
  - Dec: $c \oplus k = m$

Limitations:



- key is to be sent via the --- back channel, so that it is known to A, B.
- One time pad: can use the key only once.
- Speed of transfer = Speed of your slow channel
- security bandwidth of your transmission will be equal to only to the speed of your secure channel.

JOKE :) : you could have sent the msg itself avoiding overheads.

Use cases: • When user = receiver A = B
• You use only 1 channel (secure/insecure) at a time.

Q: How to do the following:-

① ? Low speed ~~in~~secure ch. + High speed insecure ch. = High speed secure ch.

Circumventing this impossibility → Symmetric key Cryptography

② From no sec. channel → Obtain a slow secure channel → Public key Cryptography

Pick any 2 msgs m_0, m_1 $\Pr[m_i] = \frac{1}{2^n}$ always

Proof: To show: Vernam Cipher is perfectly secret.

$\forall$ prob dist over $\mathbb{M}$ $\quad \forall\ m \in \mathbb{M}, \forall\ c \in \mathbb{C}$

$$P[C = c \mid M = m] = P[C = c] \quad \text{— (1)}$$

to show this is same as perfect secrecy axiom

$$\Rightarrow \frac{P[C=c \mid M=m]}{P(C=c)} \cdot P[M=m] = \frac{P[C=c]}{P[C=c]} \cdot P[M=m]$$

$$= P[M=m \mid C=c] = P[M=m]$$

To prove both ways

$\forall$ prob dist over $\mathbb{M}$

$\forall\ c \in \mathbb{C} \quad \forall\ m_0, m_1 \in \mathbb{M}$

$$P[C=c \mid M=m_0] = P[C=c \mid M=m_1] \quad \text{— (2)}$$

(1) $\Rightarrow$ (2) is trivial.

(2) $\Rightarrow$ (1) $\quad P[C=c] = \sum_{m \in \mathbb{M}} P[C=c \mid M=m] \cdot P[M=m]$

($P[C=c \mid M=m]$ = p)

$$P[C=c] = p \sum_{m \in M} P[M=m] = p$$

Consider the Vernam Cipher case:-

$$P[C=c \mid M=m_0] = P[c = m_0 \oplus k] = P[k = c \oplus m] = \frac{1}{2^n}$$

$$P[C=c \mid M=m_1] = P[C = m_1 \oplus k] = P[k = m_1 \oplus c] = 1/2^n$$

$\because$ k was uniformly chosen from $\{0,1\}^n$

Thm: Every scheme that meets the perfect secrecy condition has the limitation that $|K| \geq |M|$

Message space should be at least as large as the key space
key space message

Proof: Suppose not.

Let $|M| > |K|$

Fix some cipher text c

Let D be set of all decryptions of c

ie: $D = \{ m \mid \exists k \ m = Dec_k(m) \}$

$|D| < |M|$

$\exists \ m^* \in M$ but $m^* \notin D$

Considering $\forall$ probability distributions

Eg: uniform dist.

$P[M = m^* \mid C = c] = 0$

But $P[M = m^*] \neq 0$

no: of bits required to send key $\geq$ no: of bits req. to send message

- FIESTEL CIPHER:

→ based on ppl of Shannon: Diffusion & Confusion

→ Used to create a SPN (Sub. Permutation Network)

n = no: of rounds F: Fiestal function

Diffusion :- make the rel b/w cipher & plain text as complex as possible.

Confusion :- make the stat rel. cipher text & encrypted key as complex as possible.

By using complex substitution algorithm

→ DES: Data Encryption Standard :
[A symmetric key encryption]

Input: 1) 56-bit initial key $k \in \{0,1\}^{56}$
2) P = plain text block = 64 bit $\in \{0,1\}^{64}$.

Output: Ciphertext = C = 64 bit $\in \{0,1\}^{64}$
Will convert to 64 bit data for perfect secrecy

Consider plain text :-

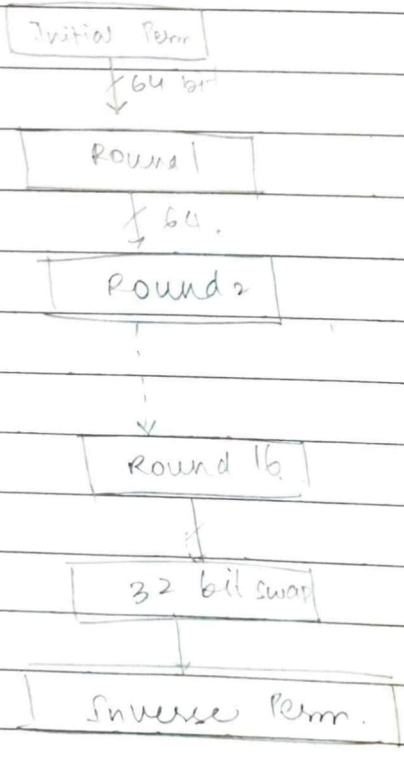
00101 | | |

64 bit ⇓ 64 bit

convert to cipher text

append 0

If message size > 64 bits divide into blocks of 64 bits



→ 1234
2341
4123
3412

Where Permutation of a set S: $S \to S$ bijection.
Eg if $S = (a_1, a_2 \ldots a_n)$ $p = \begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ p(a_1) & p(a_2) & & p(a_n) \end{pmatrix}$

A small change in the plain text/key should create a significant change in the ciphre text
→ DES proved to be strong with this prop

Inverse is computed via this bijection.

