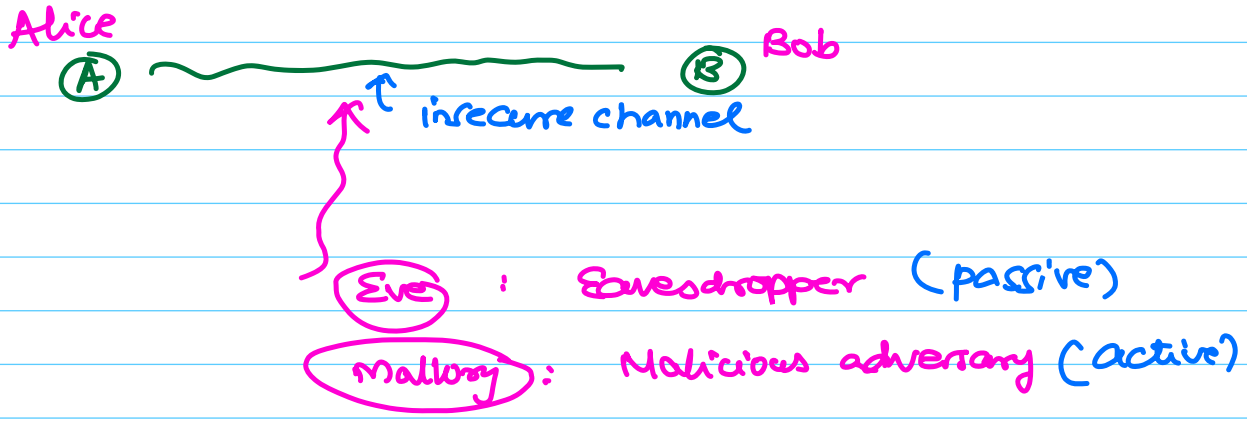
3rd Jan 2025

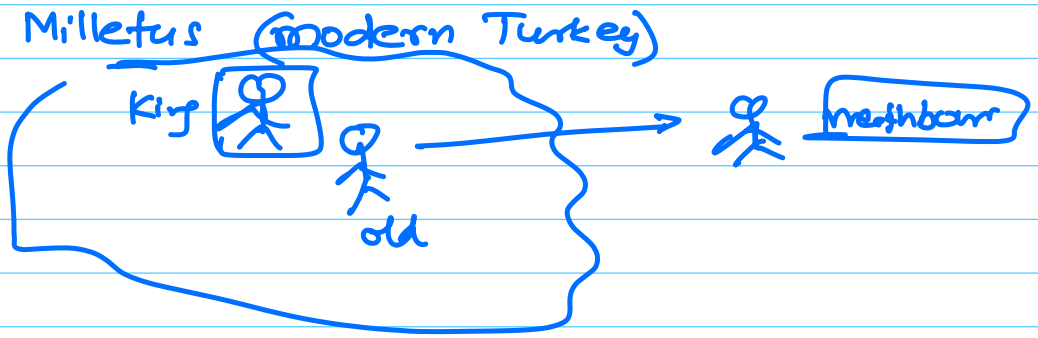
Cryptography - crypto + graphos

Science of secret communication

where is it needed -

diplomatic missions, soldiers, kings lovers, e-commerce





Slave — Shaved the head

Vietnam war — US army Jeremiah Denton captured by Guirillas

BBC Journalist — video interview

— — = 0

— —— = 1

Morse code . —

Spelled "TORTURE"

YouTube video

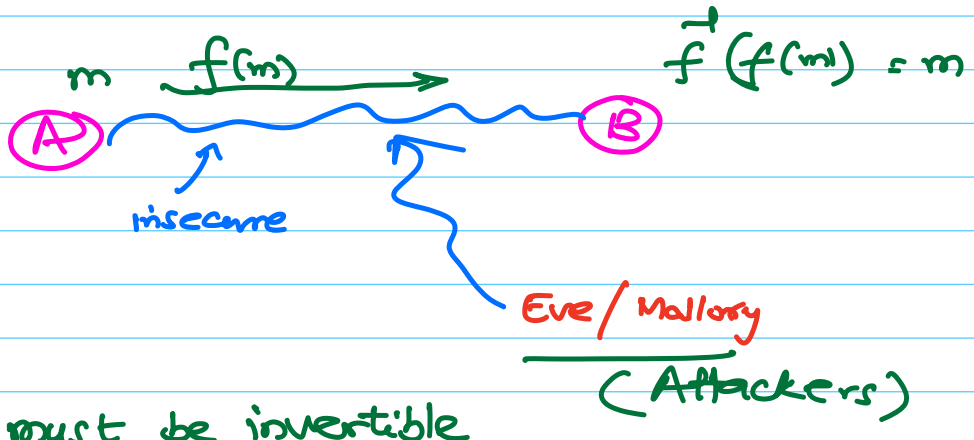
2nd war : invisible ink

MI-2 o — Microdot YouTube

Steganography - "hide the method"

Cryptography - method is open
- yet secure communication

Setting :



— method f must be invertible
— Alice knows f, Bob knows f^{-1}
But Attacker doesn't know f^{-1}.

Two way communication -
A, B : f, f^{-1}
attacker : doesn't have f^{-1}

Hiding the method

CS: Algorithm, steganograph Data → cryptography

Without keeping "something" secret between A & B
— no secure communication is possible.

Secret data = secret key (K)

- f: Encryption
- f^{-1}: Decryption
- k = secret key
- m = message / plaintext
- c = cipher text

Secret key → Encryption → Secret key Cryptography

What is meant by "secret communication" ...

Historical examples.

Julius Caesar -

A, B, C, ... Z
0, 1, 2, ... 25

$\mathbb{Z}$ = set of integers $\quad \mathbb{Z}_{26} = \{0, 1, \ldots 25\}$ of modulo 26

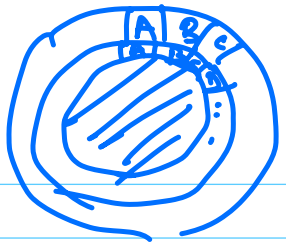
$x, y \in \mathbb{Z}_{26}$

$x + y = (x+y) \bmod 26$ remainder

key = k, msg = m, ciphertext = c

$Enc(k, m) = (m + k) \bmod 26$

$Dec(k, c) = (c - k) \bmod 26$



Encryption/decryption is letter by letter

$m, c \in \mathbb{Z}_{26}$

$K \in \mathbb{Z}_2$

Shift cipher

$Enc(k,m) = (m+k) \bmod 26$

$Dec(k,c) = (c-k) \bmod 26$

ciphertext = PIIPRZ

plaintext = ?

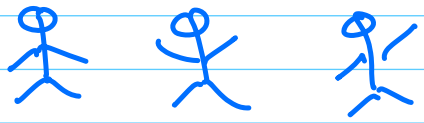
k	plausible plaintext
1	OHHOQY
2	...
3	...

complexity → brute force = $O(n)$

practically, you expect $\frac{n+1}{2}$ ←

Sherlock Holmes : A. C Doyle

"The adventure of the dancing men"



A B C .....

Secret key = mapping.

How many mappings?

$= 26!$

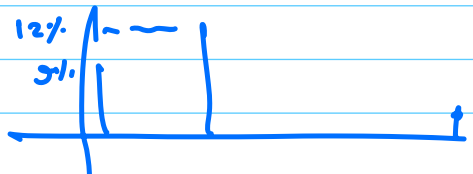
$m! \sim m^{m+1/2}$

$26^{26+1/2}$ to $26^{25+1/2}$

$\sim 2^{88}$

plaintext → ciphertext

Statistics are preserved



QU = double character

TH_

Define — secure encryption → create such methods

more security goals

---

9 Jan 25

TIGER → permute $no^s = 120$

Program — to attack **substitution cipher**

use (i) plaintext statistics

(ii) dictionary of words

(ii) can be eliminated, and the attack made completely automated.

3rd toy cipher : **Vigenére cipher**

→ extend shift cipher

Shift cipher

Plaintext → Ciphertext ← Shift

a → a+(x)

b → b+(x)

⋮

} easy attack: brute force

plaintext : attack at dawn

Secret word = **secret key** = CRY

Encryption:

attack at dawn..

+ CRY CRY CRY CRY...

ciphertext = c .. .. . . - - - - -

A=0
B=1
C=2
⋮
Z=25

Cryptanalysis :

ciphertext is given to you,

Kerckhoff's rule ← method is known

La Cryptography Militaire — but key is unknown

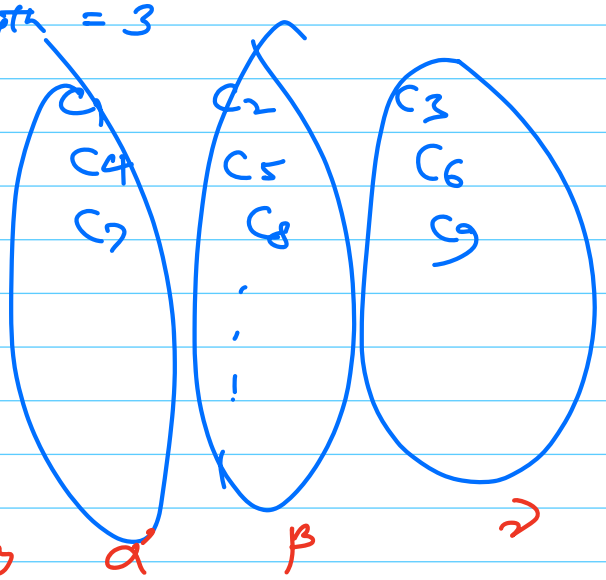
Security by obscurity (i.e hiding the method) ✗

How to attack:

Que: Suppose you knew the key length. Can you attack the system?

Ciphertext: $c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7\ c_8 \ldots\ldots$

(given) key length = 3



if prob of i^{th} letter in plaintext = p_i

($p_0, p_1, p_2 \ldots p_{25}$ = for English characters) → known

prob of ciphertext letters

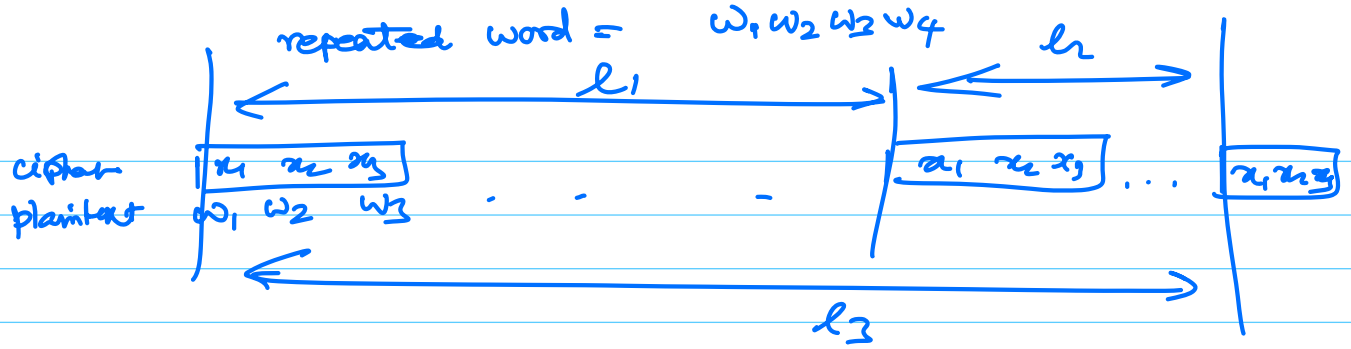
$q_0, q_1, q_2 \ldots q_{25}$

if shift = α then $p_i \simeq q_{i+\alpha}$

Part 2: How to find the length of the key word?

(i) bruteforce

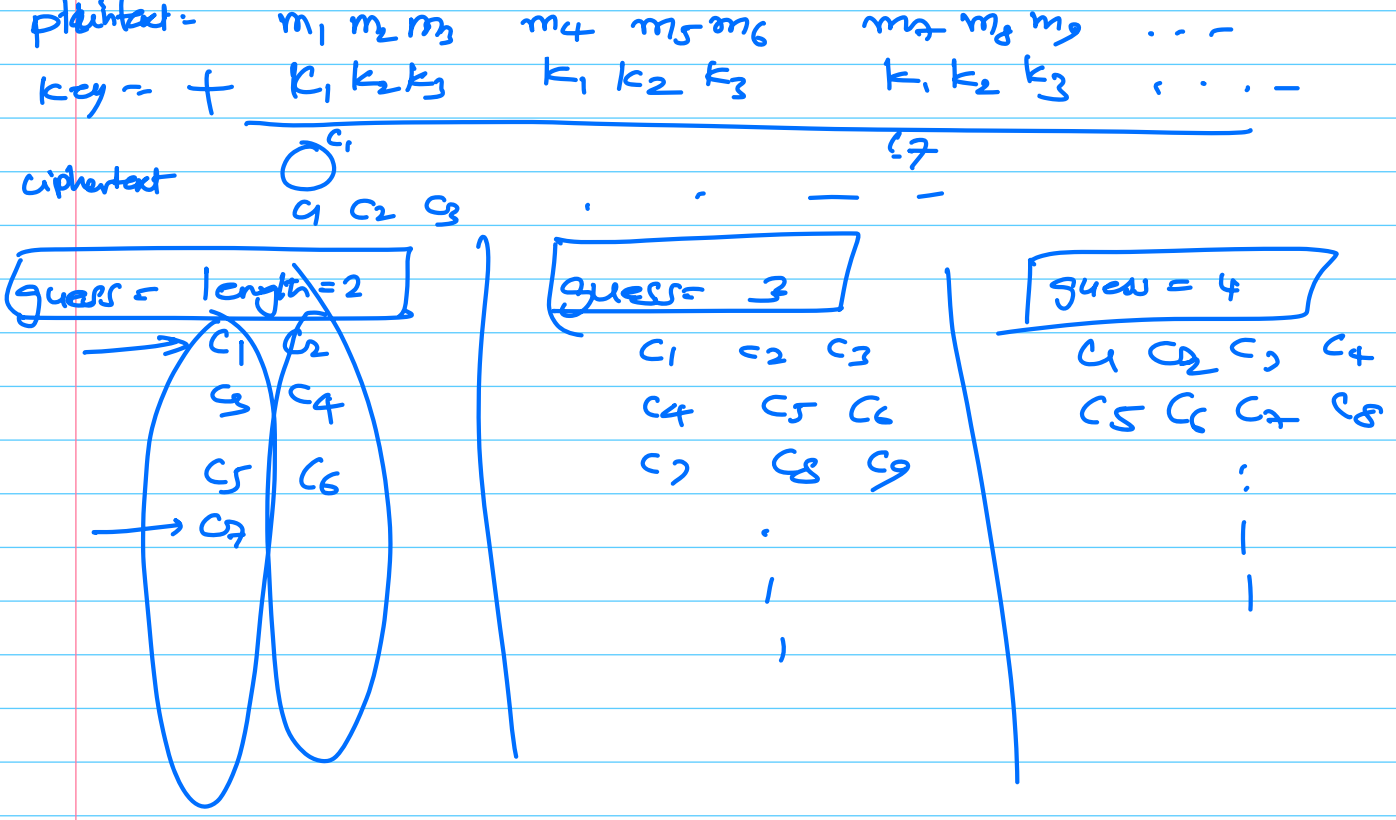
(ii) Kasiski's test



l_1, l_2, l_3 are all likely to be multiples of key length

$\Rightarrow$ key length $\leq$ gcd (l_1, l_2, l_3)

(iii) Some brute force + some statistical observation



if key length guess = correct
→ each column represent shifted English characters

if key length guess = wrong
→ random shifts

Index of coincidence

- measure of how likely is it that the column is simply shifted alphabets vs random alphabets

for correct guess

$$\sum p_i^2 = \sum (p_0)^2 + (p_1)^2 + (p_2)^2 + \dots$$

$$= 0.065$$

for any guess

$$\frac{1}{26}(p_0 + p_1 + \dots p_{25})$$

$$\simeq \frac{1}{26}$$

$$\simeq 0.038$$

Assignment 1:

ciphertext 1

↓

Some random text from internet

↓

Encrypted using substitution cipher

- plaintext = { a-z, A-Z, 0-9, ., $, ; ... }
- hidden code word

↓

to be submitted

ciphertext will be different

ciphertext 2

Encrypted by Vigenere cipher

(i) length of the key,

(ii) key word

↓

to be submitted

The Code Book : Simon Singh

What is secure encryption?

① Looking at the ciphertext, no one should be able to find the secret key

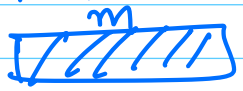
$$Enc(k, m) = m \quad ; \quad Dec(k, c) = c$$

Valid Encryption algorithm:

$$\forall k, \forall m \qquad Dec(k, Enc(k, m)) = m$$

② Looking at the ciphertext, no one should be able find the message

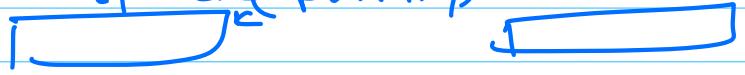
$Enc(k, m) = c$



there exists an algorithm $A(c) \rightarrow m$

③ Looking at the ciphertext, not even a single bit of the plaintext should be revealed

$c_1 = Enc(\text{appt. letter}_1)$ $\quad c_2 = Enc(\text{appt. letter}_2)$



c_1, c_2 given

goal: one should not be able to figure out if $m_1 > m_2$ or not.

non-trivial

No efficiently computable function of the message should be revealed by looking at the ciphertext.

plaintext: $\{01, 00\}$

$\text{Enc}(k, \cdot) = c$

Looking at the ciphertext, A can predict the following:

"the first bit of the plaintext is 0".

Textbook:

Introduction to Modern Cryptography.

Katz & Lindell

CRC Press.

---

10 Jan 25

Recap: What is a (secure) secret encryption?

$\mathcal{P}$ = set of plaintexts = $\{m_1, m_2, \ldots m_\ell\}$

$\mathcal{C}$ = set of ciphertexts = $\{c_1, c_2, \ldots c_{\ell'}\}$

(obviously: $\ell' \geq \ell$)

$\mathcal{K}$ = set of keys = $\{k_1, k_2, \ldots k_t\}$

Functions:

(i) **KeyGen** : every time the function is run, it produces a random secret key

$k \in \mathcal{K}$

initial part of this course

$$\mathcal{K} = \{0,1\}^n$$

Binary strings of length n bits

KeyGen in this case = produce a $k \in \mathcal{K}$ with prob. $= 1/2^n$

in practice. $n \geq 128$ bits, for higher level of security (paranoid / quantum comps) — 256 bits

IOT devices - 64 bits

(ii) **Enc** $(k \in \mathcal{K}, m \in \mathcal{P}) \rightarrow c \in \mathcal{C}$

possibility:

$Enc(k, m_1) \rightarrow c$

$Enc(k, m_1) \rightarrow c'$

$= |\mathcal{C}| = 2 \cdot |\mathcal{P}|$

Enc(·,·) can be deterministic or randomized

(iii) **Dec** $(k \in \mathcal{K}, c \in \mathcal{C}) \rightarrow m \in \mathcal{P}$

deterministic

Valid Encryption : $\forall m \in \mathcal{P}, \forall k \in \mathcal{K}$

$$Dec(k, Enc(k, m)) = m$$

Secure Encryption ?

Trivial attack 1

Given a challenge ciphertext c, aim is to find m

- guess the key k

then $Dec(k, c) \rightarrow$

Prob of success of the adv.

$= \frac{1}{|K|}$

$(Ex) = \frac{1}{2^n}$

attack cost = 1

$\rightarrow$ n should be sufficiently large

Trivial attack 2

Brute force the keys

effort = 2^n decryption calls

Success prob = 1

Given a ciphertext c, an attacker should not get some non-trivial info about the plaintext

↓

information

Claude Shannon :

information $\propto \frac{1}{\text{prob.}}$

Event $\rightarrow$ t outcomes

↓ prob.

