



# Network Security

## CSI ZG513 / ES ZG513 / SS ZG513

**BITS** Pilani  
K K Birla Goa Campus

Hemant Rathore  
Department of Computer Science and Information Systems

**Lecture Session – I**

# Introduction

---

- **Instructor**

Hemant Rathore

D - 155

Department of Computer Science & Information Systems

BITS Pilani, K K Birla Goa Campus

[hemantr@goa.bits-pilani.ac.in](mailto:hemantr@goa.bits-pilani.ac.in)

Contact Sessions:

Microsoft Teams

Course Page:

<http://taxila-aws.bits-pilani.ac.in/course/view.php?id=8972>

# Course Objectives

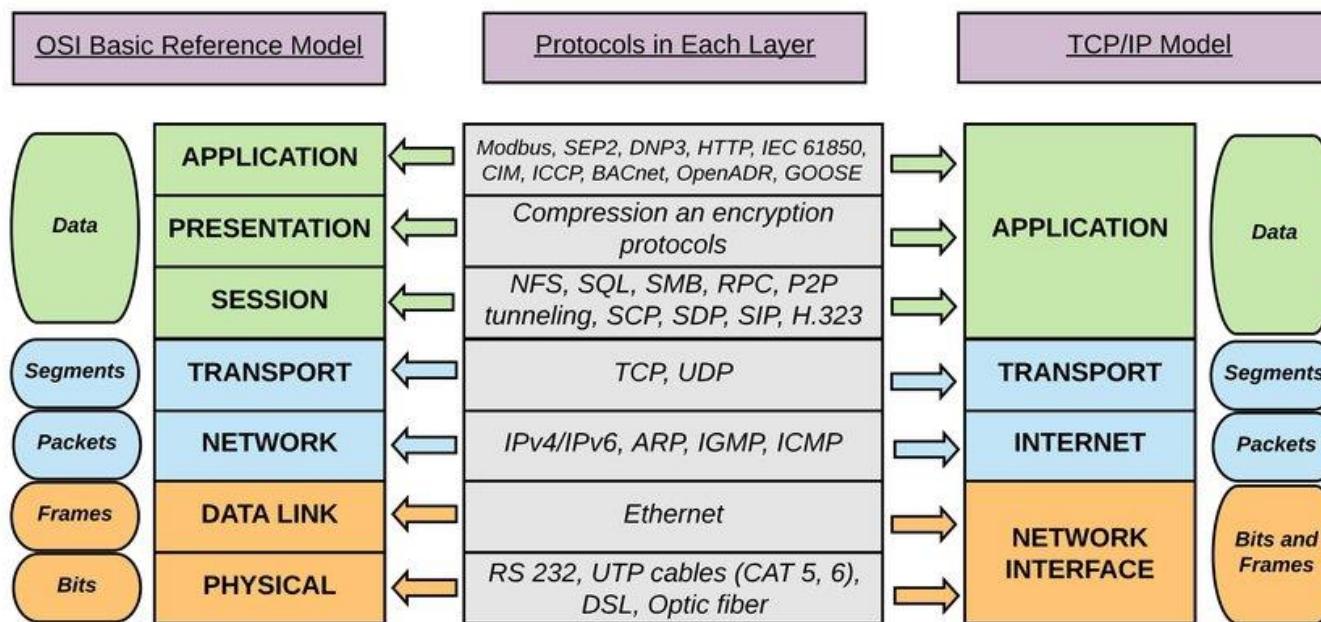
---

- Network security is an important area of information technology. This course on Network Security help audience to understand the three important security goals in the network communication–
  - Confidentiality
  - Integrity
  - Availability
  - Also Authenticity and Non-repudiation and cryptographic techniques to implement these security goals.



# Course Objectives

- The course provides a top down approach to explore the security implementations in different layers - application, transport and network.



# Course Objectives

---

- The course provides a necessary review of mathematical concepts to implement different cryptographic techniques to achieve the network security goals and then provides a deeper dive to the field of cryptography - symmetric and asymmetric key cryptography and methods to implement them.
- The course consolidates and sums up the learning taking few case studies and examples from latest trends and industry deployments.