How to convert 56 to 64 bit key.
K is converted to 64 bit key packed with 8-bits of parity.
By this obtain 16 round keys $K_1, K_2 \ldots K_n$

Rule: while doing circular left shift
if (round == 1, 2, 9, 16) rotate 1 bit
else rotate 2 bits

Consider Round i in DES :-

21-01-2020

classmate
Date
Page

Perfect security
→ forever, infinite sec.
→ Zero error

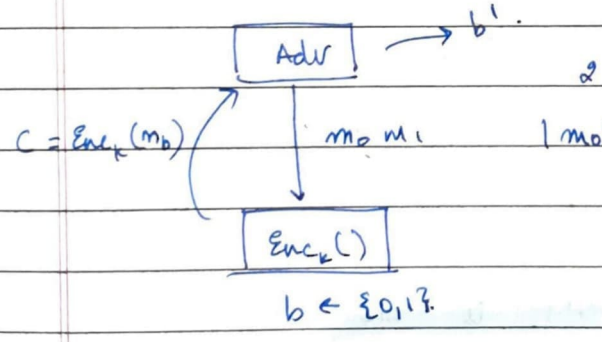
→ Topics :-
- Two famous relaxations
- One famous assumption

- #PPTM A: Probabilistic Polynomial Turing Machine.
with security parameter k: & sufficiently large k,
∀ positive poly^n p
$\exists$ k_0 s.t. : ∀ $k \geq k_0$

$$P[A \text{ can break } \pi] \leq \frac{1}{p(k)}$$ negligable.

PPTM :- ∀ n person/adversary is

Ciphertext only Attack:



$Gen(1^n) = k$ ... generates the key.

Acc. to Shannons Th. ∀ PTM A: $P[b' = b] = 1/2$.
Now, considering PPTM $P[b' = b] \leq \frac{1}{2} + negl(n)$

A function $\mu(n)$ is said to be negligeable in n if for all +ve polynomials $p()$, $\exists$ n_0 such that ∀ $n \geq n_0$

$$\mu(n) < \frac{1}{p(n)}$$

① Heuristic Security:

Given the algo assume that it meets the definition of security ∵ no one is able to disprove this.

Eg: Flipkart Amazon etc. use the AES scheme.

② Provable Security:

Make 2 assumptions: efficient adversary, negligable errors

Conjecture that some mathematical object exist

[Eg: Pseudo Random generator exists].

[Start off saying true] mathematical assumptions only.

If a secure encryption scheme was built assuming π_1 & another assuming π_2, then necessarily $\pi_1 \Leftrightarrow \pi_2$

All assumptions are equivalent.

Assumption chosen today: One-way functions exist.

③ Proven Security:

→ Message authentication, digital signatures etc will have secure algorithms if one-way functions exist. [2 relaxations included].

* A function $f: \{0,1\}^* \rightarrow \{0,1\}^*$ is said to be a ONE-WAY function if:

a) Easy to compute : $\exists$ TM M ~~given~~ halts given that it starts with x on its tape.

b) Hard to Invert:- $\forall$ PPTM A.

$$P\left[A(f(x)) = y \mid f(x) = f(y)\right] \leq negl(x)$$

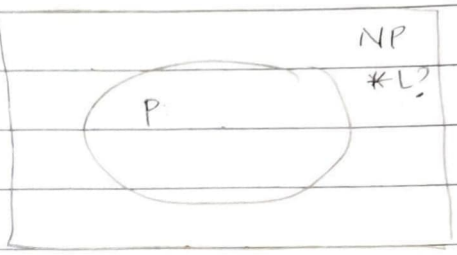
negligible probability of success

→ P versus NP

$L \in NP$

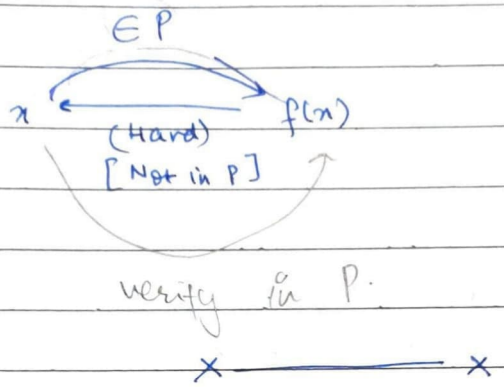
$L \notin P$.

[Reconfirm] ??



Theorem: If one-way function exists then $P \neq NP$

If:



Given $f(x)$, what is x ?

Review: → Two relations:-
- computationally bonded adversary
- Negligible prob. of error.

→ One-way func. exists.