$p_1, p_2, \ldots p_t \rightarrow$ information contained $= -\sum p \cdot \log p$

Entropy $= \sum p (\log 1/p)$

Even against infinite computational power of the adv, no advantage to her.

$$Pr(\text{plaintext} = m \mid \text{ciphertext} = c) = Pr(\text{plaintext} = m)$$

$$\forall\, m \in \mathcal{P}, \quad \forall\, c \in \mathcal{C}$$

Perfectly Secure Encryption

Adversarial setting:

Adv. Dove soap

Adv Game

Challenger — Adversary

$k \xleftarrow{\$} \mathcal{K}$

$m_0, m_1 \in \mathcal{P}$

$m_0 \neq m_1$

m_0, m_1 & $|m_0| = |m_1|$ ←

coin toss $b \xleftarrow{\$} \{0,1\}$

$c = Enc(k, m_b)$

c →

Some computations - at the end produce $b' \in \{0,1\}$ guess

Adv wins if $b' = b$

Pr of attacker wining $= 1/2$ ⇒ Perfectly Secure Encryption

Is it possible to achieve this strong security notion?

Vernam Cipher / OTP (one time pad)

$$\mathcal{P} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$$

Key Gen : $\longrightarrow$ uniformly randomly produce a key

$$k \in \{0,1\}^n$$

$$Pr(k = k^*) = \frac{1}{2^n}$$

$$Enc(k, m) = k \oplus m$$

$$Dec(k, c) = k \oplus c$$

XOR

		xor
0	0	0
0	1	1
1	0	1
1	1	0

note: $\alpha \oplus \alpha = 000\cdots0$ for all α

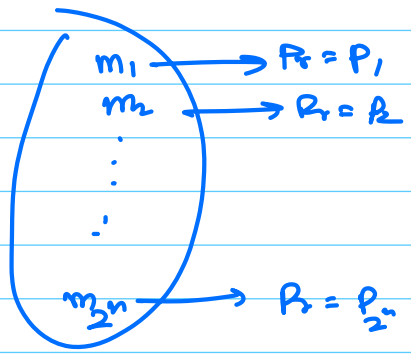
To prove : for this encryption algo.

$\forall c, \forall m$

$$Pr(\mathcal{P} = m \mid \mathcal{C} = c) = Pr(\mathcal{P} = m)$$

Notice:

$\mathcal{P}$



$\longrightarrow$ determines the RHS above

LHS: $Pr(P = m \mid C = c)$

$$= \frac{Pr(P = m \cap C = c)}{Pr(C = c)}$$

$$= \frac{Pr(P = m \cap m \oplus k = c)}{\sum_{m} Pr(P = m) \times Pr(C = c \mid P = m)}$$

$$= \frac{Pr(P = m \cap \dots \cap k = c \oplus m)}{\sum (\dots) \times Pr(k = (m \oplus c))}$$

$Pr(k = (m \oplus c)) \rightarrow 1/2^n$

$$= \frac{Pr(P = m) \times 1/2^n}{1 \times 1/2^n}$$

$$= Pr(P = m)$$

Used in Hotline between US & USSR during cold-war.

USSR army / spies — they used OTP

One-Time Pad ?

Two time pad → if the same key is used twice

$C_1 = m_1 \oplus k$

$C_2 = m_2 \oplus k$

$\Rightarrow C_1 \oplus C_2 = m_1 \oplus m_2$

Project Verona — wikipedia

14 Jan 2025

Information theoretic security

Shannon Security

$\forall m, c$: $Pr(\mathcal{P} = m \mid C = c) = Pr(\mathcal{P} = m)$

Equivalent

Perfect Indistinguishability

- m_1, m_2 picked by Adv.
- one of them randomly encrypted by the challenger
- Adv's advantage in distinguishing which of the two messages produced c = 0

$\forall m_1, m_2, \forall c$

$Pr(\mathcal{P} = m_1 \mid C = c) = Pr(\mathcal{P} = m_2 \mid C = c)$

① Shannon security $\Rightarrow$ Perfect indistinguishability

$Pr(\mathcal{P} = m_1 \mid C = c) = Pr(\mathcal{P} = m_1)$ for any m_1, c

(defn of conditional prob)

$$\frac{Pr(\mathcal{P} = m_1 \cap Enc(k, m_1) = c)}{Pr(C = c)} = Pr(\mathcal{P} = m_1)$$

$$\frac{Pr(\mathcal{P} = m_1) \cdot Pr(Enc(k, m_1) = c)}{Pr(C = c)} = Pr(\mathcal{P} = m_1)$$

$$\underset{k,c}{Pr}(Enc(k, m_1) = c) = \underset{c}{Pr}(C = c)$$

$$= Pr(Enc(k, m_2) = c)$$

$\Rightarrow \quad P_r\left(Enc(k, m_1) = c\right) = P_r\left(Enc(k, m_2) = c\right)$

(π) Perfect indist $\Rightarrow$ Shannon security

This is the strongest notion of encryption security — even against a exponential time attacker, this security is enough

Practical instantiation. OTP

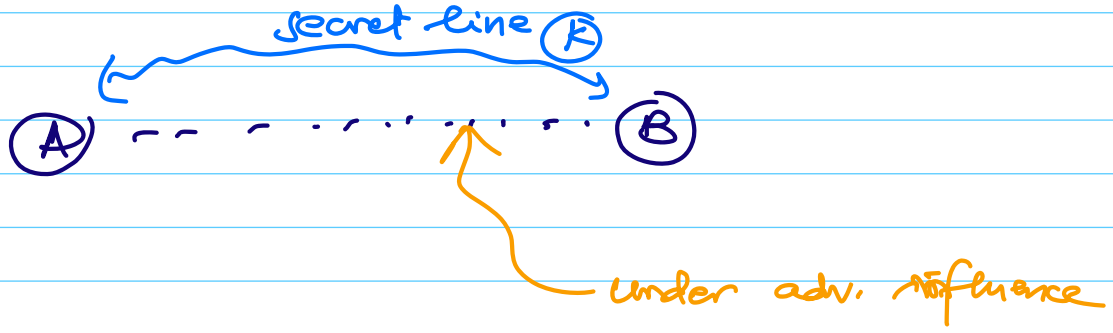
$Enc(k, m) = k \oplus m$

$Dec(k, c) = k \oplus c$

$k, m, c \in \{0,1\}^n$, k was chosen uniformly at random

Practical problem –

to encrypt a message of length n bits, we need a secret key of length n bits

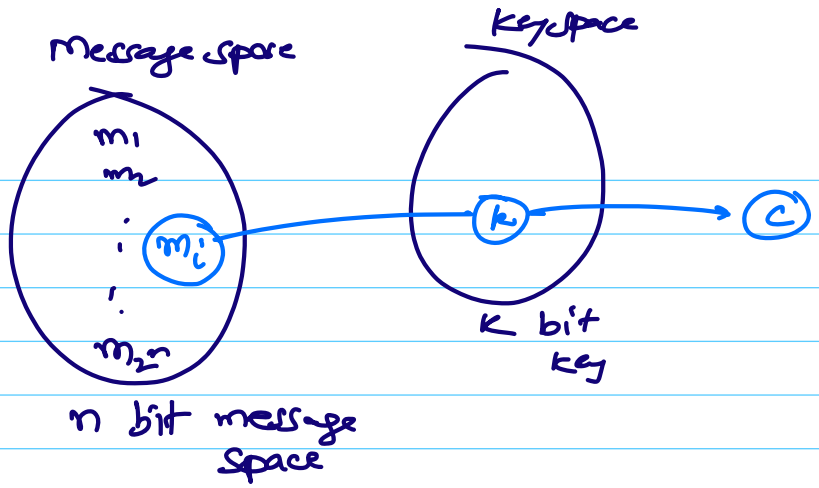


Que: Can we reduce randomness requirement?

i.e message length = n bits

but key length = k bits where $k < n$

Claim: In this setting, perfect security is impossible



Attacker's idea:

- Pick the ciphertext c
- Decrypt by all possible keys

$$\mathcal{P}' = \{ Dec(k_1, c), Dec(k_2, c), Dec(k_3, c), \ldots \}$$

$$|\mathcal{P}'| \leq |\mathcal{K}|$$

But $$|\mathcal{K}| < |\mathcal{P}|$$

$\Rightarrow \exists\ m^* \in \mathcal{P}$ which is not in $\mathcal{P}'$

$$Pr(\mathcal{P} = m^* \mid C = c) = 0$$

But for some message $m' \in \mathcal{P}'$

$$Pr(\mathcal{P} = m' \mid C = c) \neq 0$$

$\Rightarrow$ Practical limitation in using OTP in real-life

Computational Security

(this is weaker than perfect security)

(I) Now we will limit the power of the adversary.

- Adv can not run for exponential time
- allowed only poly. time operations
- Security parameter = n

(II) Attacker will have some non-zero prob. of winning the distinguishability game

$$Pr(\ldots) - Pr(\ldots) \neq 0$$

Now

$$Pr(\ldots) - Pr(\ldots) = \epsilon$$

ϵ ↓ negligible in n

$\leq \frac{1}{2^n}$

Negligible function:

Pr of winning the game should be "negligible"

$$\text{poly}(n) \times (\text{Prob of win}) = \text{negl}(n)$$

$\Rightarrow$

$$\text{negl}(n) = \frac{\text{poly}(n)}{\exp(n)}$$ or something that sort

$n \to \infty$, $\text{negl}(n) \to 0$

How to create a computationally secure encryption scheme ?

Why did OTP kill any statistics in the input ?

$Pr(b=0) = 0.9$ $\quad$ $Pr(b=1) = 0.1$

$Pr(k=0) = Pr(k=1) = 0.5$ $\quad$ uniformly random key bit

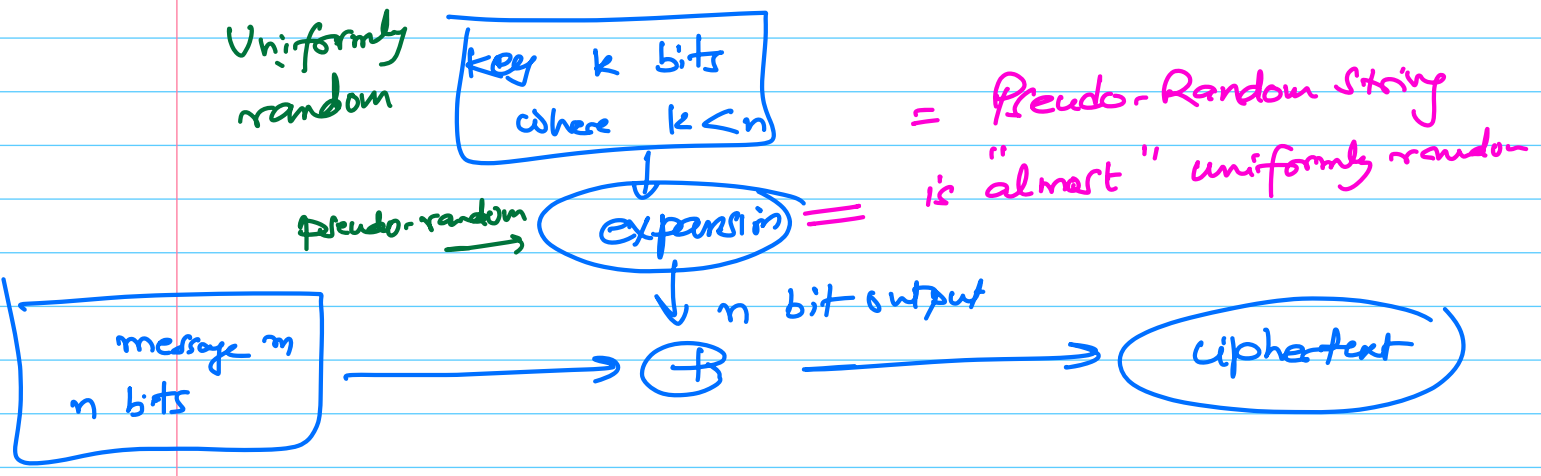
Que

$$Pr(b \oplus k = 0) = ?$$

$$= Pr(b=k=0) + Pr(b=k=1)$$

$$= 1/2 \times 0.9 + 1/2 \times 0.1$$

$$= 1/2$$



Example: rand( ) function in C, Java, ...

What is a "random" string ?

Source is random (Not the strings)

1. is 000.... 0 random ?
2. is 110011... random ?
3. is 011001.... random ?

Yao - given a string  if next bit

(Next bit test)

Statistical test of randomness

NIST Test suite

↙

(National Institute of Std. & Tech.)

↓

Dept of Commerce US govt.

- Java randomness → entropy pool
- Lavarand - wall of entropy

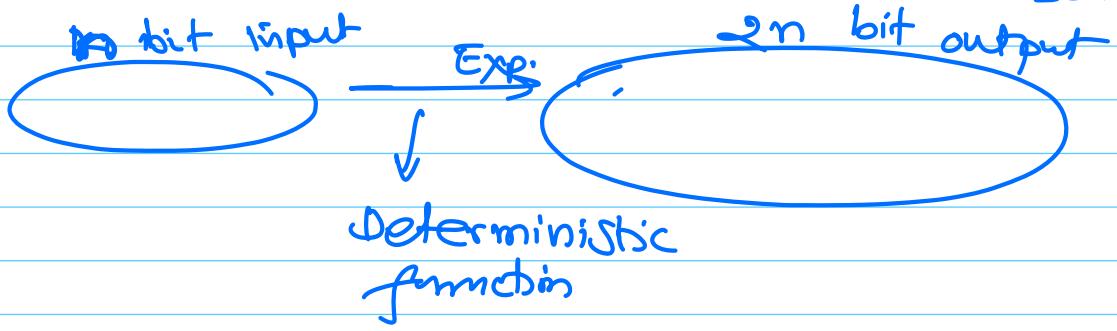
PRG (Pseudo-Random Generator)

Idea: input: small amount of randomness

output: large " of pseudo-randomness

↙

Something which can't be identified by "someone" as coming from random vs. pseudo-random source



How many possible outputs for this function?

$= 2^n$

Size of output set $= 2^{2n}$

if we pick any random string in the output set, it is not likely to have been produced by this algo

$$\Pr\left[\begin{array}{c}\text{a random string}\\ \text{from output is}\\ \text{actually produced}\\ \text{from this algo.}\end{array}\right] = \frac{1}{2^n}$$

Brute force attack will win with prob ≈ 1

— to prevent this, we will not allow attacker this exponential power

---

16-Jan-25

PRG - Pseudo-random generator (G)

Idea: a PRG takes a short seed and produces a long output which "looks like" random

& PRG is a deterministic function.

$G : \{0,1\}^s \rightarrow \{0,1\}^l$ where expansion $l > s$

should be "looking like" random

Adv A : PPT : Probabilistic Polynomial Time algorithm

↳ (i) internal coin tosses
(ii) also get a random string as input

Real world	Ideal world
$s \xleftarrow{\$} \{0,1\}^n$	$y \xleftarrow{\$} \{0,1\}^l$
$y = G(s)$	

A : PPT algorithm which is going to distinguish between the two worlds
: Distinguisher for PRG G

$A(x) \to 1$ if x is random
$\to 0$ if x is generated by G.

if

$$Pr\left(A(G(s)) \to 1 \text{ where } s \xleftarrow{\$} \{0,1\}^n\right) - Pr\left(A(y) \to 1 : \text{where } y \xleftarrow{\$} \{0,1\}^k\right) \leq negl(n)$$

then G is a PRG.

Example 1.

$G(s) = s||s$ n bit $\to$ 2n bit

(1) Que: is G a PRG ?

Adv: $A(x) = A(x_1||x_2)$ where $|x_1| = |x_2|$

check if $x_1 = x_2$
if yes → output 0
else → output 1.

Dist prob = $1 - 1/2^n$ = not negligible

(2) G is a given PRG : $\{0,1\}^n \to \{0,1\}^{2n}$

$x \xleftarrow{\$} \{0,1\}^n$

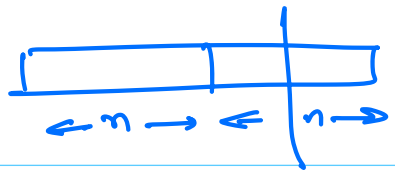
$G(x) = y_1||y_2$ where $y_1, y_2 \in \{0,1\}^n$

$G'(x) = G(y_1)||G(y_2)$ ←

$G' : \{0,1\}^n \to \{0,1\}^{4n}$

Que: is G' a PRG ?

③ G is a PRG : $\{0,1\}^n \to \{0,1\}^{2n}$

$G(x) =$ 

$G'(x) =$ first $\frac{3n}{2}$ bits of $G(x)$

Why did we not permit exponential time adversary?

$G(s) \longrightarrow \{0,1\}^{\ell} = y$ ($s \in \{0,1\}^n$)

Adv get y

Create a table of output sequences

$T = \{ G(00\cdots0),\ G(0\cdots01),\ G(0\cdots10), \cdots$

$\cdots \}$ Size 2^n where each entry is of length ℓ bits

When Adv gets a string x

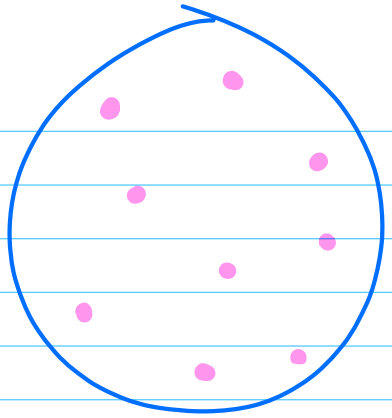
Adv checks if $x \in T$ or not

if $x \in T$ ↓ Adv says PRG

if $x \notin T$ ↓ Adv say random ✓

$\Pr(\text{A distinguishing}) = 1 - \frac{2^n}{2^\ell}$

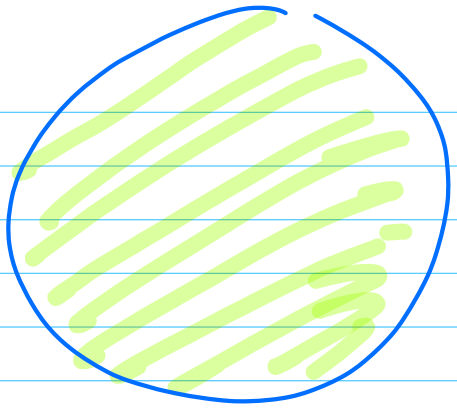
PRG:



ℓ bit outputs

coverage = 2^n

Random

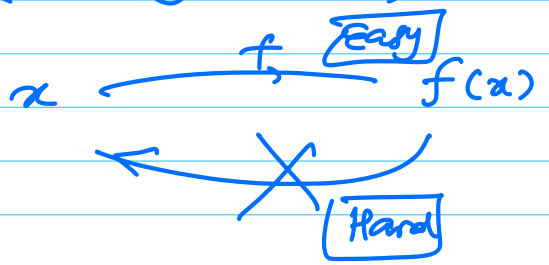


coverage = 2^ℓ

Do we have PRG's ?

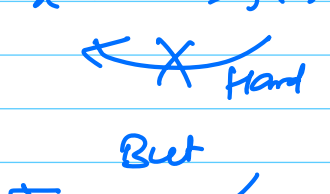
- We don't know
- We believe they exist

OWF: (One-way function)



OWF $\Leftrightarrow$ PRG

Trapdoor OWF



y: trapdoor

Using PRG to create Encryption Algorithm

$m \leftarrow$ plaintext $\in \{0,1\}^\ell$

Secret key $k \leftarrow \{0,1\}^n$

$\ell > n$

Enc: $Enc(k,m) = G(k) \oplus m = c$

Dec: $Dec(k,c) = c \oplus G(k) = m$

Stream cipher

— description of the PRG

Microsoft Word → encrypt a file with a password

key ← f(password)

ciphered doc = PRG(key) + Word Doc

Earlier - RC4 was used by microsoft

RC4:

State array of size 256 byte

Init (1) Initialization: S[i] = i

KSA (2) Use key to randomize the state

key is of size 8 byte to 16 byte

eg. K = $k_0, k_1, k_2, k_3, k_4, k_5, k_7$

PRGA (3) PRG key is not used anymore

State → one byte at a time and jumble up the state again

RC4 was one of the most used stream ciphers in history

— Used in Microsoft Word, ... Later notes,

— IEEE 802.11 Wireless protocol

WEP

(Wired Equivalent Privacy)

other stream ciphers -

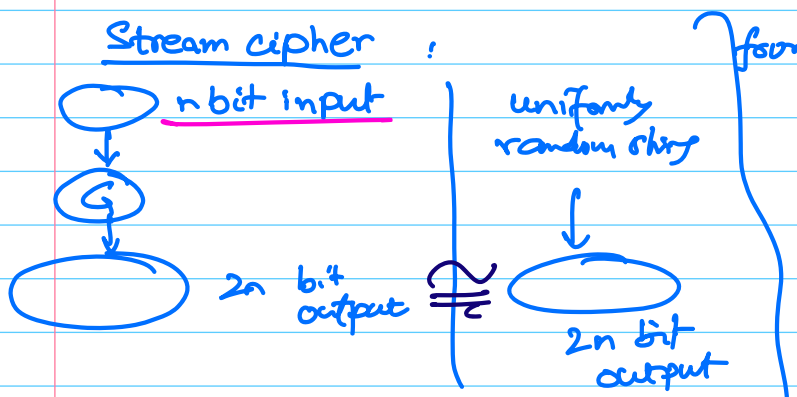
A5/1 , A5/3 ....