• Discrete logarithm Problem [DLP]

Input : Prime p & generator g & $y = g^x \bmod p$

Output : x

If g is a generator $\exists$ a unique x in $1 - (p-1)$ that gives $y = g^x \bmod p$.

→ Conjecture : this is a one way func.

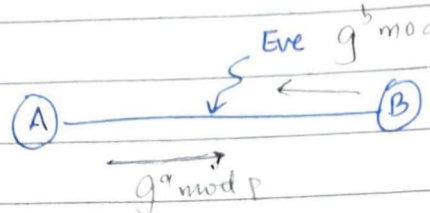
$f(x) = g^x \bmod p$... satisfies dfn of one way

Security parameter : $\log p$.

If p is not prime, generator may not exist.

Z_p^*

- **Diffie Hellman key Exchange Protocol**



A chooses a random el. $a \in_R \{1 \cdots p-1\}$

b picks a random el $b \in_R \{1 \cdots p-1\}$

A computes the key as:

$$K = [g^b \bmod p]^a \bmod p \qquad K = [g^a \bmod p]^b \bmod p$$

There are no efficient methods to compute $g^{ab} \bmod p$ given $g^a \bmod p$ $g^b \bmod p$.

No efficient adversary can distinguish g^{ab} from a random r beyond a ~~with~~ negligible advantage.

- **Pseudo Random Generator (PRG) :**

A deterministic program $G : \{0,1\}^n \longrightarrow \{0,1\}^{l(n)}$ is a PRG if:-

(a) Expansion condition $l(n) > n$

(b) Pseudo randomness $\forall$ PPTM D

Should be indistinguishable from random generator by an efficient algorithm.

$$\left| P\left[D(U_{l(n)}) = 1\right] - P\left[D(G(U_n)) = 1\right] \right| \leq \text{negl}(n)$$

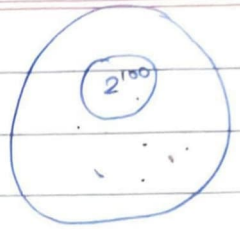
eg: let $n = 100$, $l(n) = 1000$

2^{1000} strings possible.

CASE1: $\frac{1}{2^{1000}}$

Computational hardness
Computa

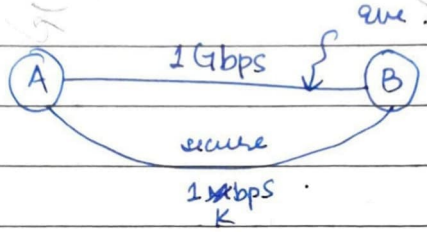
CASE 2: You pick a number from the subset 2^{100} And then apply some G



You should not be able to distinguish between these two worlds. [efficient distinguisher].

Analogy: A is weak B is stronggg C is stro~~ggggggg~~
A cannot differentiate.

Consider



With a PRG :-

- A sends a key k via 1 kbps
- Both A & B apply $k = G(k)$. [l(n) expansion]
- Now apply one-time-pad $m \oplus k$
- Even though secure channel is slow, expand it on both ends.
- Adversary cannot distinguish k from a true random no:
- G has a deterministic program.

(!) We do not know whether PRG exists.

Let if possible PRG exists :-

$G(s) = y$

The function of getting s looking at the left half of y should be a hard problem

∴ looking at the right half, you have no idea about the right half.

If we have a one way func ⇒ can construct PRG

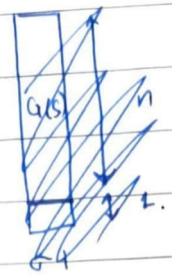
Creating a Pseudo Random Generator:

Claim: Assuming DLP is hard, I can generate random n

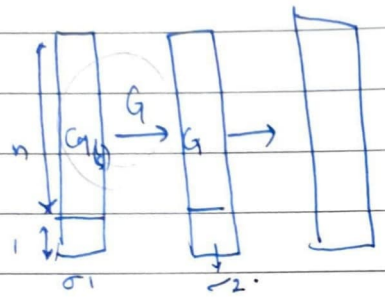
Consider: $G: \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ [minimal expansion]

$G': \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ .... is possible.

(Reduction) Define $G'(s)$: [s: seed]



$G'(s)$:



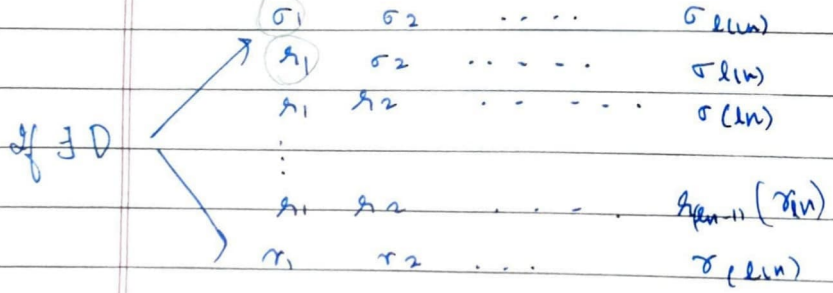
Apply this $l(n)$ times

You get $l(n)$ bits

If $G(s)$ is PS $\Rightarrow$ $G'(s)$ is PS.