Sosemanak, Trivium, ...

Kargil War - Musharraf ↔ Deputy army chief

Jan 21, 2025

Stream cipher :

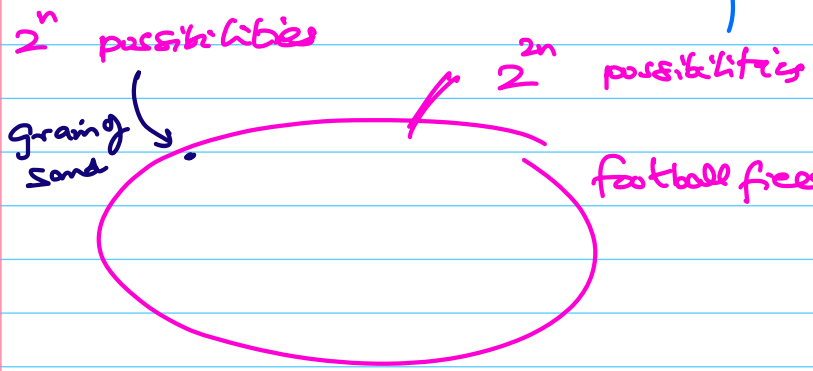


from a PRG G

(i) length extending

(ii) deterministic

(iii) output of G on a randomly chosen input "looks" uniformly random



$\approx_c$

computationally indistinguishable

Stream cipher from a PRG

Secret key = $k \xleftarrow{\$} \{0,1\}^n$

$m \in \{0,1\}^\ell$ where $\ell > n$

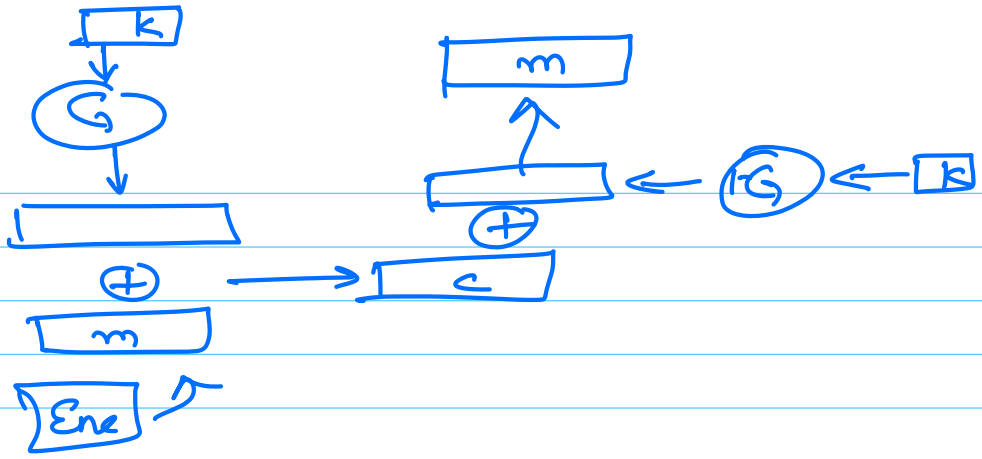
Enc: $c = m \oplus G(k)$

where $G: \{0,1\}^n \to \{0,1\}^\ell$

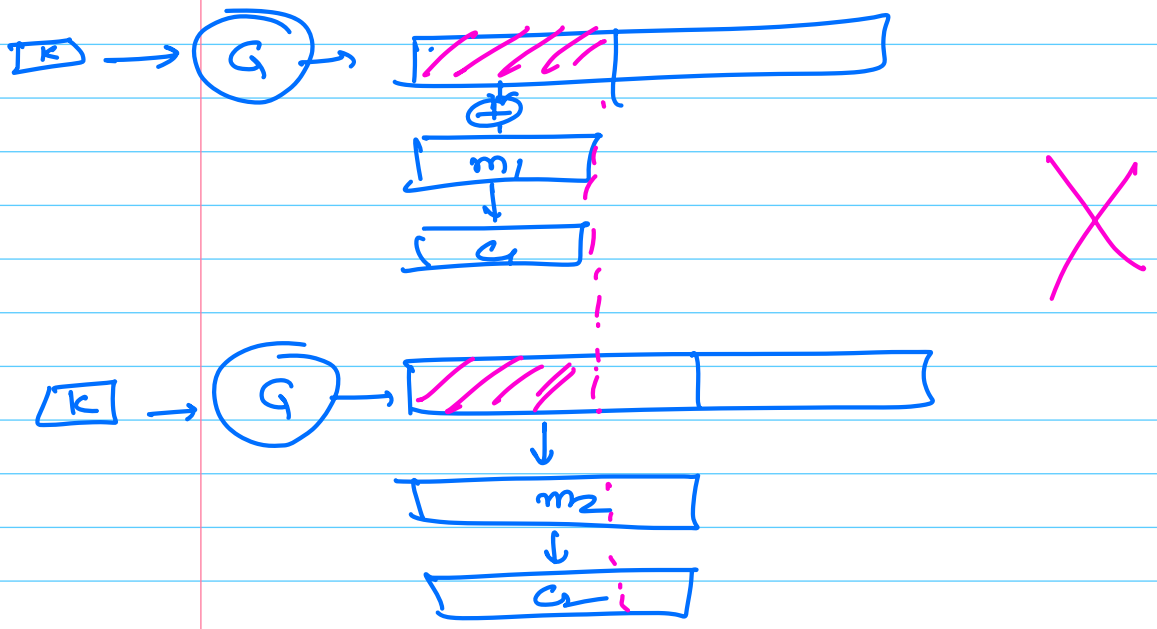
Indirect of c_1 & c_2

$c_1 = m_1 \oplus G(k)$, $c_2 = m_2 \oplus G(k)$

Dec: $m = c \oplus G(k)$

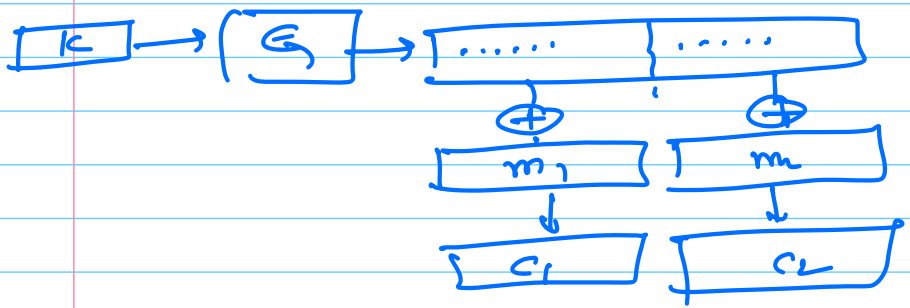


Suppose we had 2 (or more) messages to encrypt.



Instead:

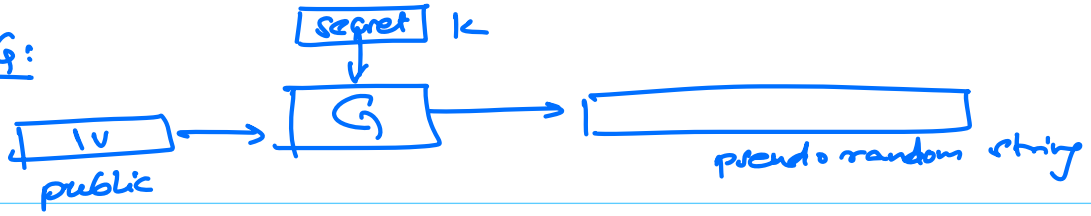
Synchronous Stream Cipher



Initialization Vector (IV)

- random string
- it can be public (it need not be hidden)

PRG:



Stream cipher:

Encrypt message m_1 with key k

(i) Generate a random string IV_1

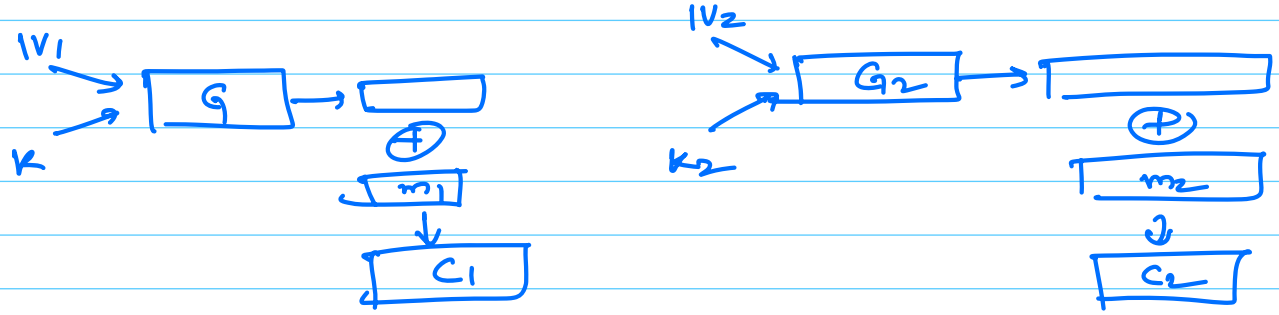
(ii) $G(IV_1, k) \oplus m_1 = c_1$

(iii) ciphertext $= (IV_1, c_1)$

Decryption:

ciphertext $= (\alpha, \beta)$

Decryption $= G(\alpha, k) \oplus \beta$



IV based PRG:

Security requirement

$G(IV_1, k)$ & $G(IV_2, k)$ are comp. indist.

$\simeq_c$

random string

G (known value, unknown key) $\simeq_c$ rando

WEP – design IEEE 802.11 standard for wireless communication

Wired Equivalent Privacy

Based on RC4

old design by Ronald Rivest

3 byte IV = 24 bits

2^{24} possibilities

16,000,000

IV repeat problem is less likely

Key in RC4 was upto 16 bytes
= 128 bits

Export control law in US

Only 40 bit keys were permitted
= 5 bytes

Designers of WEP:

key will be between 5 bytes to 13 bytes
+ 3 bytes of IV
= 8 bytes to 16 bytes

8 bytes → Export

Internal use

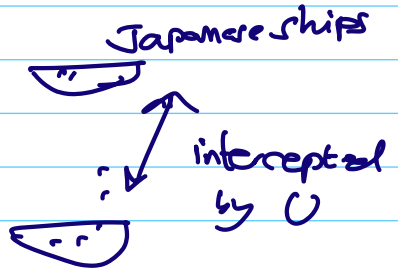
RC4 was very weak –

Deprecated – not expected to be used anywhere

So far: only the ciphertext is available to the attacker

WW II:

Island in Atlantic



AF is to be attacked

Plan: desalination is used for drinking water

spread news: desalination plant at midway island is broken down

Attacker can not only observe ciphertexts but can also choose plaintexts

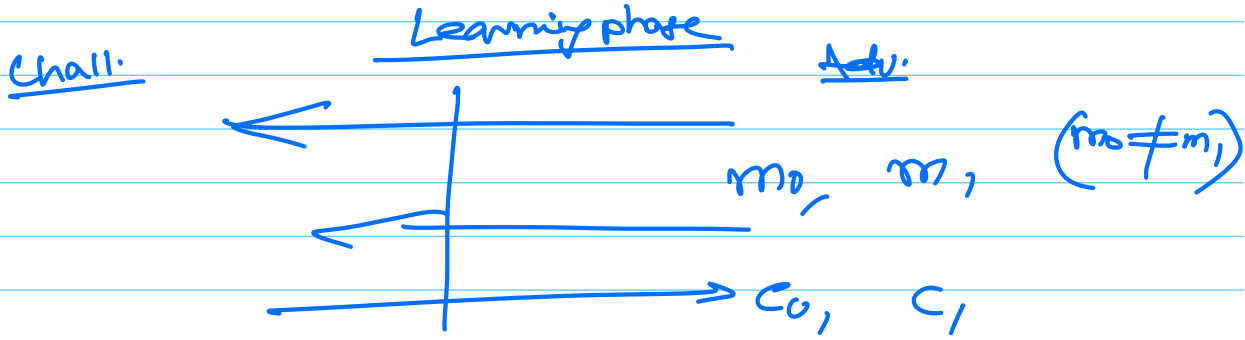
CPA (Chosen Plaintext Attack)

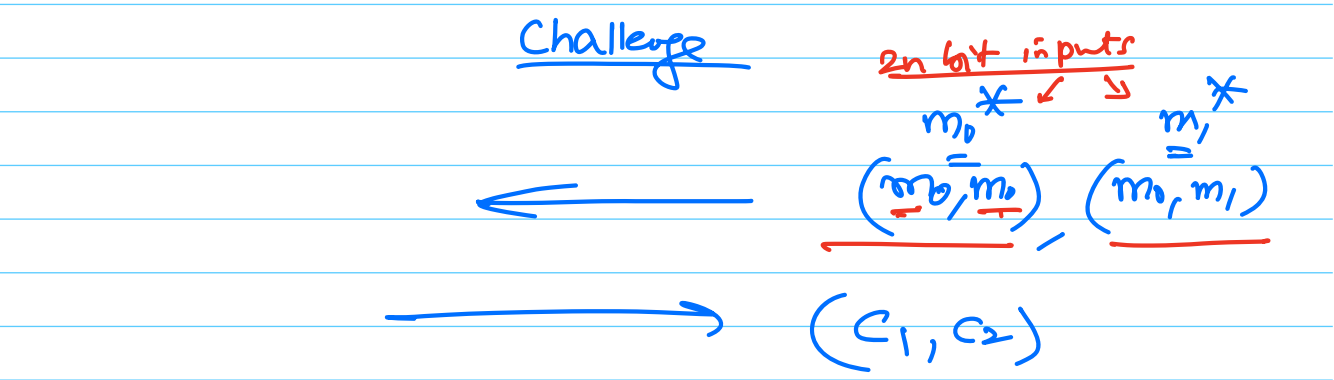
Adv wins if $(b' = b)$

CPA secure encryption

Observation: no deterministic encryption algo. can be CPA secure.

Why? we show an CPA attack against any deterministic encryption algo.





Check if $(c_1 == c_2)$

yes → m_0^*

no → m_1^*

Prob of winning = 1.

23 Jan 2025

CPA security – Chosen plaintext attack

Various attack models:

More power ↓

① Eavesdropping only – only power is to observe ciphertexts

② Known-plaintext attack –

attacker knows $(m_1, c_1), (m_2, c_2), \ldots (m_t, c_t)$

t pairs of messages & ciphertexts are known to the attacker

Goal remains same as previously studied

i.e. Indistinguishability

Adv (m_0^*, m_1^*) → challenger

challenger ← $b \xleftarrow{\$} \{0,1\}$, $c = Enc(k, m_b^*)$

prediction $b' \in \{0,1\}$ →

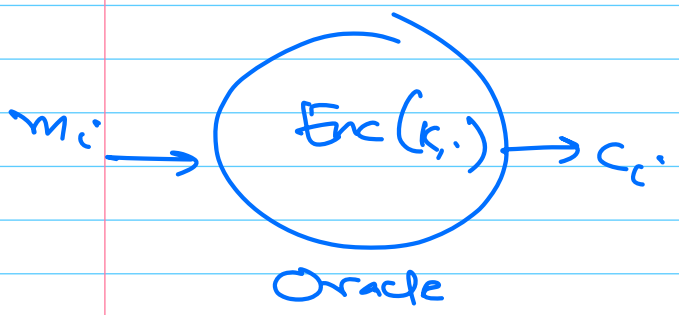
③ Chosen plaintext attack – (CPA)

attacker chooses messages m_i for $i = 1$ to t

receives c_i for these

Same goal as previous

≡ Encryption Oracle is available to the attacker



– $Enc(k, \cdot)$ is to be made available to the attacker

– But NOT the key k.

⑥ Chosen ciphertext attack (CCA)

CCA attacker = CPA attacker + ...

↙

Chosen ciphertexts can be decrypted

≡ Decryption oracle is available to the attacker

Restriction: can't ask for decryption of the challenge ciphertext

CCA (Non-adaptive CCA) CCA-1

Adv. challenger

I + II

$m_0, m_1, m_2 \ldots$ → Enc oracle

← $c_0, c_1, c_2 \ldots$

$c_0', c_1', c_2', \ldots$ → Dec oracle

← $m_0', m_1', m_2' \ldots$

challenge phase

III

m_0^*, m_1^* → $b \xleftarrow{\$} \{0,1\}$

c ← $c = Enc(k, m_b^*)$

$b' \in \{0,1\}$ → win or loss

(Adaptive CCA) CCA-2

I + II + III + I & II

Ultimate goal for encryption = CCA-2 Secure Encryption schemes

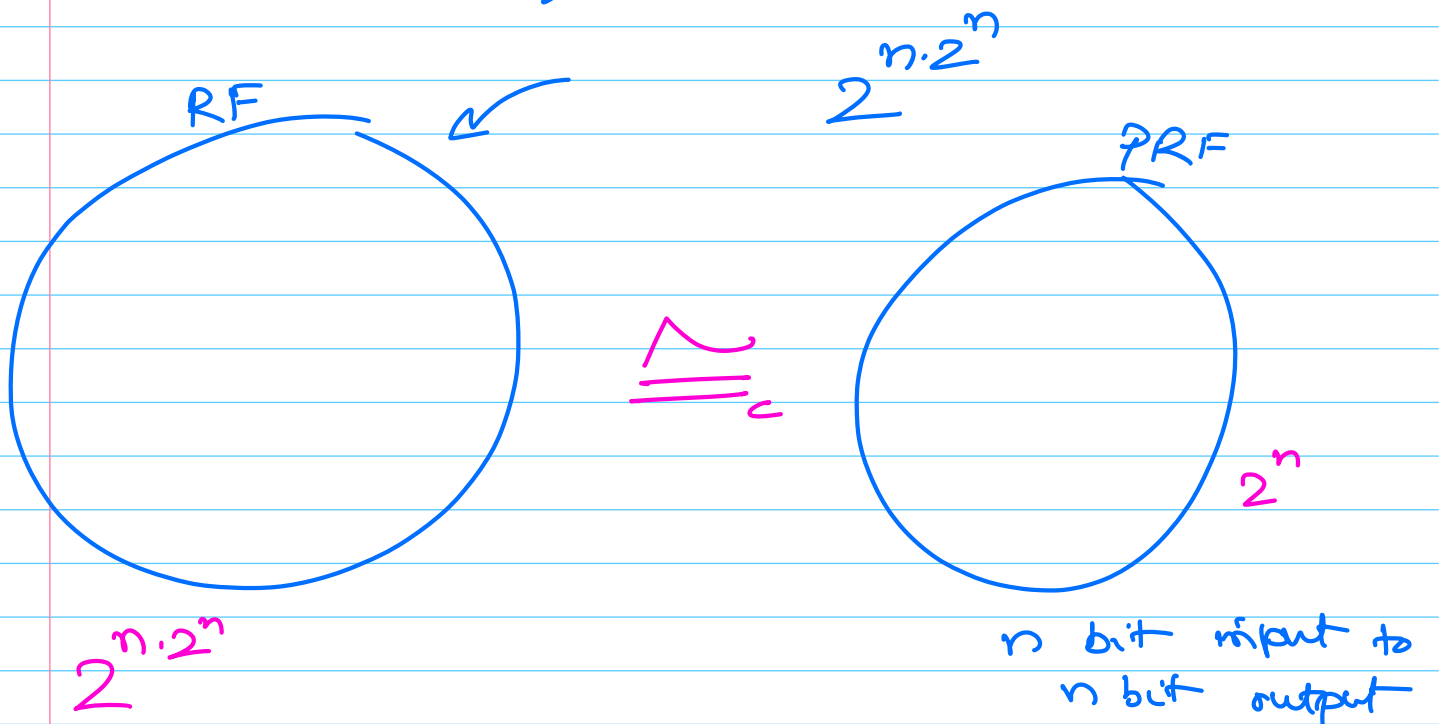
CPA Secure Encryption :

- From previous class
  - Deterministic Encryption can't be CPA secure

PRF : Pseudo-Random function

Consider functions from $\{0,1\}^n \to \{0,1\}^n$
n-bit to n-bit function.

Que. How many such functions exist ?



+ additional parameter
= key K
($\in$ n bits)

Keyed function:

$F(K, \text{input}) \to \text{output}$

If key is fixed

$F_K(\cdot) \to$ deterministic

RF Computation in simulation -

$T[\,] = \{\ \}$ empty table

- user asks query i
- Challenger checks if $T[i]$ = empty or not
  - if empty then answer (n-bit random string) also add $T[i] = \ldots$
  - If not empty then answer $T[i]$

consistency in repeated queries

PRF Computation

- key k is chosen at random & fixed
- user asks query i

Challenger answers $F(k, i)$

Encryption algorithm using PRF

Enc: $k \leftarrow$ randomly generate

m to be encrypted $\in \{0,1\}^n$

with PRF $F(-,-)$

- randomly generate $r \leftarrow \{0,1\}^n$
- Compute $F(k, r)$

ciphertext = $(r,\ F(k,r) \oplus m)$

e.g. m being asked for encryption twice

first time ciphertext $= (c_1, c_2) = (r_1, f(k, r_1) \oplus m)$

second time $= (c_1', c_2') = (r_2, f(k, r_2) \oplus m)$

This scheme is CPA secure because of PRF property

This scheme is NOT CCA secure

- ask for encryption of m_1

$(r_1, f(k, r_1) \oplus m_1)$

$m_0^*, m_1^* \longrightarrow$ one of them is encrypted

$\text{lsb}(m_0^*) \neq \text{lsb}(m_1^*)$

output we get is (c_1, c_2)

ask for decryption of

$(c_1, c_2 \oplus 1)$

No class tomorrow – Recording will be made

28 Jan 2025

PRF : $\{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$

fixing a key, fixes the function

Table | inp | out | gets determined completely

indexed functions
↳ indexing is by the key

Enc. a message of size n-bits

$f(k,\cdot)$ $\quad f_k(\cdot)$ = PRF obtained by fixing the key to be k.

$$Enc_k(m) = (r, f_k(r) \oplus m)$$

PRP - Pseudo-Random Permutation

$\{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$

index → $\{0,1\}^k$

restriction from PRF case: the new object is a permutation.

inp	out
all 2^n values	"

PRF vs PRP

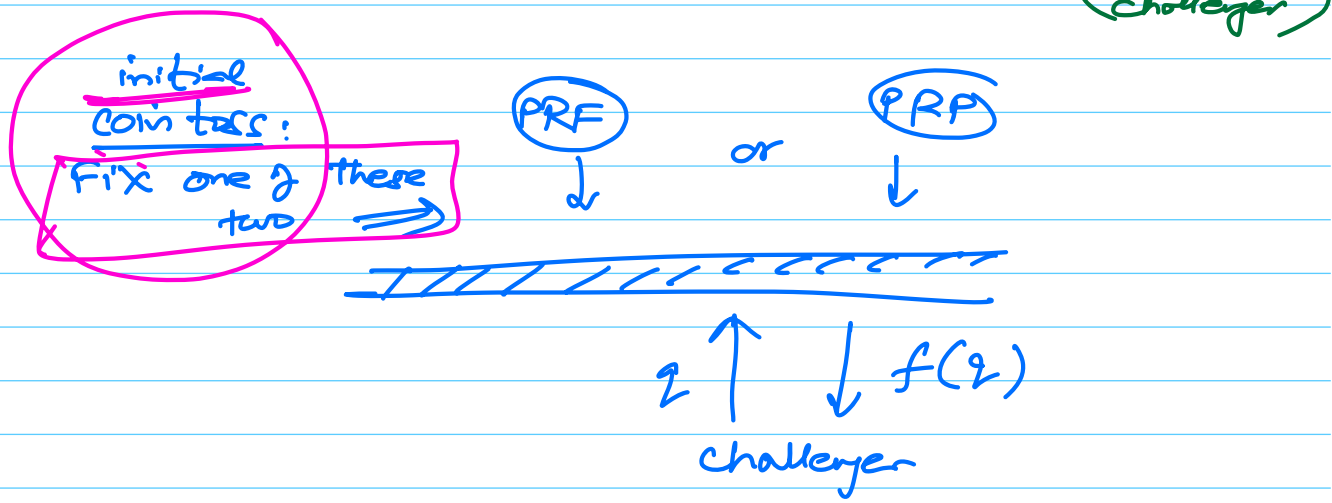
(i) toss a coin — choose either a PRF or a PRP
(ii) challenger is given oracle access
(iii) goal is to find out which one?

Suppose challenger has q queries, what is the pr. that the challenger distinguishes?

what happens if a query is repeated?

- Because answers are consistent, challenger doesn't benefit

- hence all queries should be distinct (to benefit challenger)



$$\text{Pr of (distinguishing)} = \text{Pr}\left(\text{distinct two queries outputs colliding}\right)$$

PRP–PRF switching lemma:

$$\text{Pr}\left(\text{distinguish a PRP } \& \text{ a PRF with } q \text{ queries}\right) \leq \frac{q^2}{2 \cdot 2^n}$$

Out of q queries $\rightarrow$ ${}^{q}C_2$ pairs

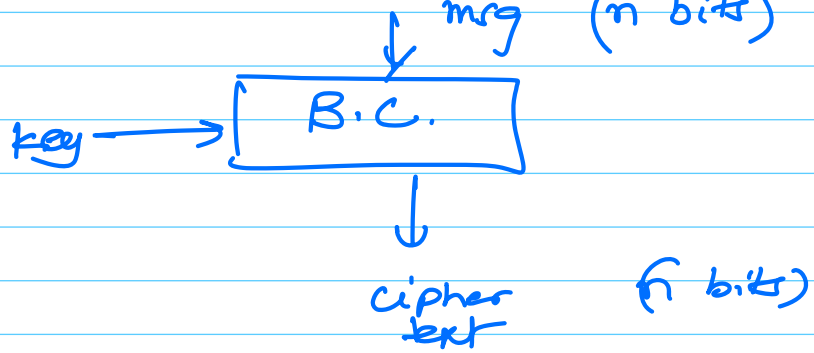
$$= \frac{q(q-1)}{2} \sim \frac{q^2}{2}$$

if $q = \text{poly}(n)$ then $\boxed{\cdots \leq \text{negl}(n)}$

PRP's are realized in practice by a construction known as Block Cipher

alternately,

Block cipher = PRP in practice



This is a deterministic construction

⇒ Should not be used for encryption in a direct manner.

Triangle inequality

Objects O_1, O_2, O_3

Prob. (to distinguish between O_1 & O_3) $\leq$ Pr (distinguish between O_1 & O_2) + Pr (dist. between O_2 & O_3)

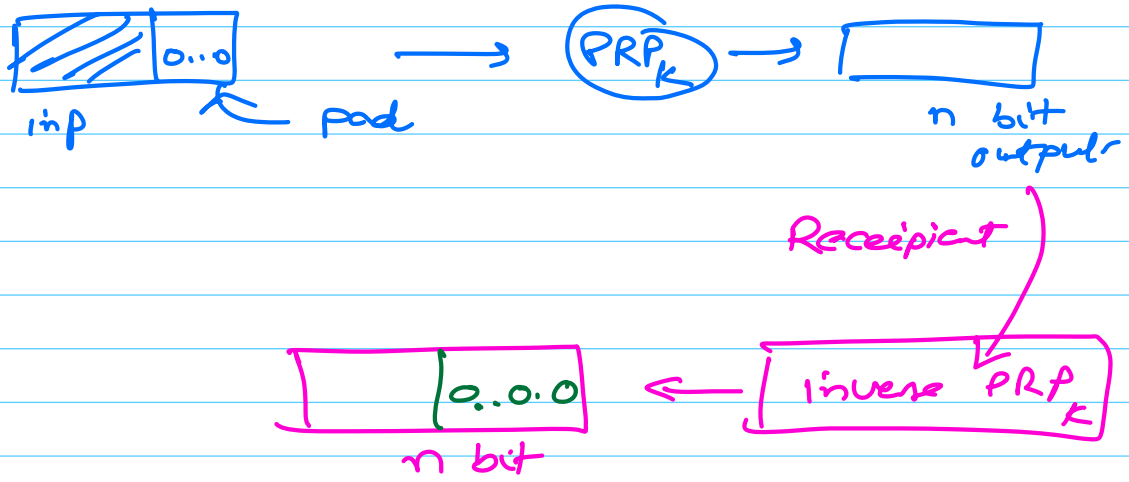
RF $\longrightarrow$ PRF $\longrightarrow$ PRP

$\epsilon(n)$ $\qquad$ $q^2/2^{n+1}$

How to encrypt messages which are not of n-bits ?

(how to evaluate PRP on inputs of length $\neq n$.)

(1) if message is small → do some padding

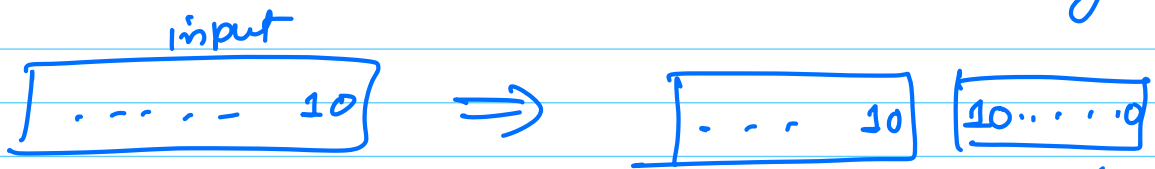


0^* padding does not work

Usual : 10^*

Any invertible padding is fine

(2) if padding is used — then it is required to be used always



(3) when message length is $> n$

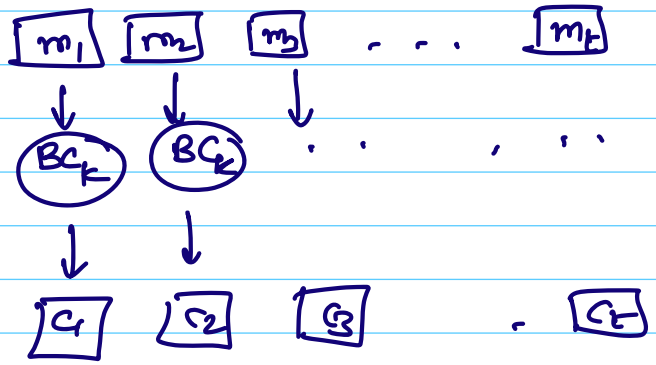
use padding + enough to make it a multiple of n bits

Blocks of size n-bits will be processed by a block cipher

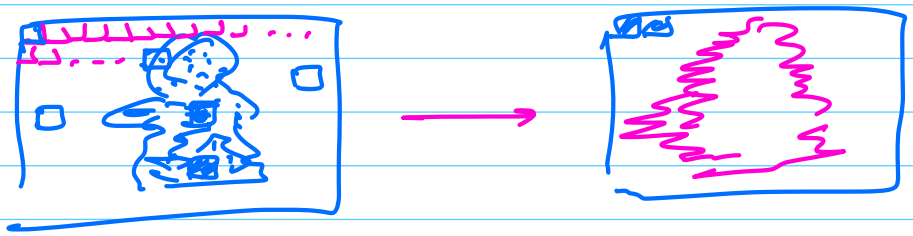
Mode of Operation

method for domain extension of a block cipher

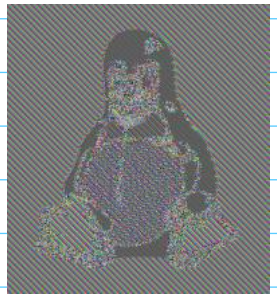
① ECB (Electronic Code Book)



Reordering ciphertext → reordering of plaintext



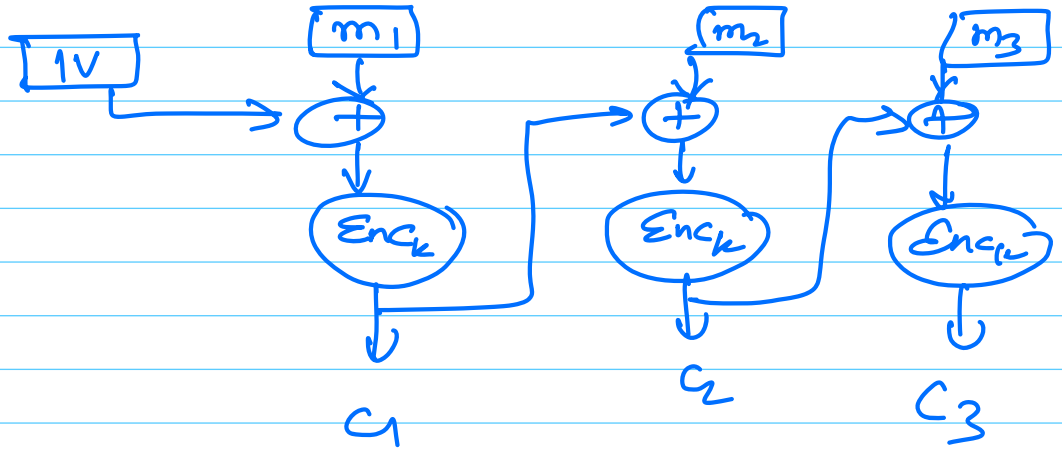
Penguin



ECB penguin

Masterkey → session key$_1$, session key$_2$, ... - - used in practice

② Cipher Block Chaining (CBC)



$inp = (m_1, m_2, m_3)$

$ciphertext = (IV, c_1, c_2, c_3)$

---

30 Jan 2025

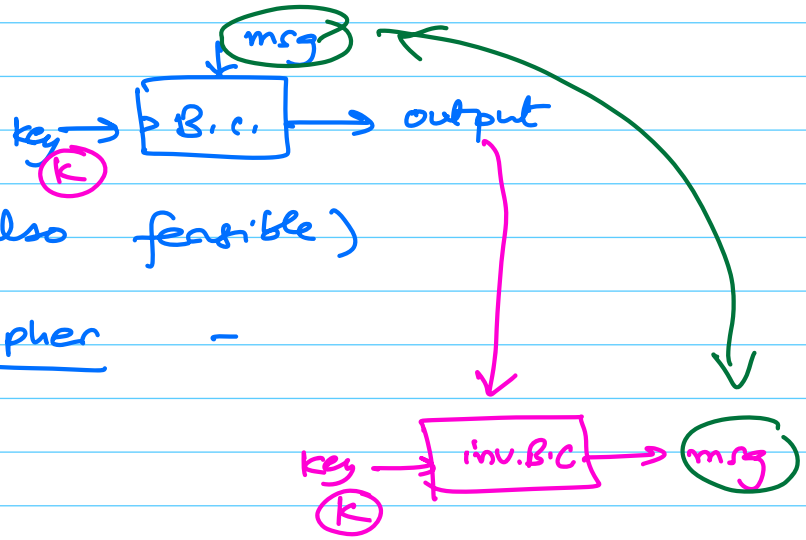
PRF → PRP

↳ in practice = Block Cipher

Block Cipher:

- an practical realization of a PRP
- n bit → n bit permutation
- deterministic construction

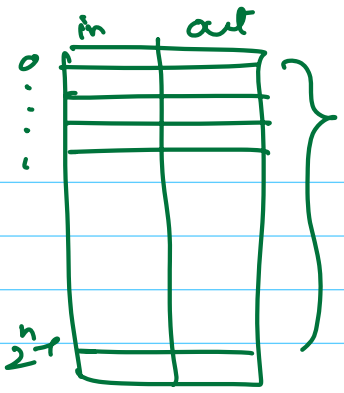
Syntax:



(inversion is also feasible)

Inverse Block cipher –

PRP:



PRP ($\#$ = 2^k) $\simeq_c$ RP $\#$ = $(2^n)!$

Security game:

Challenger

$b \xleftarrow{\$} \{0,1\}$

if ($b==0$) pick a RP

else pick a random key K & setup $PRP(K, \cdot)$

Adversary

$\leq poly(n)$ queries

interaction

Predict b'

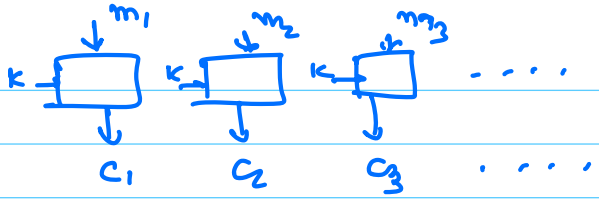
if ($b' == b$) Adv wins

else she loses

Construction is a PRP if $Pr(\text{Adv wins}) = \frac{1}{2} + \epsilon(n)$

For practical / real-life encryption scheme, we need to use a Mode of operation

① ECB : (Electronic Code Book)



inp = $(m_1, m_2, m_3, \dots)$

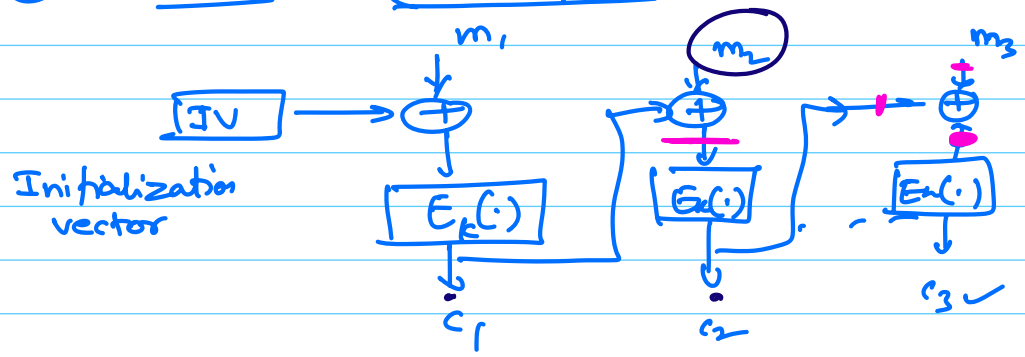
ciphertext = $(c_1, c_2, c_3, \dots)$

Weaknesses: is deterministic construction
⇒ insecure

Easiest attack :
- given (c_1, c_2, c_1) → send (c_2, c_1, c_3)
- image encryption (Penguin)

useful only when encrypting 1 block
(or when used rarely with guarantee that all blocks are different)

② CBC : (very popular)



inp = $(m_1, m_2, \dots)$

ciphertext = $(IV, c_1, c_2, \dots)$

ciphertext length = 1 block more than plaintext

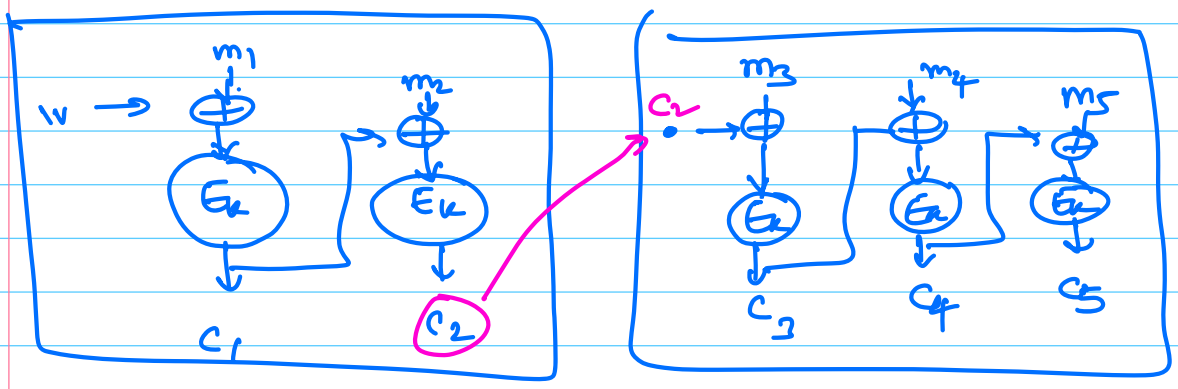
if $c_i \oplus m_{i+1} = c_j \oplus m_{j+1}$

then $c_{i+1} = c_{j+1}$

Adv (an adversary can break security property of CBC mode) $\leq$ Adv (breaking the property of block cipher) + Pr (inp. collision)

③ Chained CBC

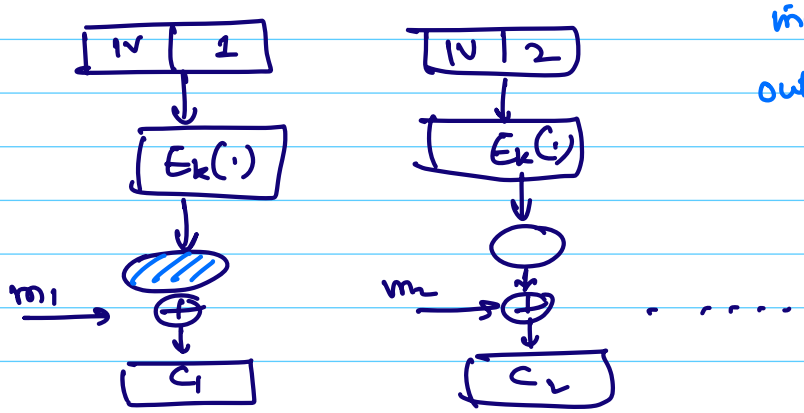
one problem of CBC is that you need to generate an IV for every new message



used in SSL 3.0 & TLS 1.0

④ CTR mode (Counter mode)

- Extremely popular



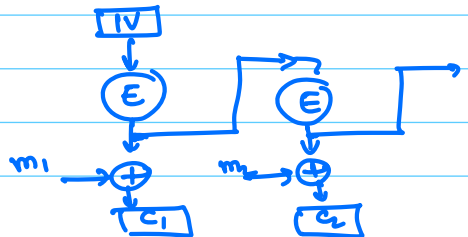
inp = $(m_1, m_2, \ldots)$

out = $(IV, C_1, C_2, \ldots)$

Converts a block cipher into a stream cipher

- no decryption circuit needed
- any block can be decrypted without waiting for other blocks
- Error propagation

⑤ OFB (Output feedback mode)



How to design a block cipher ?