Suppose $\sigma_1 \sigma_2 \ldots \sigma_{l(n)}$ is not pseudo random

Truly Random $r_1, r_2 \ldots r_{l(n)}$.



If ∃ a D ST it can distinguish b/w the two worlds then there must exist

i^{th} row distinguish from $(i+1)^{th}$ row

Suppose not, then $R_1 \not\stackrel{c}{\approx} R_2$ $\quad R_2 \not\stackrel{c}{\approx} R_3$

ie: $R_1 \not\stackrel{c}{\approx} R_3$

$\Rightarrow R_1 \not\stackrel{c}{\approx} R_{len}$

How? Select R_i, R_{i+1}. $\in_R$

these are identical in all but i^{th} position.

If you can distinguish b/w σ_{i+1} & R_{i+1} then you got a distinguisher for G.

But G is PRG $\Rightarrow$ no. distinguisher D exists.

$\therefore$ # contradiction.

→ Showing +1 expansion:

$$f(x) = g^x \bmod p$$

Given $f(s)$ you cannot find s

$\therefore$ Among all predicates $s.s.$ Hard core predicate

$$f(x) = g^x \bmod p$$

$$G(s) = g^s \bmod p \,\|\, h(s)$$

(single bit of s that is hardest to find.)

To prove: MSB is the hardest.

Algo: If you can predict the MSB of s, you can predict all other bits

DLP: $g^x \bmod p \longrightarrow x$?

DLP*: $g^x \bmod p \longrightarrow MSB(x) = ?$ or Is $x > \frac{p-1}{2}$

To show If $\exists$ an algo for DLP* $\exists$ an efficient algo to crack DLP**

One time pad is CPA secure.

Construct a poly^n time algo $DLP \leq_P DLP^*$

Proof: ① If $g^x \bmod p$ → what is $LSB(x) \in P$ exists
② if $\exists$ $g^x \bmod p$ → what is $MSB(x)$ exists then x can be fully found.

$\in DLP \in P$

①: Suppose compute $y^{\frac{p-1}{2}} \,\%\, p$

$= g^{\frac{(p-1)x}{2}} \,\%\, p$

LSB(0) → 1

LSB(1) → $= -1$.

Consider

$g^x \bmod p$

LSB = 0 | LSB (1)

sqrt $g^{x/2} \bmod p$ | $g^{x-1} \bmod p$

$g^{x/2} \bmod p$, $g^{x/2 + \frac{p-1}{2}} \bmod p$

LSB = 0 | LSB = 1

→ Review :-

- PRG
- $Enc_k(m) = G(k) \oplus m$

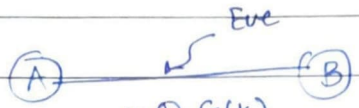
Works as long as new parts of key k are used.

→ Today's Topics :-

- Chosen Plaintext attack [CPA].
- Designing CPA secure

One time-pad: as long as you're using new fresh parts of the $G(k)$ generated.

But now A & B have to maintain state (drawback)



Maintaining state → bad option.

G(k) G(k)

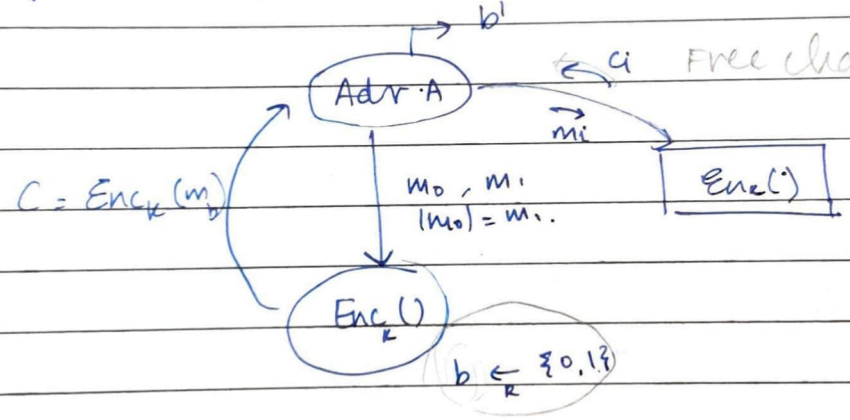
G(k) = |m1| |m2| |m3|

→ Chosen Plaintext Attack :- (secure algo)
Ensure it will remain secure even if the adversary has the encryptions of chosen plain text messages.

↓ chosen by adversary

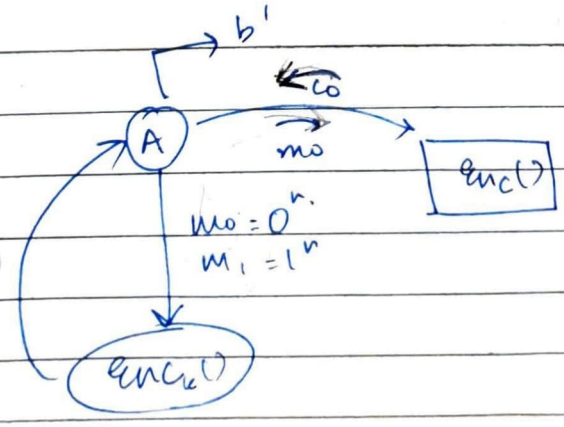
$m_1 \rightarrow c_1$
$m_2 \rightarrow c_2$
$m_3 \rightarrow c_3$

Scenario :-



Yet $P[b = b'] \leq negl(n)$

Eg:



$Enc_k(m) = m \oplus Gen(k)$
Doesn't work.

Prob $b' = \begin{cases} 0 & \text{if } c_0 = c \\ 2 & \text{if } c_0 \neq c \end{cases}$

Stream cipher $G(I_v, k)$
Initialization vector

No deterministic algo is CPA secure

The enc. algorithm is deterministic. for the same key k & message m, the c will be the same.
∴, It isnt CPA secure.

→ If you use PRG & use it like one-time pad it will be CPA secure. But we have to keep track of states

Thm: No deterministic $Enc_k()$ is CPA secure

Decryption algos have to be deterministic.] meets
Encryption algos are probabilistic.

Idea: Don't even encrypt the message 😉.

→ Consider an encryption algo $Enc_k()$ that is deterministic
Define $Enc_k : \{0,1\}^n \longrightarrow \{0,1\}^n$ (block cipher)
Probabilistic Enc

$r \in_R \{0,1\}^n$

cipher $C = \langle r, m \oplus Enc_k(r) \rangle$
(r: noise)

length doubling encryption

$Dec(u, v) = Dec(r, v)$

$m = v \oplus Enc_k(r)$

Adversary cannot break in
∴, $\langle r_b, m_b \oplus Enc_k(r_b) \rangle$

Through free channel :- he'll be obtaining random encryptions ∴, cannot distinguish b/w 2 messages.

→ This k is communicated once via the secure channel.

• Modes Of Operations (Block cipher) :-

1. Cipher Block chaining.
2. Output Feedback Mode.
3. Randomized Counter Mode.

→ Output feedback mode :-

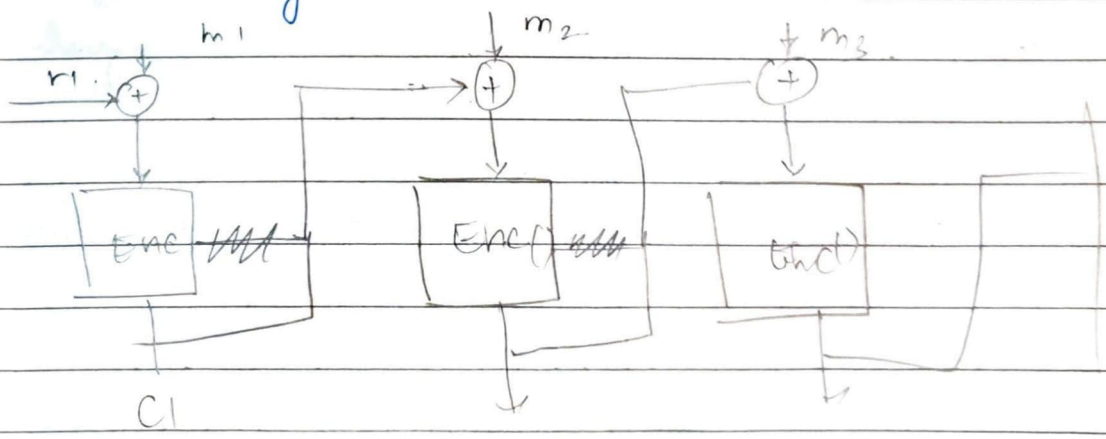
$$M = m_1 m_2 \ldots m_t$$
$$r_i = Enc_K(r_{i-1})$$
$$C = \langle r_0, V_1, V_2 \ldots V_t \rangle$$

Though not length doubling, It's time consuming.

→ Randomized counter Mode :-

$$r_i = r_0 + i$$
$$C = \langle r_0, V_1, V_2 \ldots, V_t \rangle$$

→ Cipher Block chaining :-



$$C_i = Enc_K(m_i \oplus C_{i-1})$$
$$\langle C_1, C_2, \ldots C_t \rangle$$

REVIEW

→ Probabilistic Encryption

→ CPA Study

→ ciphertext = $\langle r, F_k(m) \rangle$ → must be a true random function

what should F be?

$$F : \{0,1\}^n \times \{0,1\} \longrightarrow \{0,1\}^n$$

$(2^n)^{4^n}$ possible functions.

choose one of them uniformly and $\frac{1}{(2^n)^{4^n}}$ prob for each one

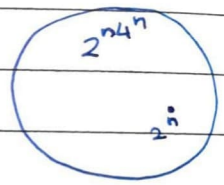
How many coin tosses $\log\left(2^{n4^n}\right) = n4^n$

∴ Algorithm is exponential with $O(n4^n)$

Lets say I do n coin tosses instead

→ Pseudo random function will take from 2^n space

→ Truly random picks from the entire space



→ If no efficient distinguisher can distinguish b/w the two, we can prove its CPA secure

CPA secure you need PRF
ciphertext you need PRG ?