Computerization → banks

Lloyds bank (London)

NIST (NBS → National Bureau of Standard)
(→ National Inst for Std. & Tech.)

under the dept of commerce, US govt.

Requirements → asked for designs

revised → asked again

IBM – designed Lucifer

Don Coppersmith, Horst Feistel, ...

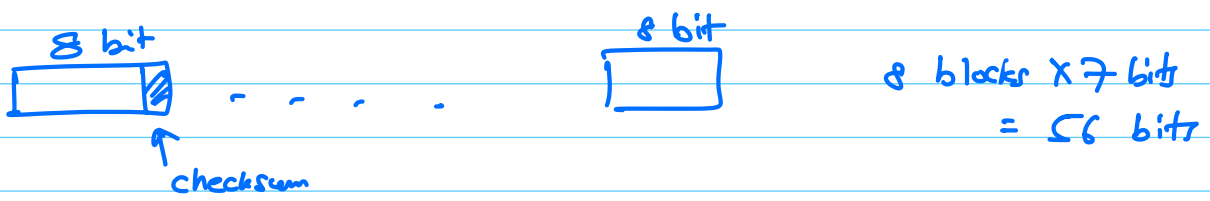
Revised by NSA –

→ Became Data Encryption Standard

DES (70s)

64 bit block cipher

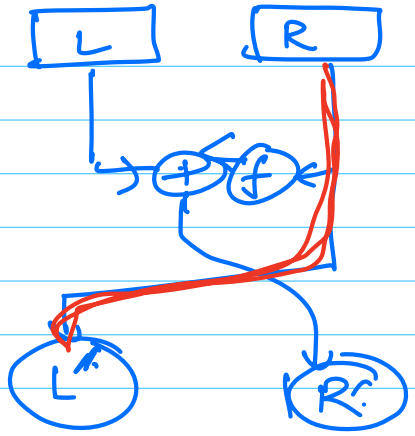
56 bit secret key



(i) The biggest criticism of DES — small key size

(ii) design criteria are not public
– fear of hidden trapdoors

feistel round



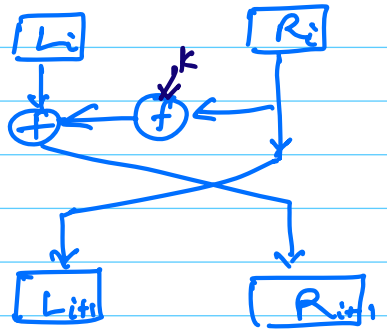
$L_{i+1} = R_i$

$R_{i+1} = L_i \oplus f(k, R_i)$

Jan 31, 2025

(Horst feistel)

1-round of feistel structure



$L_{i+1} = R_i$

$R_{i+1} = L_i \oplus f(k, R_i)$

How to invert ?

given (L_{i+1}, R_{i+1})

$R_i = L_{i+1}$

$L_i = R_{i+1} \oplus f(k, L_{i+1})$

⇒ Claim

if $f(k, .)$ is a PRF (n → n bit) then 1-Round of feistel structure is a PRP (2n → 2n bits)

Correct :

if $f(k, .)$ is a PRF then 1-round of feistel is a permutation.

Que: Prove that claim is wrong.

Proof: Supply input (L, R) → get ans (x, y)

check if $(x == R)$.

Symmetric key crypto community considers two types of PRPs -

① PRP (or weak PRP)

② SPRP (or strong PRP) → PRP ; in theory

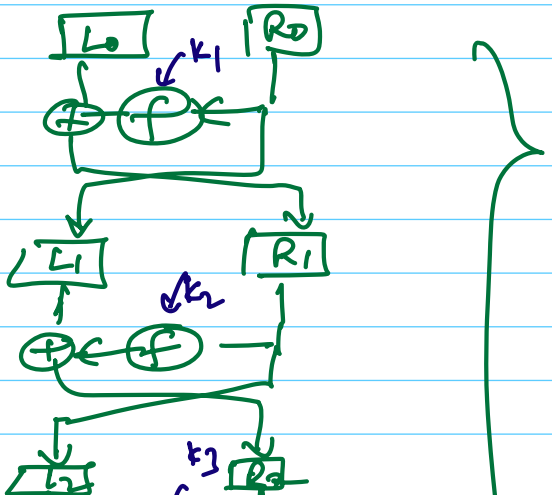
① adv gets access to only the forward computation (or the permutation only)

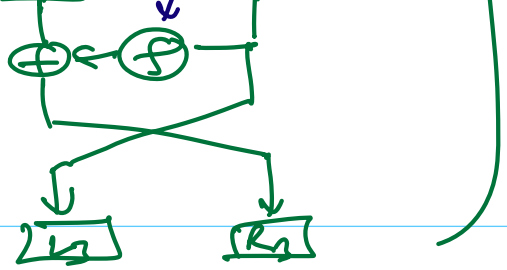
② adv gets access to both permutation as well as inverse permutation

Results:

① if each round function of feistel structure is a independent PRF then 3-rounds of feistel is a PRP

② if each round function of feistel structure is a independent PRF then 4-rounds of feistel is a SPRP.



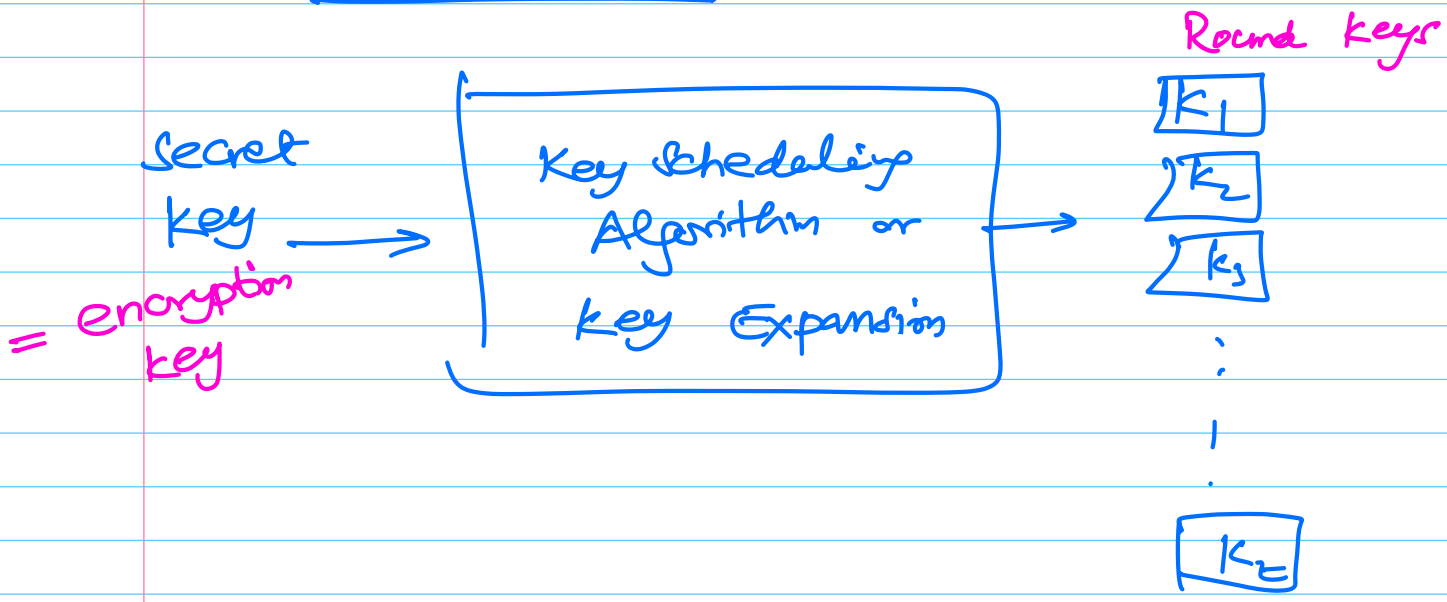


+ 1 more round
= SPRP

Most block ciphers are "iterated block ciphers"

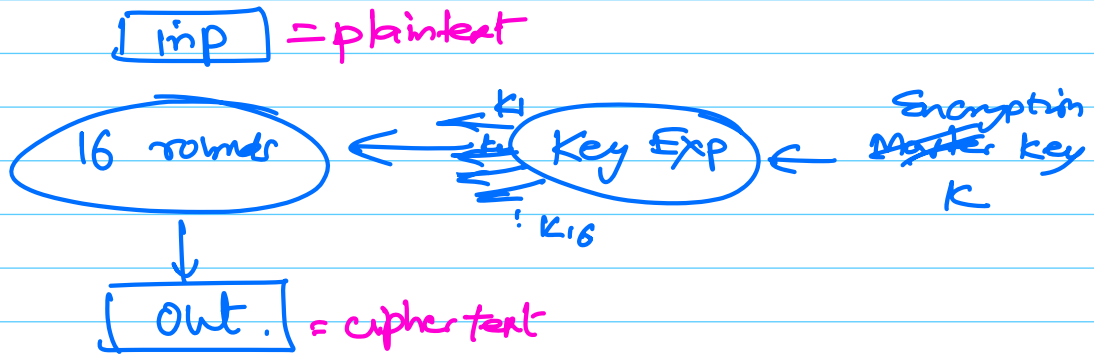
- iterate a specific structure many times

How to generate keys for PRF calls in each round?



DES is an old cipher (now deprecated)

Feistel design with 16 rounds



inp = 64 bit = out

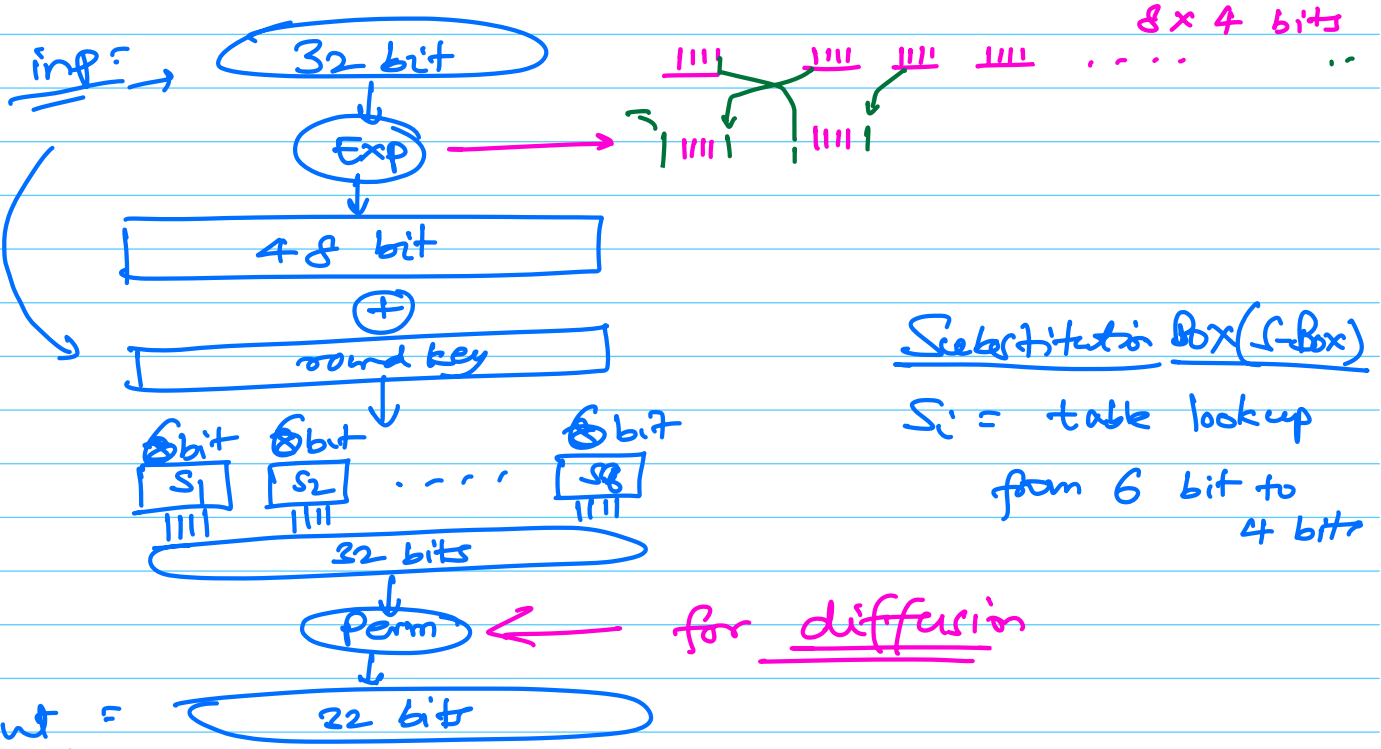
key = 56 bits

Key Expansion

56 bit → 11 round keys of 48 bits each

f function :

32 bits + 48 bits → 32 bits



f-function of DES

Substitution Box (S-Box)

S_i = table lookup from 6 bit to 4 bits

for diffusion

S-box specification : 64 input → 4 bits

bits of input

$x_1 x_2 x_3 x_4 x_5 x_6$

$x_1 x_6$ \ $x_2 x_3 x_4 x_5$	0 . . . . . f
00	
01	
10	
11	

4x16 table each entry containing a hex value

each row is a permutation of nos $\{0, 1, 2, \ldots f\}$

Differential Cryptanalysis (90's)

fix some input diff. $:= \delta \xrightarrow{\text{out}} \delta'$

pick inps $(x, x \oplus \delta)$

get outs $\downarrow y \quad \downarrow y'$

$(x, x \oplus \delta) \rightarrow (y, y')$

where $y \oplus y' = \delta'$

$\Rightarrow$ good pair

$\longrightarrow$ Secret key extraction

Biham & Shamir - applied this technique on all known block cipher at that time

LOKI, ... Broke

DES strong

~ 90's : Coppersmith et. al → "Design criteria of DES"

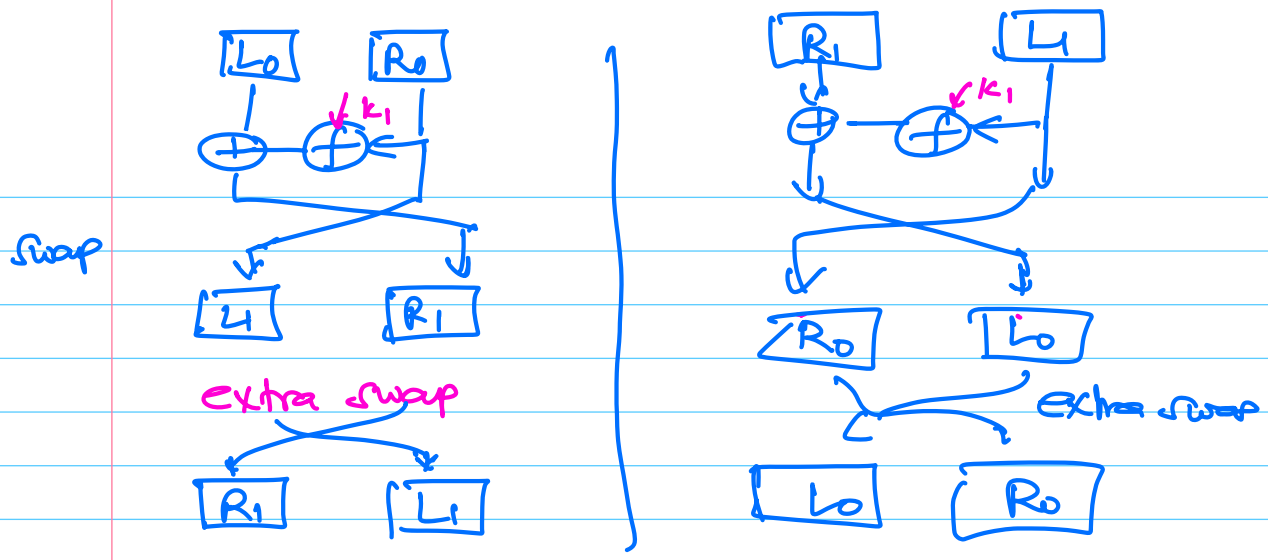
T-attack = diff. cryptanalysis

92' Cryptanalysis of DES by Biham & Shamir

full 16 rounds were broken

Requirement: 2^{47} chosen plaintexts are needed

56 bit key is recovered in $< 2^{56}$ time



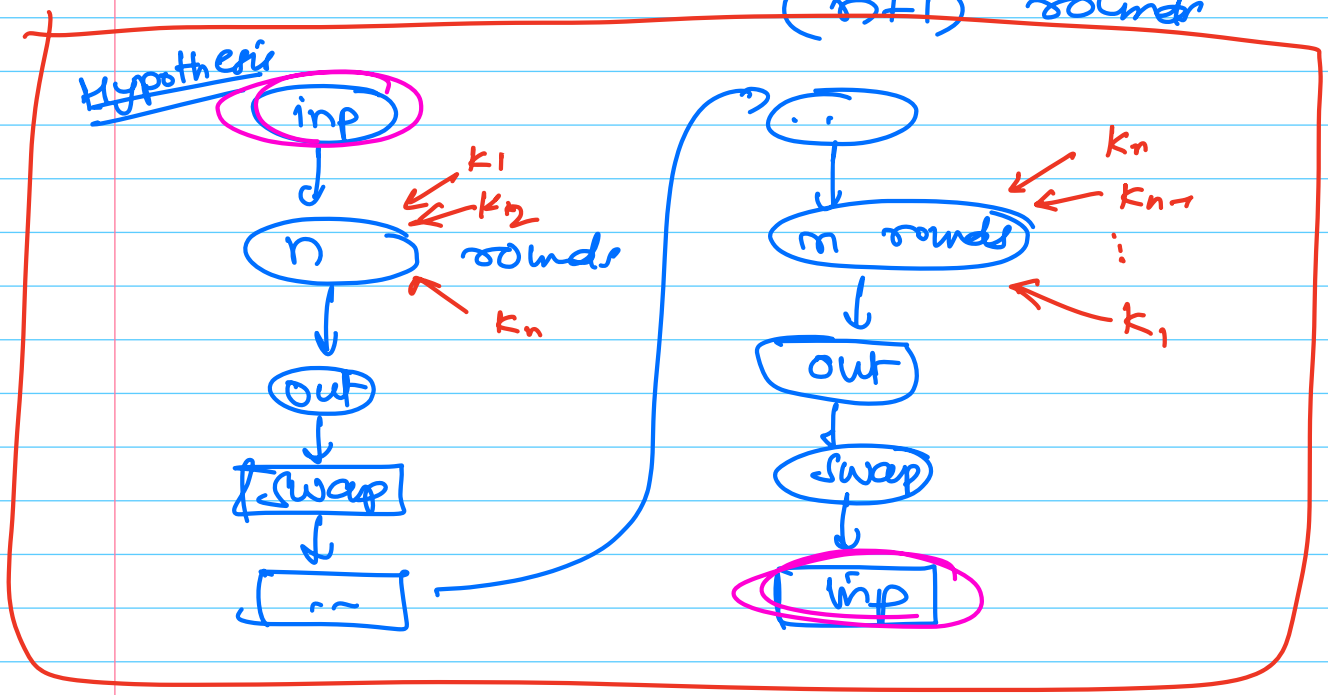
Any no. of rounds can be decrypted this way.

Proof By using Mathematical Induction:

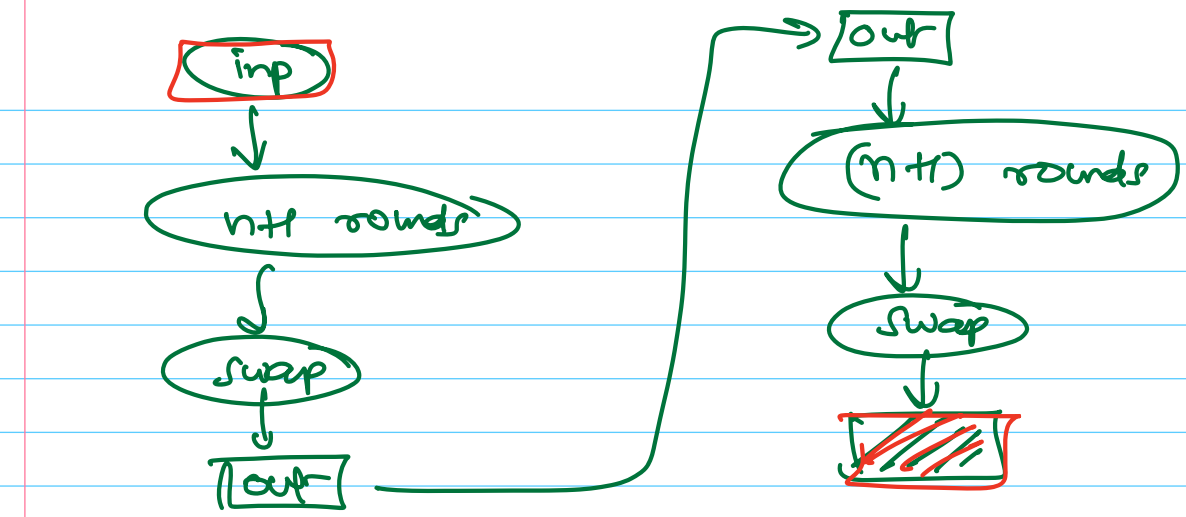
Basis: 1 - round — we showed above

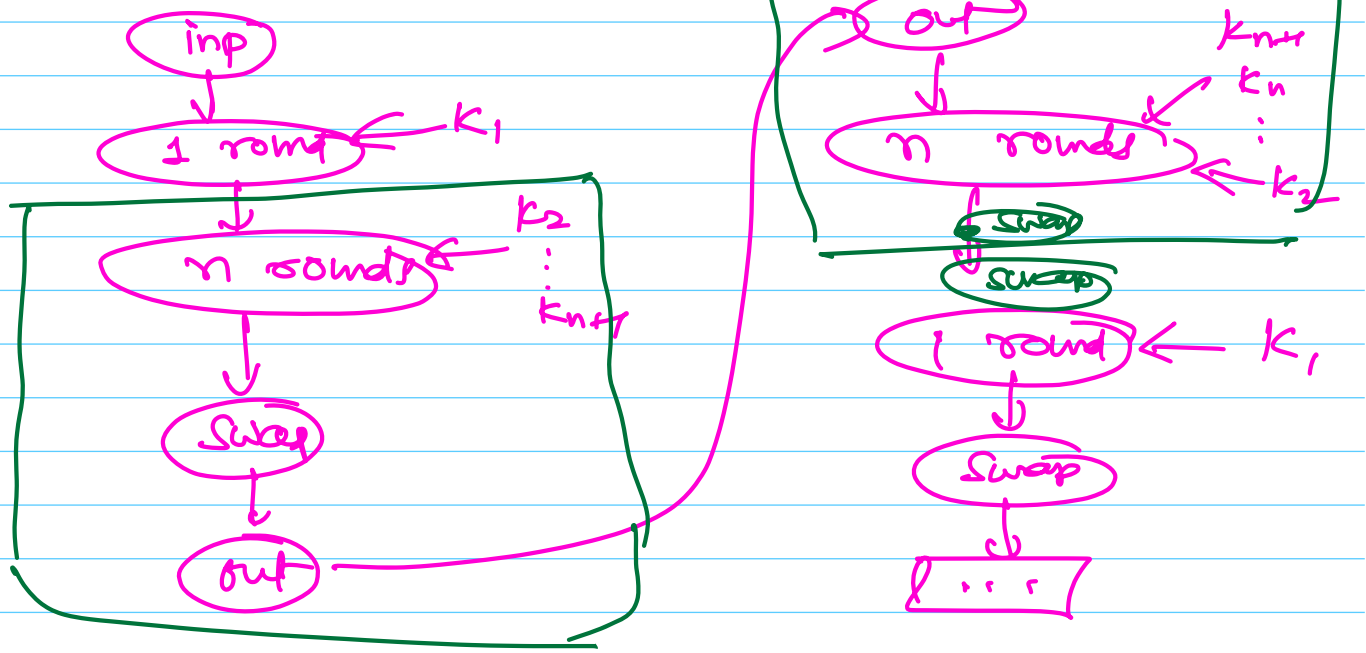
Hypothesis: Assume this works for n-rounds

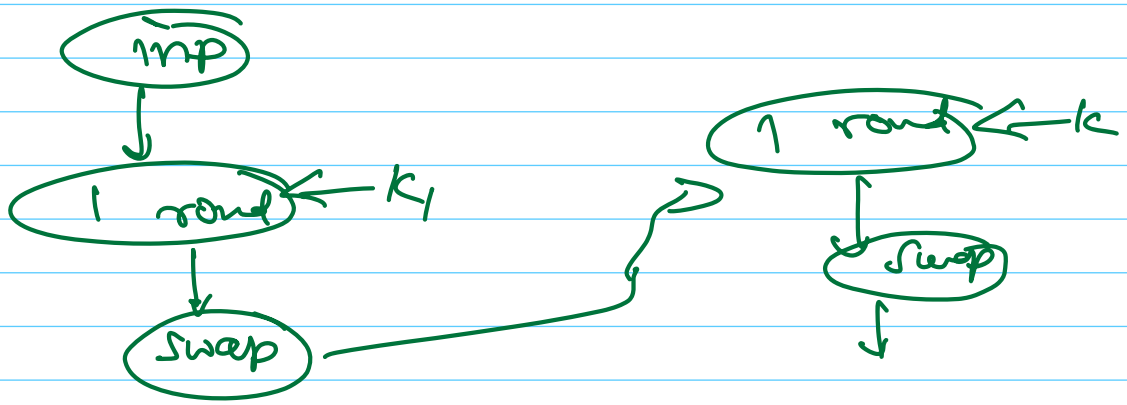
Inductive step: to prove that it works for (n+1) rounds



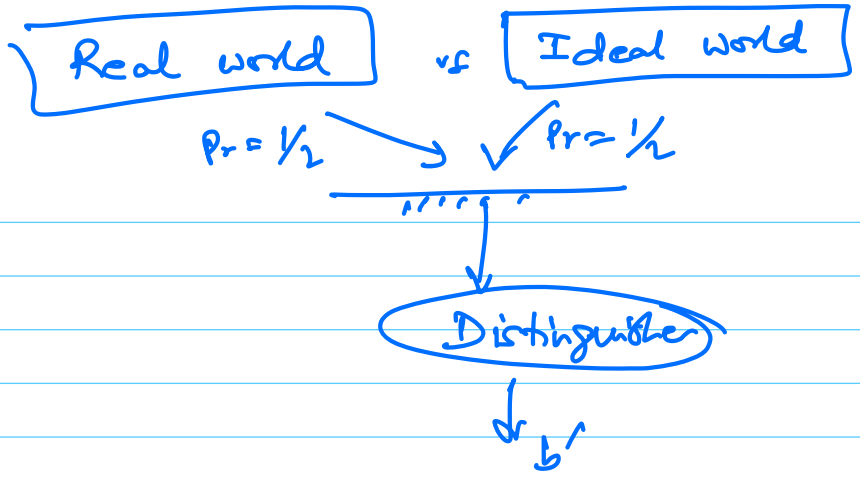
for (n+1) rounds





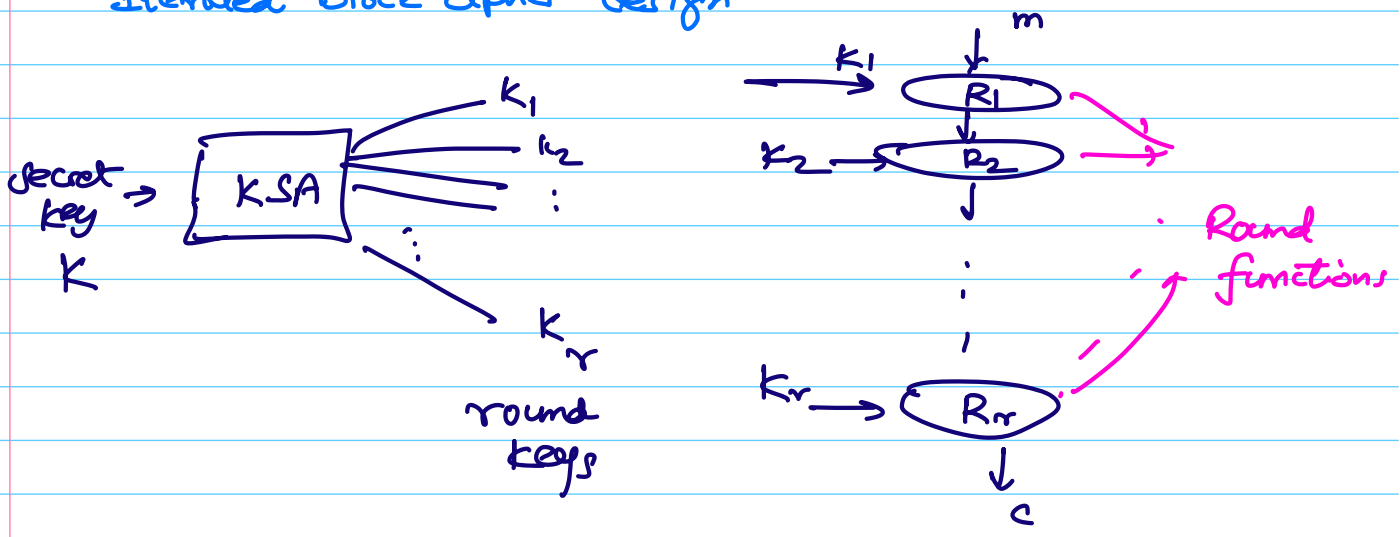


4 · feb 2025



Last class:

Iterated Block cipher design



Round function: DES had a round function designed as a feistel structure

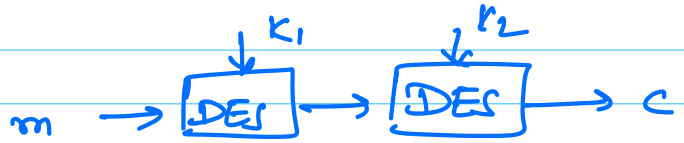
Issues with DES:

(i) hidden design – (fear of hidden trapdoors)

(ii) Small key size

(Lucifer had 128 bit keys)

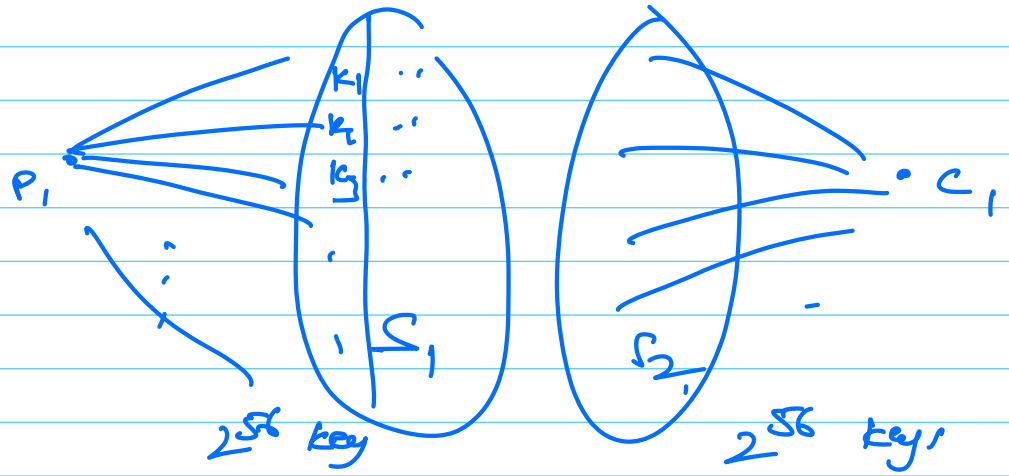
Can we increase the key size?



2-DES = 112 bit key

2-DES is not 112-bit secure.

Suppose we have $(P_1, C_1), (P_2, C_2), \ldots$



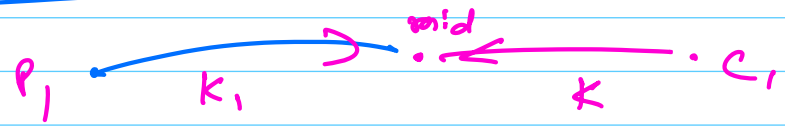
Practically,

(i) Create S_1 : effort $= 2^{56}$ calls to DES

(ii) Sort S_1 on ciphertext part

(iii) for loop for diff K^* $\{ Dec(K^*, C_1) \}$

if match:



$\Rightarrow$ Potential key $= K_1 \| K_2$

$=$ Cost $= 2^{56}$ calls to 2-DES

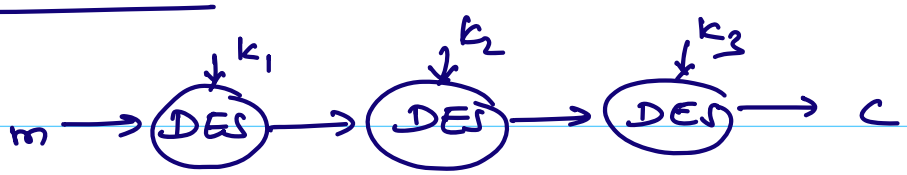
$+$ memory 2^{56} entries $(= 56 + 64$ bits$)$

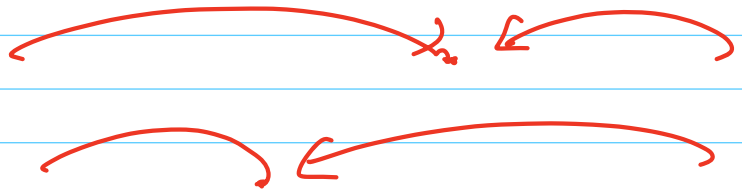
Check the potential keys with another pair (P_2, C_2)

$\Rightarrow$ w.h.p. only 1 candidate key is left

What next ?

3-DES





make $k_1 = k_3$ ⇒ 2 key 3-DES

≃ 112 bit security

Change management:

DES → 3DES in individual machines

k_1, k_1^{-1}, k_2

$= k_2$

90's - trapdoor fear

NIST - Call for proposal

- Open to world (~ 1997)

Advanced Encryption Standard — competition

Evaluation criteria - (i) security

(ii) efficiency in both s/w & h/w

(iii) elegance & resistance to cryptanalysis

(iv) no patent

(available world-wide without any royalty)

~ 3 years of effort - public scrutiny
3 workshops

3rd round - 5 designs left

✓ (i) Rijndael - Rijmen & Daemen
(ii) Twofish
(iii) TIGER
(iv) ..
(v) ..

AES: Standard 128 bit block
keys $\in \{128, 192, 256\}$

$\Rightarrow$ AES-128, AES-192, AES-256

(Rijndael supported block sizes of 128, 192, 256)

Iterated Block cipher.

Based on finite field operations

Group: Set $(S, *)$ | Abel

Additional property:
if $\forall a, b \in S$
$a * b = b * a$
then group is commutative

(i) Closure if $a, b \in S$, then $a * b \in S$

(ii) Associativity if $a, b, c \in S$
then $(a * b) * c = a * (b * c)$

(iii) Identity special element $e \in S$
s.t $\forall a \in S \quad a * e = a$
$e * a =$

(iv) inverse $\forall a \in S$

$\exists b \in S$ s.t. $a * b = e$

denoted as $b = a^{-1}$

$\mathbb{Z}_p^* = \{1, 2, 3, \ldots (p-1)\}$ for a prime p

operation = $\times \bmod p$

$a \times b \neq 0 \bmod p$

Inverse?

for any $a \in \mathbb{Z}_p^*$ $\exists b \in \mathbb{Z}_p^*$ s.t.

$ab \equiv 1 \bmod p$

Proof:

$\mathbb{Z}_p^* = \{1, 2, 3, \ldots (p-1)\}$

$a \cdot \mathbb{Z}_p^* = \{a*1, a*2, a*3, \ldots a*(p-1)\}$

Suppose $a \cdot i = a \cdot j \bmod p$ → Impossible for distinct i, j

$\Downarrow$

$a(i-j) = 0 \bmod p$

Finite field: $(S, +, \cdot)$

(i) $(S, +)$ is a commutative group, with 0 as additive identity

(ii) $(S - \{0\}, *)$ is a commutative group

(iii) $+$, $*$ distribute

$$a * (b+c) = (a*b) + (a*c)$$

6-feb-25

Recall: Group: $(G, *)$

Necessary

(i) closure: $\forall a, b \in G, \; a*b \in G$

(ii) Associativity: $\forall a, b, c \in G,$

$$a*(b*c) = (a*b)*c$$

(iii) Identity: $\exists e \in G$ s.t.

$\forall a \in G$

$$e*a = a*e = a$$

(iv) Inverse: $\forall a \in G$

$\exists b \in G$

s.t $a*b = e$

Additional: (v) Commutativity

$$\forall a, b \in G, \quad a*b = b*a$$

when it is satisfied for a group G, G is called an Abelian group

Consider: $G = \{ P(x) \text{ of degree} \leq 7 \}$

$G = \{ (a_0 + a_1 x + a_2 x^2 + \dots + a_7 x^7) \}$

where coefficients $\in GF(2)$

$a_0, a_1, \dots a_7 \in \{0, 1\}$

Representation:

1 byte

$* \Rightarrow \quad a(x) * b(x)$

$= \quad a(x) + b(x) \mod 2$ (for each coeff)

$P_1(x) = x + 1$

$P_2(x) = x^2 + x$ $\Rightarrow$ $P_1(x) + P_2(x) = x^2 + 1$

Notice that this creates a group.