PRF exist iff PRG exist. .... Thm.

easy

→ Here distinguishing the 2 worlds is possible ∵ the input in the real world will be an exponentially long string

A function $F_k : \{0,1\}^n \longrightarrow \{0,1\}^n$

takes an n-bit string

$\forall$ PPTM D, suff large n

Give D oracle access to the function

$$\left| P\left[D^{F_K(\cdot)}(1^n) = 1 \right] - P\left[D^{g(\cdot)}(1^n) = 1 \right] \right| \leq \text{negl}(n).$$

(1^n encoded in unary)

If such functions exist, then we know CPA secure encryption scheme.

Thm: PRF exist iff PRGs exist

$$\boxed{\begin{array}{l} \text{PRG} \Rightarrow \text{PRF} \\ G \to F. \end{array}}$$

Let $G: \{0,1\}^n \longrightarrow \{0,1\}^{2n}$ be a PRG takes n bit to $2n$ bit

$G_0(s)$ = left half of $G(s)$
$G_1(s)$ = Right half of $G(s)$

$G(s) = G_0(s) \,||\, G_1(s)$

truly random string

Define $F_k(r) = n$ bit string.

$r = r_0\, r_1\, r_2 \cdots r_{n-1}$ n bit string.

$$F_k(r) = G_{r_{n-1}}(\cdots(G_{r_1}(G_{r_0}(k)))).$$

Suppose we start with $r = r_0$.

apply G once

k

$G_0(k)$ $G_1(k)$

if $r=0$ G_0 is answer $r=1$ G_1 is answer.

$G_{00}(k)$ G_{01} G_{10} G_{11}

If F is not PR, at least one leaf node can be distinguished.

Move along the path in the tree to track $G_r(k)$
If such a leaf found ⇒ insecure at level just before this.

QUIZ:

(1) Number n is said to be b-smooth if the max prime factor of $n \leq b$

a) Show that DLP has an algorithm polynomial in $\log_2 b$ if $(p-1)$ is b-smooth.

b) Given x in base q ≥ 2 and if $p-1 = sq^r$, $q \nmid s$ $r \geq 1$, show that the $(r+1)^{th}$ digit of x is a hard core predicate of $g^x \bmod p$.

Given $p-1 = s2^r$ s is odd show that $(r+1)^{th}$ bit of x is hard core predicate of $g^x \bmod p$ using (a)

c) For DH SKE what kind of primes do you reccommend. Why?

(2) Show that for mono-alphabetic sub cipher $MSC(\cdot)$

→ a) MSC is perfectly secret if only a single character is encrypted (ie: $M = \{a, \dots z\}$.

→ b) $\forall$ $2 \leq k \leq 26$ show that $\exists M \subseteq \{a \dots z\}^k$ s.t
- MSC is perfectly secret over M.
- " " not " " " $\overline{M}$

→ c) $\forall$ $k > 26$, MSC is never p.s for M $\{$

BOF ATTACK:

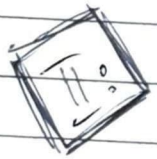
① Key pts for BOF :-
- Heap & stack opp directions
- Local var such as buffer is written low → high mem address. If buffer overflows, will corrupt return address.

② Shell code: If the attacker injects a malicious code in an array variable of the vulnerable program. Most obv. exploit involves spawning a shell.

③ Shift sys call no. to %eax then gen software interrupt

④ Attacker must check :-
- Shell code gets executed
- Return address overwritten by shell code add.
- Experimentally get top of the stack frame.

→ Thereby get top of the stack, address of shell code, return add. Inject shell code + overwrite return address



⑤ Uses existing code in LibC to spawn a shell.

return into libc

⑥ Start add. of every lib. function or a particular OS can be easily determined.

⑦ Makes a call to the system() library function which internally invokes execve syscall passing ("/bin/sh")

⑧ Address of exit() is placed below the address of system(). "/bin/sh" loaded somewhere in memory.

SIS exit address

⑨

BOF defenses:

→ Progrem / Lang
→ OS
→ Compiler
→ Hardware

i) Develop in type safe lang C# / Java
ii) Avoid libC func: gets sprintf, strcpy
iii) Audit C/C++ code use strncpy
iv) Some OS like Linux var, Windows XP make stack non executable. (doesn't prev. Return to libC)

v) At compiler level include a 32 bit rand (canary) placed between FP & return address. Before return check to see if modified

vi) Randomize layout of memory base address, lib functions all random

vii) Use safe C compilers check ret address but usually result in overheads

viii) Hardware opt: store ret address in register instead of stack

FORMAT STRING ATTACK.

→ Use the format specifiers to read & write arbitrary locations in memory.

→ SQL INJECTION ATTACK :-

→ Multi tier web apps : web, application & the database
→ In some apps with limited business logic, the web tier & app fused. web directly interacts with db
→ Consider a form being passed as a POST request. form param passed ~~as body~~ by HTTP: post or as a query string in URL.

● Defense :-

→ Prevent the usage of chars like ; " "
built = cannot be avoided.

→ Allow certain types of inputs based on regex expression evaluation.

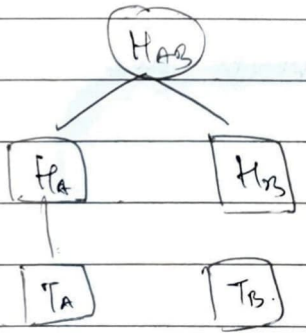
BLOCKCHAIN

→ Block chain : chain of blocks storing potential information.
→ Blocks of digital info in a db (chain)
→ First block : Genesis block.

→ Adv :-
- no single owner hence decentralized.
- Data crypt. stored.
- BC is immutable

→ Main ideology behind bitcoin, no 3rd party involvement.

→ Merkel Tree Root :-



T_i : i^{th} transaction
H_i : Hash of i^{th} transaction
$H_{AB} = H_A \oplus H_B$

DES:

→ Diffusion :- make stat rel between the plain text & cipher text as comp as possible
- Permuting constantly then applying a permutation

→ Confusion : Make rel b/w ~~plain~~ cipher text & value of enc key as compl.
- complex sub algo

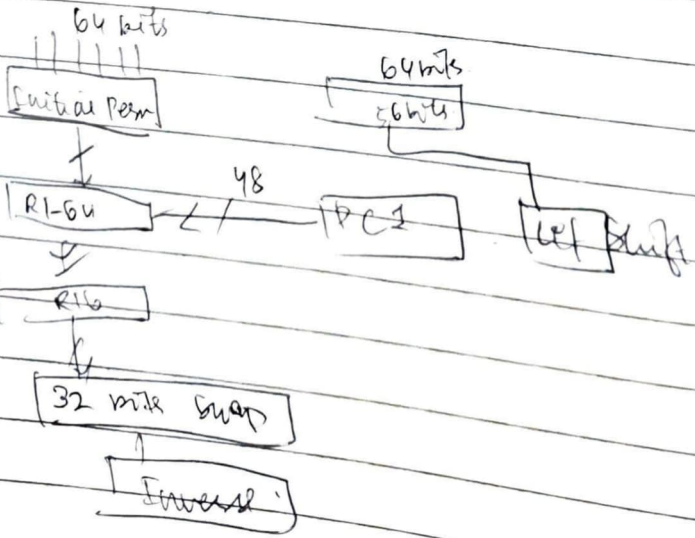
→ Feistal Cipher : modern day ciphers for block
- cipher
- conf & Diff.
- const sub perm network

→ DES:

- Data encrypted in 64 bit block using 56 bit key.

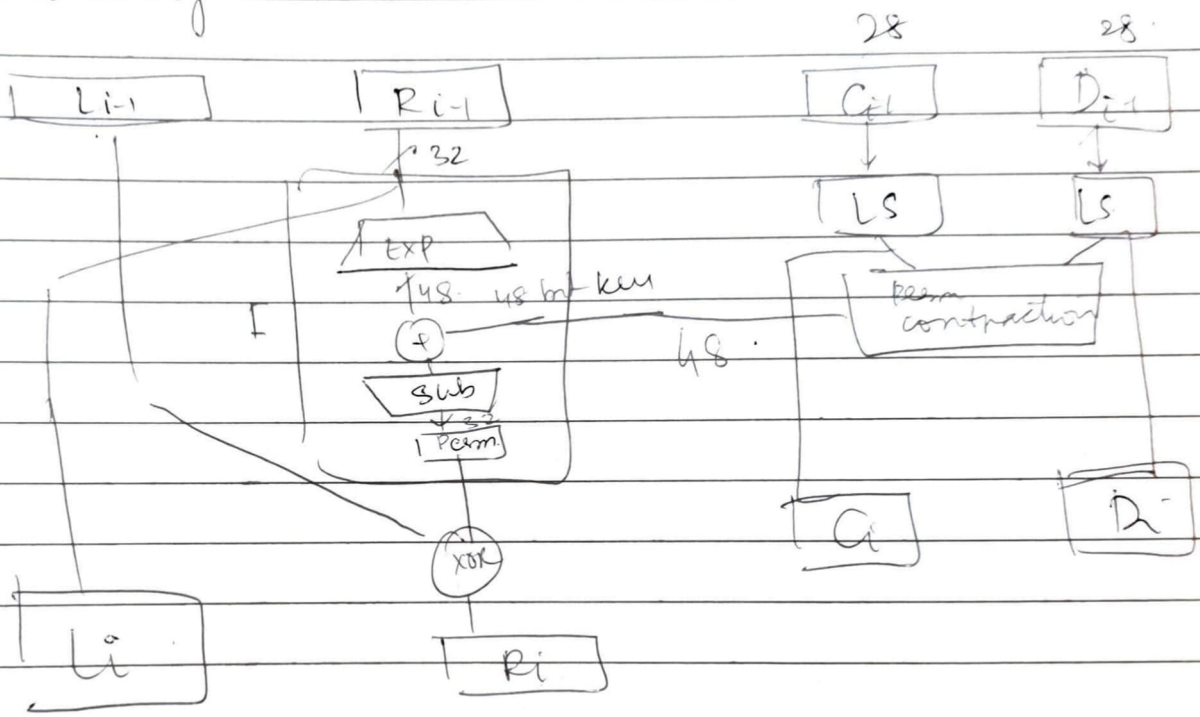
$$\{0,1\}^{64} \times \{0,1\}^{56} \longrightarrow \{0,1\}^{64}.$$