$\rightarrow$ Abelian group

What if we wanted multiplication of polynomials?

$P_1(x) \cdot P_2(x) \mod$ (degree 8 poly. $p(x)$) (terms mod 2)

$(x^8 + \dots -)$

irreducible polynomial

Consider Z_p^*

$Z_n = \{0, 1, 2, \dots (n-1)\}$

$\{0\}$

$* =$ multip. mod n

Is it a group? —— if n is prime then already proved

what if n is composite.

$\mathbb{Z}_6 - \{0\} = \{1, 2, 3, 4, 5\} \mod 6$

$2 * 3 = 0 \rightarrow$ Not closed

$$\mathbb{Z}_n^* = \{ x : 1 \leq x < n \text{ s.t } \gcd(x, n) = 1 \}$$

$\mathbb{Z}_6^* = \{1, 5\}, * \mod 6$

$\mathbb{Z}_{10}^* = \{1, 3, 7, 9\}$

$\mathbb{Z}_{12}^* = \{1, 5, 7, 11\}$

$7^{-1} = 3$

$3^{-1} = 7$

AES polynomial:

11 B

0001 0001 1011

1 1 B

$= (x^8 + x^4 + x^3 + x + 1)$

Coeff' are mod (2) $\equiv$ Coeff are in GF(2)

Galois field.

Field: $(G, +, \cdot)$

(i) $(G, +)$ is an Abelian group with identity as $\overline{0}$

(ii) $(G - \{0\}, \cdot)$ is an Abelian group

(iii) Distributive operation

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$
$$\forall\ a, b, c \in G$$

GF(2)

$\{0, 1\}$ + mod 2
× mod 2

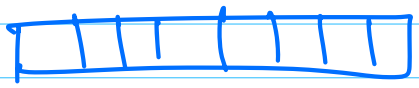
⊕ AND

+	0	1
0	0	1
1	1	0

×	0	1
0	0	0
1	0	1

Rijndael finite field / AES f.f.

Contains = 2^8 elements



← 8 bits → 0 or 1

+ ≡ usual addition of polynomials with coeff ∈ GF(2)

① Every finite field of same size is isomorphic to each other

② The no. of elements in a finite field are p^n for some prime p & some int n.

Most common finite fields in Crypto are

$GF(2^n)$

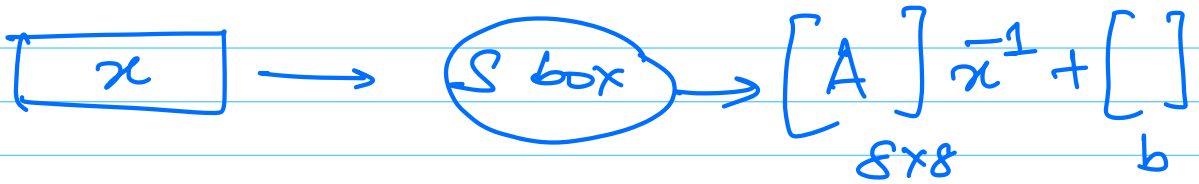
AES finite field = $GF(2^8)$

Polynomial of degree ≤ 7 with coeff ∈ GF(2)

mod 11B

S-box in AES.

8 bit → 8 bit



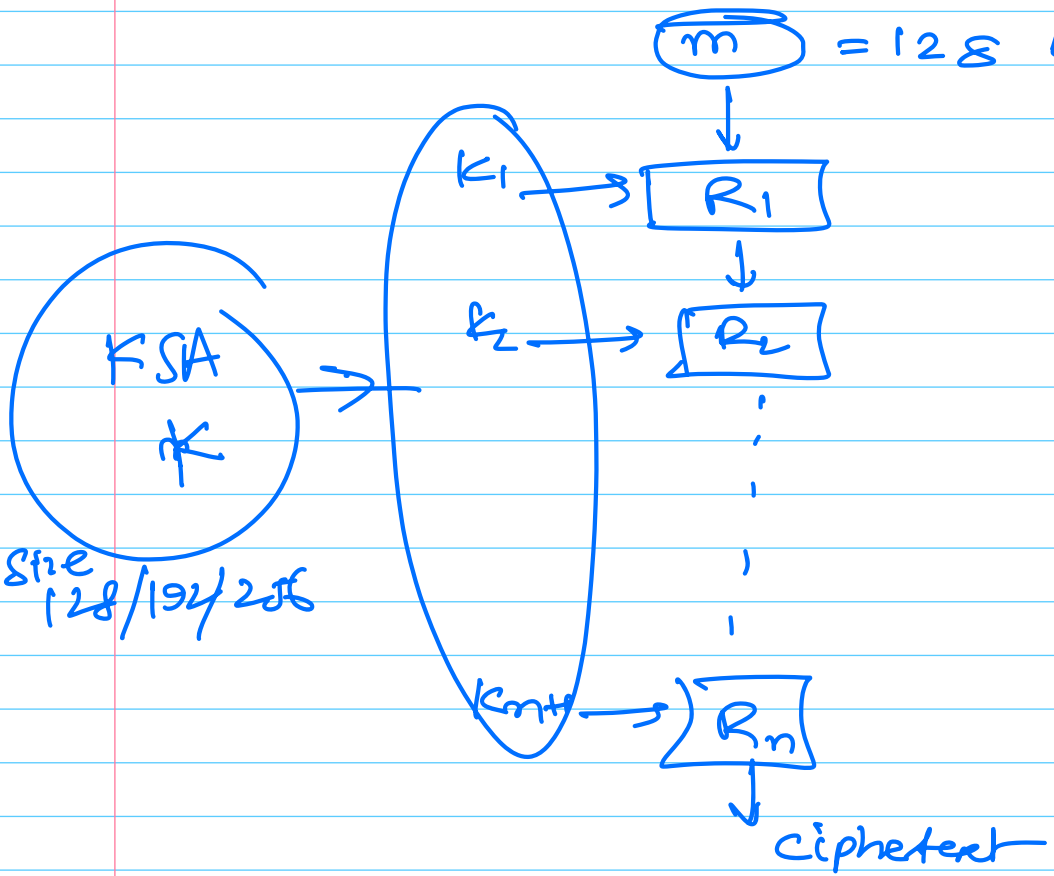
Where we take $\vec{0}$ as

AES : Iterated block cipher

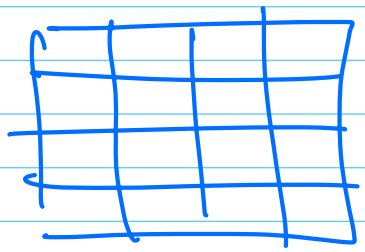
AES-128 : 10 rounds

AES-192 : 12 rounds

AES-256 : 14 rounds



Each round has 4 operations, which are implemented on a state array of 128 bits

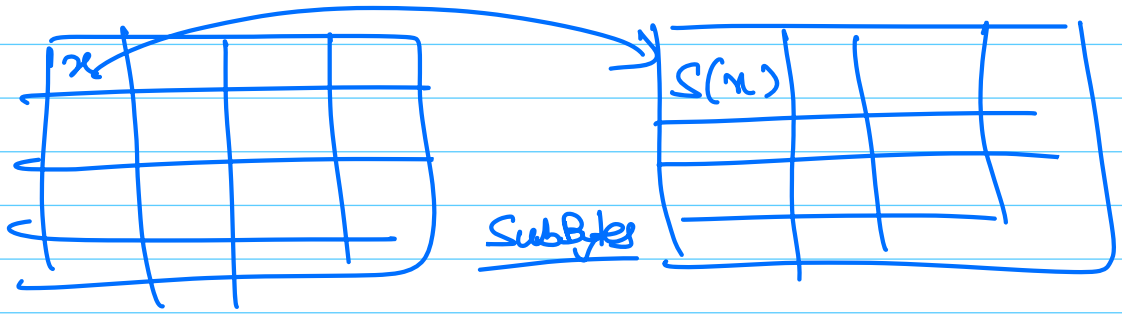
State 

4x4 matrix of bytes

16x8 = 128 bits

Round function: Round i

1. SubBytes(S)
2. ShiftRows(S)
3. MixColumns(S)
4. ~~Add~~ Round Keys(S, Ki)



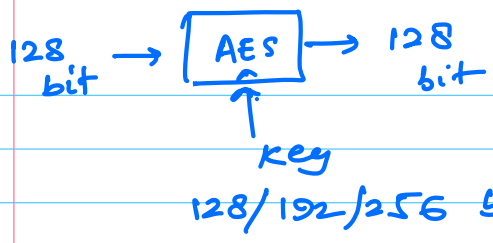
$$S(x) = Ax^{-1} + b$$

some const.

Precomputed S Box size:

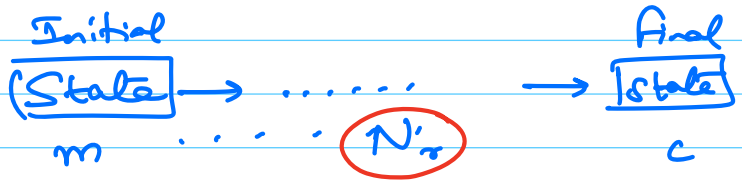
2^8 inputs x 1 byte each = 256 bytes

11 Feb 2025



+ some mode of operation (for domain extension)

Input: state array of 4×4 bytes = 16 bytes = 128 bits



	secret key size (bits)	No. of rounds (N_r)
AES-128	128	10
AES-192	192	12
AES-256	256	14

AES-128

Round keys $(K_0, K_1, K_2, \ldots, K_{10})$ = AES keyschedule (K)

$K_i \in \{0,1\}^{128}$

K ← secret key

State[0] = input msg

m_0	m_4		
m_1	m_5		
m_2	.		
m_3	:		

key whitening → State[0] = State[0] ⊕ K_0

for (i = 1 to 9) ← first 9 rounds

{

Round(i, State[i], K_i);

}

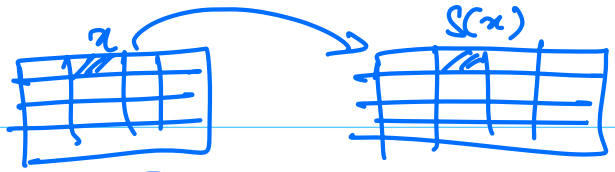
Final Round (10, State[10], K_{10});

→ Last step in final Round will also be key XOR

Round function:

inverse in $GF(2^8)$

(i) SubBytes

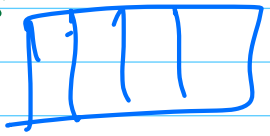


$S(x) = A x^{-1} + b$

(ii) ShiftRows

→ circular rotation of each row by a constant

(iii) MixColumns



→ Replace each column by a different column

(iv) AddRoundKey

→ new state

$C = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$ → 1 column of 4 bytes

$\equiv a_0 + a_1 x + a_2 x^2 + a_3 x^3$

where $a_i \in GF(2^8)$

$C * M(x) = (\ldots)$ → degree may increase

divide by a polynomial of degree = 4

$(x^4 + \ldots)$

$$(a_0 + a_1 x + a_2 x^2 + a_3 x^3) * (b_0 + b_1 x + b_2 x^2 + b_3 x^3) \quad \text{mod } (x^4+1)$$

$$= a_0 b_0 +$$
$$(a_0 b_1 + a_1 b_0)\, x +$$
$$(a_0 b_2 + a_1 b_1 + a_2 b_0)\, x^2 +$$
$$(a_0 b_3 + a_1 b_2 + a_2 b_1 + a_3 b_0)\, x^3 +$$
$$(a_1 b_3 + a_2 b_2 + a_3 b_1)\, x^4 +$$
$$(a_2 b_3 + a_3 b_2)\, x^5 +$$
$$(a_3 b_3)\, x^6$$

Notice:

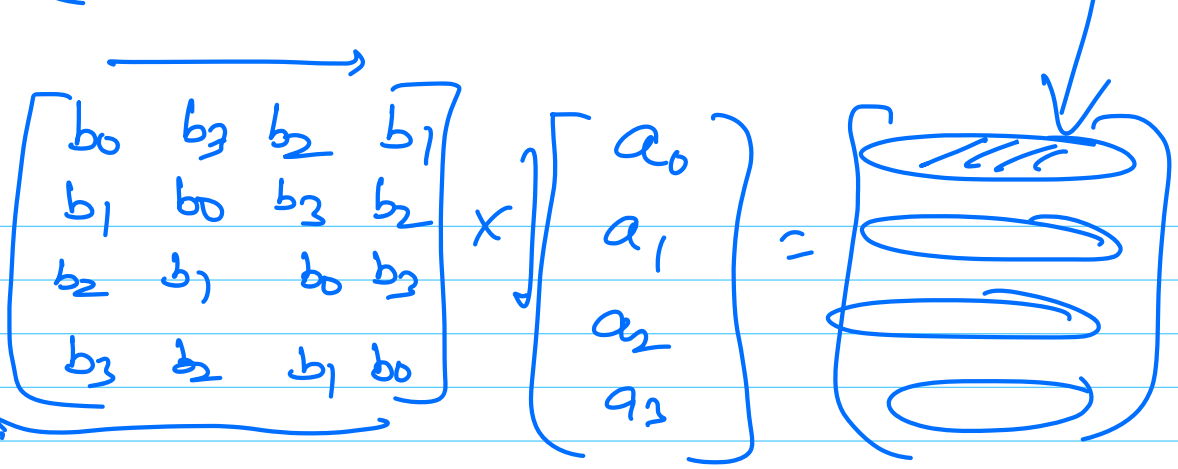
$$x^4 / (x^4+1) = \frac{x^4+1+1}{x^4+1} = 1$$

$$x^5 / (x^4+1) = x$$

$$x^6 / (x^4+1) = x^2$$

$\Rightarrow$

$$(a_0 b_0 + a_1 b_3 + a_2 b_2 + a_3 b_1)$$
$$+ (a_0 b_1 + a_1 b_0 + a_2 b_3 + a_3 b_2)\, x$$
$$+ (a_0 b_2 + a_1 b_1 + a_2 b_0 + a_3 b_3)\, x^2$$
$$+ (a_0 b_3 + a_1 b_2 + a_2 b_1 + a_3 b_0)\, x^3$$



Unfortunately, (x^4+1) is not irreducible

$$(x^4+1) = (x^2+1)^2$$

$\Rightarrow$ this matrix arose from a polynomial

$$b(x) = (b_0 + b_1 x + b_2 x^2 + b_3 x^3)$$

Ensure that $b(x)$ is coprime to (x^4+1)

Why is last round different?

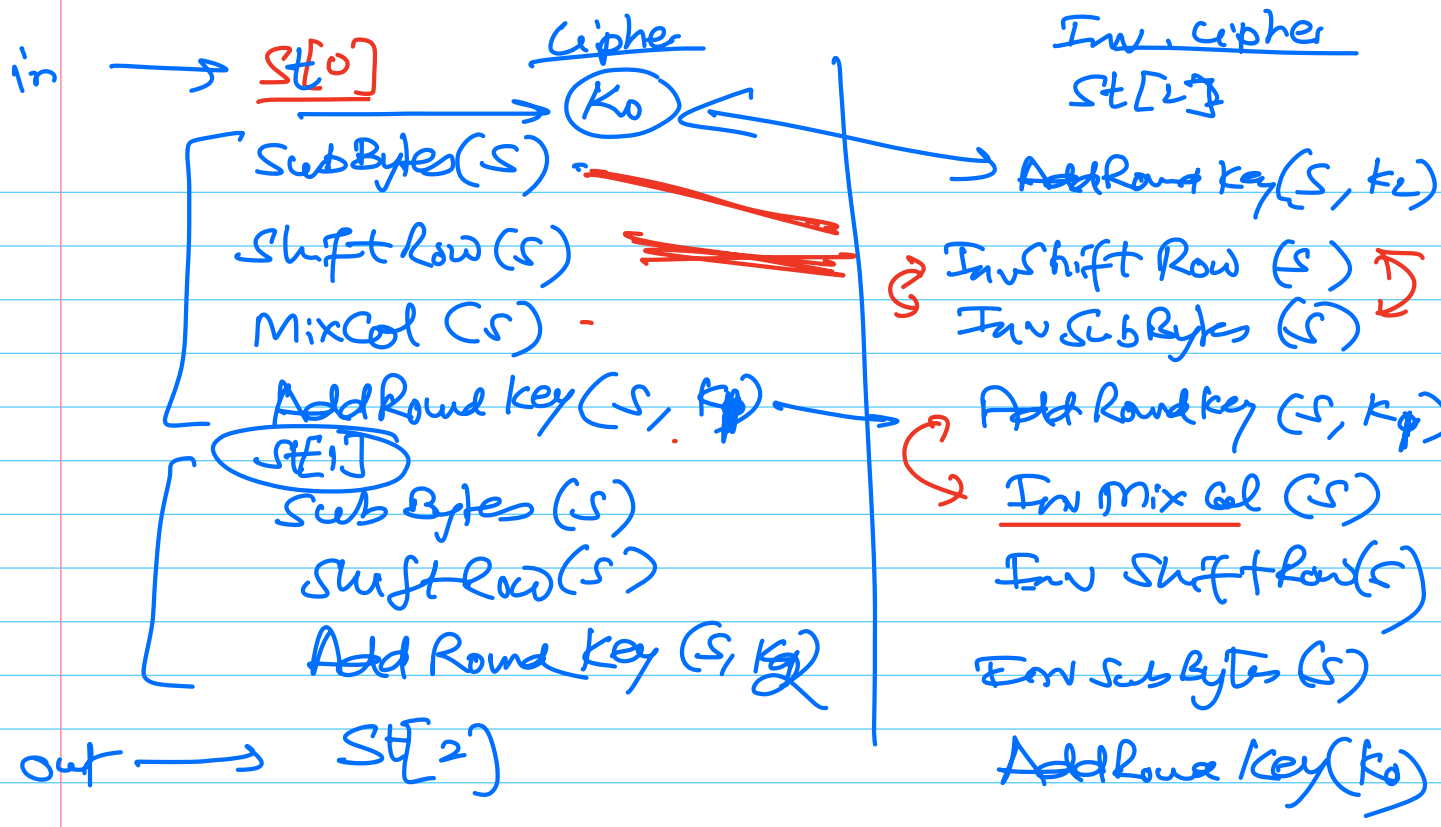
to make the structure of encryption & decryption operations same

SubBytes

ShiftRow

~~MixCol~~

~~Add~~ RoundKey



MixCol(S) → M.S

AddRound(S, K) → S ⊕ K

→ modify the key to reverse the operation

$$M \cdot (S \oplus \vec{m}^{-1} K) = MS \oplus K$$

Cipher Round keys:

$K_0, K_1, K_2 \ldots K_{10}$

Inv round keys:

$K_{10}, (K_9)', (K_8)' \ldots (K_1)', K_0$

modified keys

SPN ciphers -

Substitution permutation Network

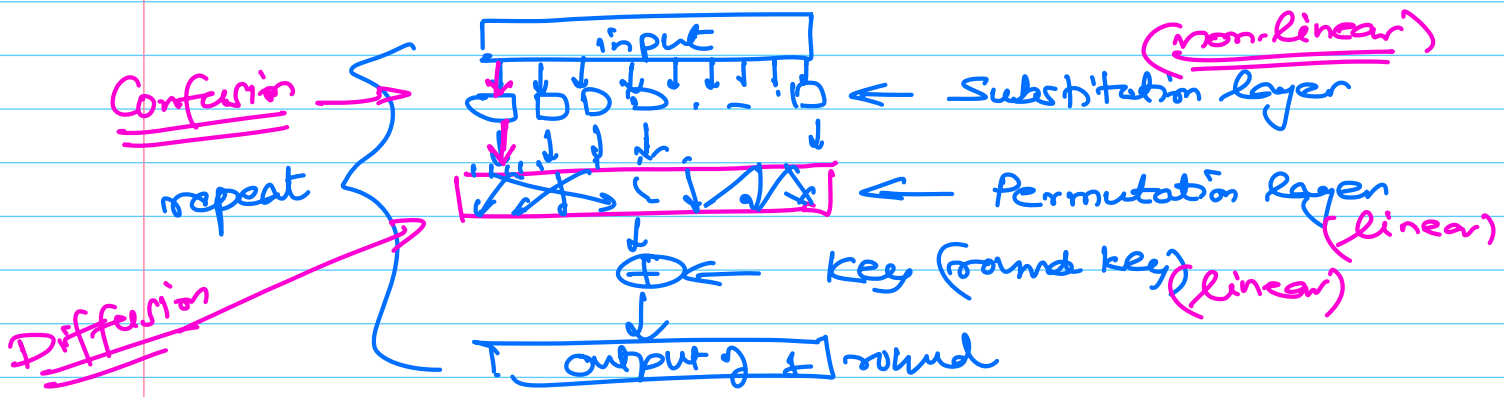
AES design document

— NIST

for ref.