$$DES_k^{56}(P^{64}) = C^{64}$$



- 56 bit key passed with 8 parity bits at 8x positions.
- If R 1, 2, 9, 16 : rotate 1
  else " 2

→ 1 Round of DES :-



$$F(R_i, k_i) = (pS_i(F(R_i) \oplus k_i))$$

→ expansion box in Add 2 Right Ckt. | Sub 1 left ckt.

→ S box :-
1 & 6th bit ~~ckt~~ row rem cols.

→ Decryption function :- Use all 16 keys in r

ie:

$$DES_{16,15,14,\ldots,3,2,1}\left(DES_{1,2,3,\ldots,16}(x)\right) = x$$

→

$$\overline{DES(k, x)} = DES(\bar{k}, \bar{x})$$

Brute force attack to break comp reduces from 2^{56} to 2^{55}

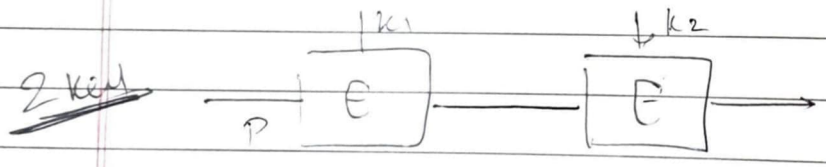
→ Avalanche effect in DES. small change in the plaintext results in very large chain in the ciphertext.

• Limitations :-

→ 4 + 12 + 48 weak keys

↓

all 00 FF 00FF FF00.



$$E_{k_2}\left[E_{k_1}(P)\right] = C.$$

$$E_{k_1}(P) = D_{k_2}(C)$$

- Store all $enc_{k_1}(P)$ → in table
- Calculate $Dec_{k_2}(C)$.

At each stage check if match found if so check for $\langle P', C' \rangle$.

- 2^{112} keys. 2^{64} plain text

2^{48} ~~double cipher text plaintexts~~

keys same cipher

∴ 2^{48} false alarms.

— With 64 more bits ⇒ 2^{-16} per alarm

∴ Prob correct = $(1 - 2^{16})$

→ Plaintext attack succeeds against 2DES order 2^{56}

~~3DES~~ Vul to meet in middle
$O(2^{120 - \log w})$

3DES with 3 keys
— No break found yet.
— Used in PGP pretty good privacy

- AES: Advance Enc. Standard.
  128 bit & key 128 p.
  Secure efficient.

DIGITAL SIGNATURES

$\langle P, A, K, S, V \rangle$ — sig, sig func

P: messages, K: keys, V: ver func.

$ver(x, y) = true$ if $y = sig_k(x)$, else false

$x \in P$, $y \in A$

- **DSA**: based on the difficulty of computing logarithms.

p: prime $2^{L-1} < p < 2^L$ $\quad 512 < L < 1024$.

q: 160 bit length prime div of $(p-1)$ $\quad 2^{159} < q < 2^{160}$

$g = h^{\frac{(p-1)}{q}}$ $\quad 1 < h < p-1$ $\quad h^{(p-1)/q} \bmod p > 1$

- Private key: x $\quad 0 < x < q$
- Public key: $y = g^x \bmod p$

$0 < k < q$ $\quad k \in RI$

★ signing :-

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

$s = [k^{-1} H(M) + xr] \bmod q$

Send $(M, (r, s))$.

★ Verification phase :-

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

$u_1 = [H(M') \cdot w] \bmod q$

$u_2 = r'(w) \bmod q$

$v = (g^{u_1} y^{u_2} \bmod p) \bmod q$

$v = r'$

RSA Based on the difficulty of computing the factors of a composite no: into 2 prime factors

- p, q ... prime & large.
- $n = p \times q$
- $\phi(n) = (p-1)(q-1)$
- e ... $\gcd(e, \phi(n)) = 1$ $\quad 1 < e < \phi(n)$
- $d = e^{-1} \bmod \phi(n)$.

$KU = \{e, n\}$

$KR = \{d, n\}$

$x \longrightarrow$ CP message

Sig Generation :-

$$y = sig_K(x) = x^d \pmod n$$

Verification $\quad x' = y^e \bmod n$

Valid if $x = x'$ $\quad$ else ✗

Verification phase :-

$$y^e \bmod n$$

$$(x^d \bmod n)^e \bmod n.$$

$$= (x^{de} (\bmod n)) \bmod n$$

$$= x.$$