13-02-2025

SPN structure:



'if a cryptosystem is linear Then it is easy to break. → $C = Ax + k$

$= [A'][x\ 1]$

Gaussian elimination → solve for A'.

Ref. → Hill cipher

Avalanche effect — a small input change causes drastic change in the output

(Caused by combination of Subs. & Perm. layers)

AES- a very strong block cipher
Has withstood the test of time

So far- **Confidentiality**

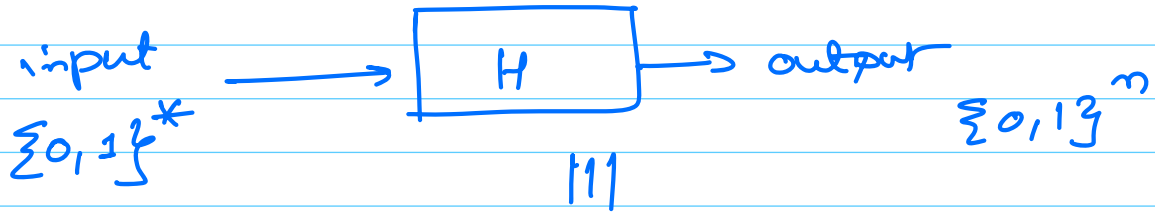
↳ achieved using encryption

Integrity : data is not modified in transit

Today's class: a cryptographic object which does not have a secret key & yet has some security guarantees

Object:

Hash function

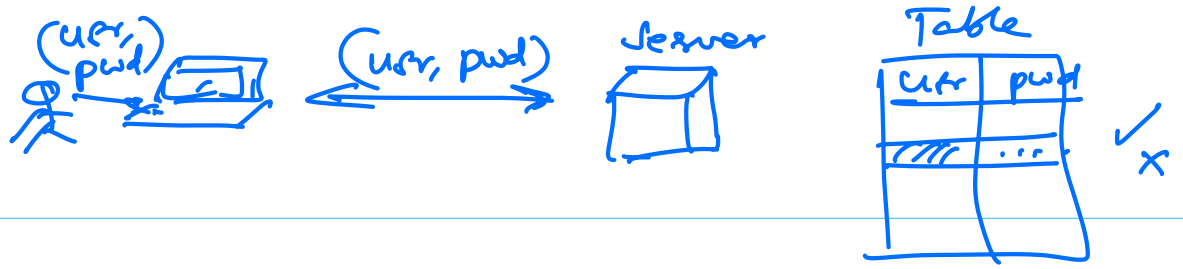


Random Oracle:

pick a random x, y : $\Pr[f(x) = y] = \frac{1}{2^n}$

① **Use-case 1:**

Password security



A very bad idea

Why?

data breach → devastating

Store the passwords as —

hash (pwd)

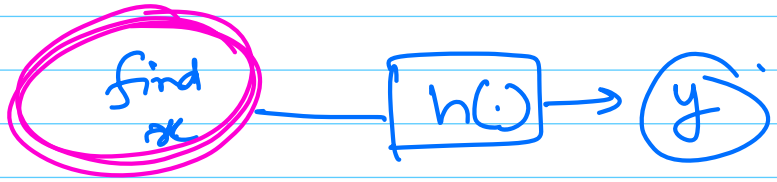
usr	h(pwd)
⋮	⋮

if someone knows h(pwd), it is computationally hard to find pwd.

Preimage resistance:

given h(x) it is computationally hard to find x

Generic attack:

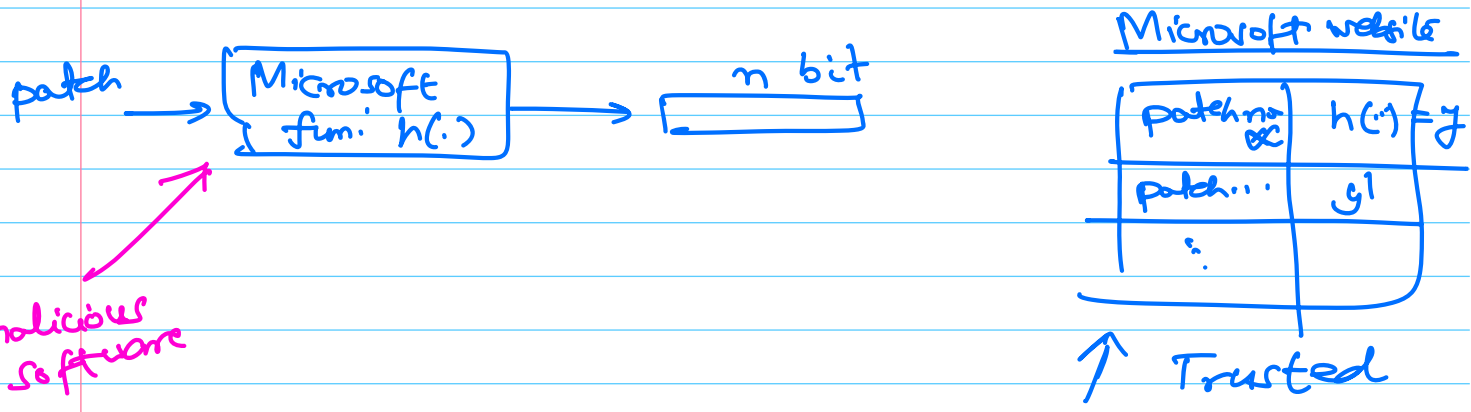


y is known

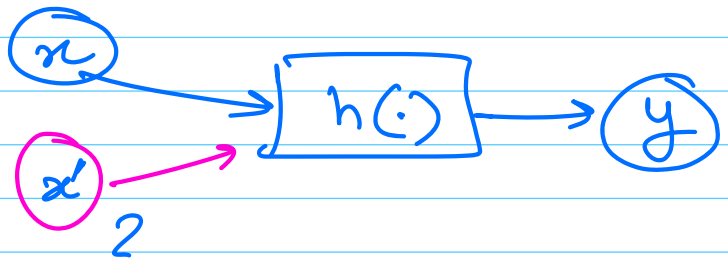
effort = 2^n

② Use-case no. 2

Software download



No one should be able to create a malicious software which has the same hash value as the original patch.



$x \neq x'$

Second Preimage attack:
Given (x, y), it should be computationally hard to find an x' s.t $h(x') = y$

Generic attack:

input $x \longrightarrow h(x) = y$

give y to preimage attacker
— success in $\geq 2^n$ trials

2^{nd} preimage in $(2^n + 1)$ trials

③ Use-case 3:

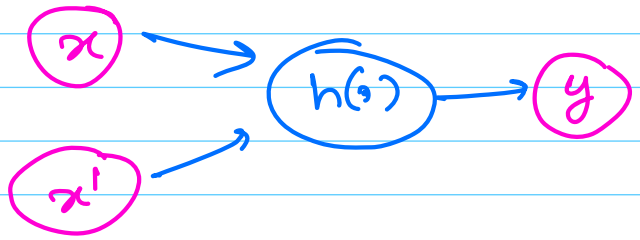
Co-development of software

- we want to check if contents of the file got modified

Small digest of the document

Collision resistance:

$x \neq x'$



It is computationally hard to find two diff. inputs which have the same output.

Collision attacker $\leq$ Second preimage attacker

Jan	
feb	18
Mar	5
Apr	
May	13, 5, 25, 27
June	27, 7, 23

July.	11, 3
Aug	- 13, 23, 12, 19
Sep	- 9, 6, 17, 29, 1, 6, 27, 16, 15
Oct	- 17, 2, 21, 12
Nov	- 3, 18,
Dec	- 29, 27 (13, 13)

Collisions happen with high pr. in about $2^{n/2}$ trials.

$x \longrightarrow h(x)$ $\quad$ $h(\cdot)$ is a random oracle

$S = \{x_1, x_2, \ldots x_t\}$, N = size of output set

Pr that there is some collision in S

$$Pr(\text{collision}) = 1 - Pr(\text{no collision})$$

$$p = 1 - \left(1 \cdot \frac{N-1}{N} \cdot \frac{N-2}{N} \cdots \frac{N-t+1}{N}\right)$$

What if $t \geq 366$, Pr = 1

$$p = 1 - \prod_{i=1}^{t-1} \left(1 - \frac{i}{N}\right)$$

assume $i << N$

$$(1 - i/N) \approx e^{-i/N}$$

$$p = 1 - \prod e^{-i/N}$$

$$= 1 - e^{\frac{-1}{N}\left(\frac{(t-1)(t)}{2}\right)}$$

$$e^{-\frac{1}{2N} t(t-1)} = 1 - p$$

$$-\frac{1}{2N} \cdot t(t-1) = \ln(1-p)$$

$$\frac{1}{2N} \cdot t(t-1) = \ln\left(\frac{1}{1-p}\right)$$

$$t^2 \simeq 2N \cdot \ln\left(\frac{1}{1-p}\right)$$

$$\boxed{t \simeq \sqrt{N} \cdot \left(2 \cdot \ln \frac{1}{1-p}\right)}$$

const

for $N = 365$

$p = 1/2$ $t = 23$

$p = 0.75$ $t = \dots$

$p = 0.99$ $t = 90$

for $p = ?$

const = 1.4

Birthday paradox

(Not really a paradox, but a counter-intuitive statistical fact)

18 Feb 2025

Cryptographic Hash functions:

Main security properties:

① **Preimage resistance:**

Given a random y, it should be computationaly hard to find an x s.t.

$$h(x) = y$$

② **Second Preimage Resistance**

Given x, it should be comp. hard to find an x' s.t.

$$x \neq x', \quad h(x) = h(x')$$

③ **Collision resistance:**

find x, x' s.t. $x \neq x'$,

$$h(x) = h(x')$$

Generic attacks:

① .. 2^n

② .. 2^n } calls to $h(\cdot)$

③ .. $2^{n/2}$ → "Birthday Paradox"

In reality, hash functions are "swiss army knife"

e.g. **near collision resistance:**

it should be hard to find x, x' s.t. $x \neq x'$ $(h(x) \oplus h(x'))$ has low Hamming weight

$x = 10100$

$y = 11010$

$\cdots$

$hw(x, y) = 3$

Syntax: $h(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^n$

- produces an n-bit digest of an arbitrary length string
- output looks "random" & "unpredictable"

Random Oracle

even if you have made q queries to $h(\cdot)$

Queries
$x_1 \rightarrow y_1$
$x_2 \rightarrow y_2$
$\vdots$
$x_q \rightarrow y_q$

$\Pr[h(x^*) = y^*]$ ← any random y^*
where x^* has not been queried
$= 1/2^n$

the only way to know the response on a query is to compute $h(\cdot)$ on that query
→ no "shortcuts" are possible

How do we design such hash functions?

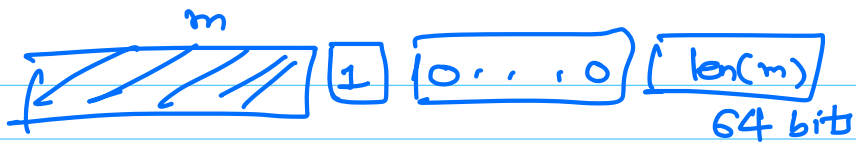
First problem: How to handle arbitrary length messages?

(i) Ensure by pre-processing that the input is multiple of some block length

→ $\text{padding}(m) = 10^* \,||\, \text{length}(m)$ in bits

↑ how many bits?

typically. block size = 512 bits



this padding permits only 64 bits for the length of the message.

⇒ the largest no which can be written in 64 bits $= (2^{64}-1)$

We can handle messages of length $(2^{64}-1)$ bits or less.

→ Not a practical restriction

Merkle-Damgard (MD) padding scheme

Bad example of designing a hash function:

padded message: $m =$ m_1 m_2 m_3 m_4 . . m_L

$h(m) = m_1 \oplus m_2 \oplus m_3 \oplus \cdots \oplus m_L$

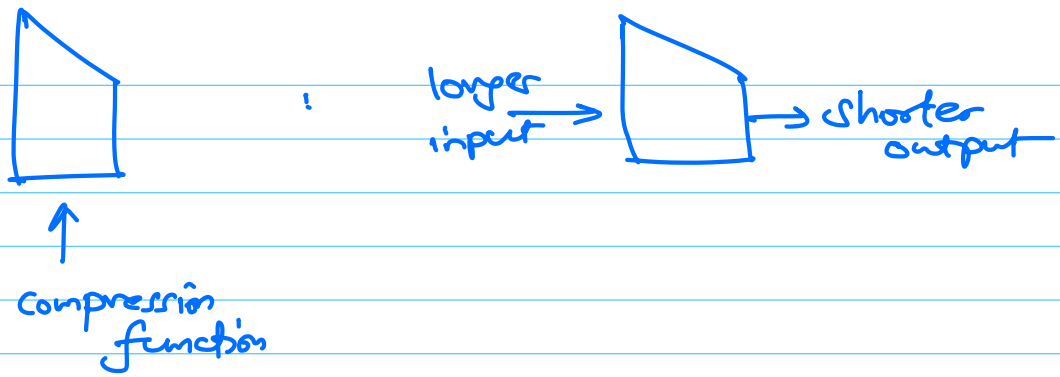
Preimage attack: output $= y$

input $= y \| x \| x \| z \| z \| \cdots$

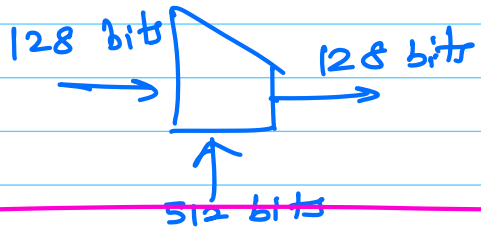
output $= y$ ...

input $= y$

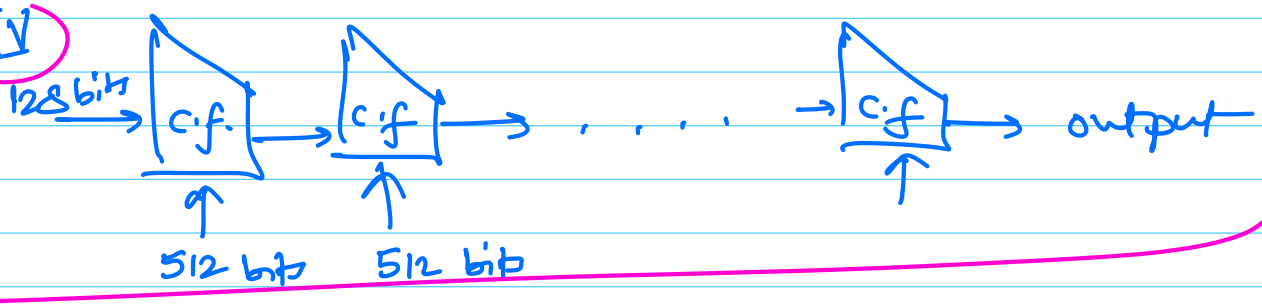
MD Design:



Example:



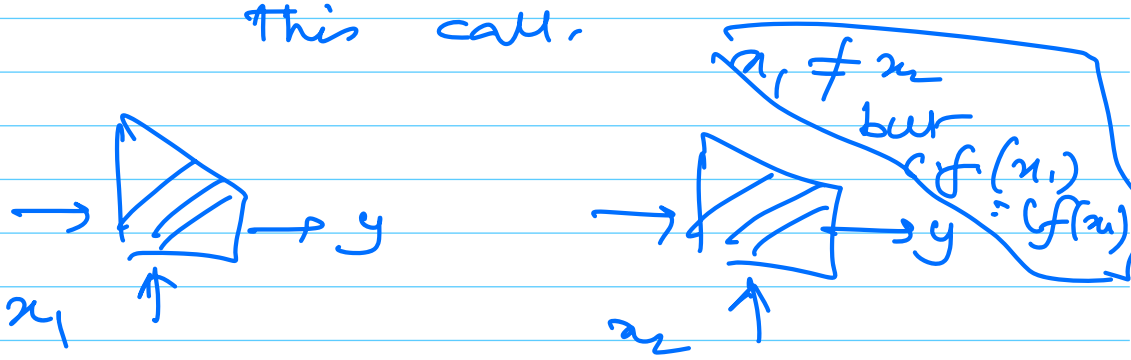
IV



MD theorem:

If $h(m_1) = h(m_2)$ for two different inputs m_1 & m_2

then there must be a collision for compression function somewhere in this call.



if compression function is collision resistant
then the hash function is collision resistant

Example functions

			Output size	No. of rounds
Rivest →	MD4	–	128	48
" →	MD5	–	128	64
NIST older	SHA-0	–	160	80
new	SHA-1	–	160	80
SHA-2	SHA-256		256	64
	SHA-512		512	80

Secure Hash Algorithm = SHA

SHA-1 collisions – Marc Stevens.

SHA-3 competition:

Keccak – Bertoni, ... Daemen

Sponge structure

As of today NIST → SHA-3

Proof of work in Bitcoin

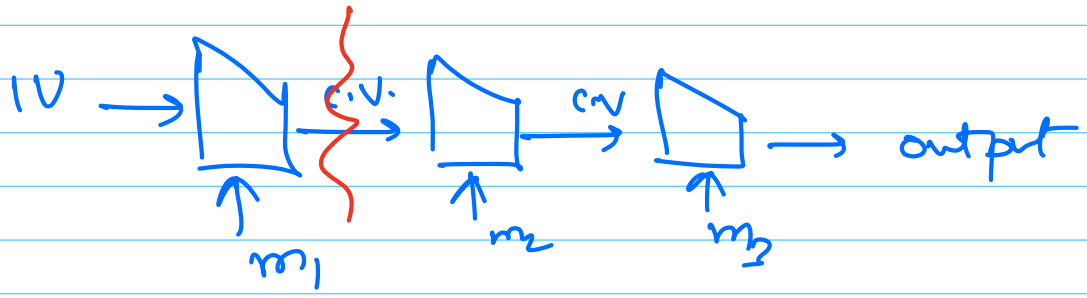
find an x s.t $h(x) = 0\ldots0$ ☐

What about IV in MD Design

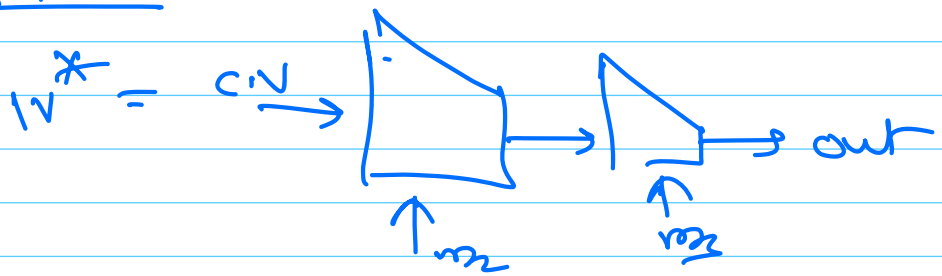
What if allow IV to be chosen like in block ciphers?

$h(m) =$ IV || val.

✗ Bad



Attack:



Solution:

Do not allow IV to be chosen

make IV to be fixed.