# Text & Reference Books

---

- Stallings William: Cryptography and Network Security - Principles and Practice, Pearson India, 6th Edition, 2014.
- Forouzan B A, Mukhopadhyay Debdeep : Cryptography and Network Security, McGraw Hill, 2nd Edition, 2010.
- Schneier Bruice: Applied Cryptography : Protocols, Algorithms And Source Code In C, Wiley India, 2nd Edition, Reprint – 2013
- Kurose James F and Keith W. Ross: Computer Networking: A Top-Down Approach, Pearson India, 5th Edition, 2012.

# Text & Reference Books

---

- Christof Paar and Jan Pelzl: Understanding Cryptography - A Textbook for Students and Practitioners, Springer, 1st Edition, 2010.
- Being a WILP level course, no single book is sufficient.
- Materials from different sources.

# Evaluation Scheme

---

- **EC1:** Three Quizzes, each of 5% weightage.
  - Quiz-I      August 16 to 30, 2022
  - Quiz-II      September 16 to 30, 2022
  - Quiz-III      October 16 to 30, 2022
- **EC2:**
  - Mid-Semester Test 35% (Open Book)      23/09/2022 (FN)
  - Mid-Semester Make-up Test 35% (Open Book)      07/10/2022
- **EC3:**
  - Comprehensive Exam 50% (Open Book)      25/11/2022 (FN)
  - Comprehensive Make-up Exam 50% (Open Book)      02/12/2022 (FN)

# Contact Sessions

---

- 22 Lectures each of 50 minutes.
- CS-1:
  - Network Security and OSI Security Architecture
  - Review of Attacks, Mechanisms and Services, Network Security Model
- CS-2:
  - Network Security Model
  - Techniques to Implement Network Security

# Contact Sessions

---

- CS-3:
  - Cryptography, Classical Encryption
  - Breaking the Cryptosystem
- CS-4:
  - Modular Arithmetic, Groups and Rings
  - One example each in classical substitutive and transposition cipher
- CS-5:
  - Random numbers, its types and usage.
  - TRNG, PRNG, CSPRNG
  - Review of BBS

# Contact Sessions

---

- CS-6:
    - Stream Ciphering
    - RC4 algorithm
  - CS-7:
    - Basic Number Theory
    - Extended Euclidean Algorithm.
  - CS-8:
    - Galois Field
    - Polynomial Arithmetic
  - CS-9:
    - Block Ciphering
    - Confusion and Diffusion Theory
-

# Contact Sessions

---

- CS-10:
  - AES and its importance in security
  - Efficient implementation of AES.
- CS-11:
  - Recapitulation of all the sessions / problem solving before mid-semester exams
- CS-12:
  - Modes of Operation and its applications
  - Multiple Encryption and Meet-in-the Middle Attack
- CS-13:
  - SHA-1 and SHA-3
  - HMAC and CBC-MAC and its Security

# Contact Sessions

---

- CS-14:
  - Model of Asymmetric Key Cryptography
  - Factorization and other methods for Public Key Cryptography
- CS-15:
  - RSA and OAEP
  - Diffe-Hellman Key Exchange and its Security Aspects
- CS-16:
  - Distribution of Symmetric and Asymmetric Key
  - Digital Signature: DSA
  - X.509 Certificate
  - Man-in-the Middle Attack

# Contact Sessions

---

- CS-17:
    - User/Entity Authentication
    - Kerberos
  - CS-18:
    - Review of PGP - Authentication and Confidentiality
    - Review of MIME and S/MIME
  - CS-19:
    - Review of Web Security
    - Review of SSL and TLS
-

# Contact Sessions

---

- CS-20:
  - Malicious Software and its Detection Techniques
- CS-21:
  - Review of Intrusion and Intrusion Detection
- CS-22:
  - Recapitulation of all the sessions / problem solving before comprehensive exams.

# Quote

---

“The art of war teaches us to rely not on the likelihood of the enemy's not coming, but on our own readiness to receive him; not on the chance of his not attacking, but rather on the fact that we have made our position unassailable”

Sun Tzu

# Background

---

- Information Security requirements have changed in recent times.
  - Traditionally provided by physical and administrative mechanisms.
  - Computer use requires automated tools to protect files and other stored information.
  - Use of networks and communications links requires measures to protect data during transmission.
-

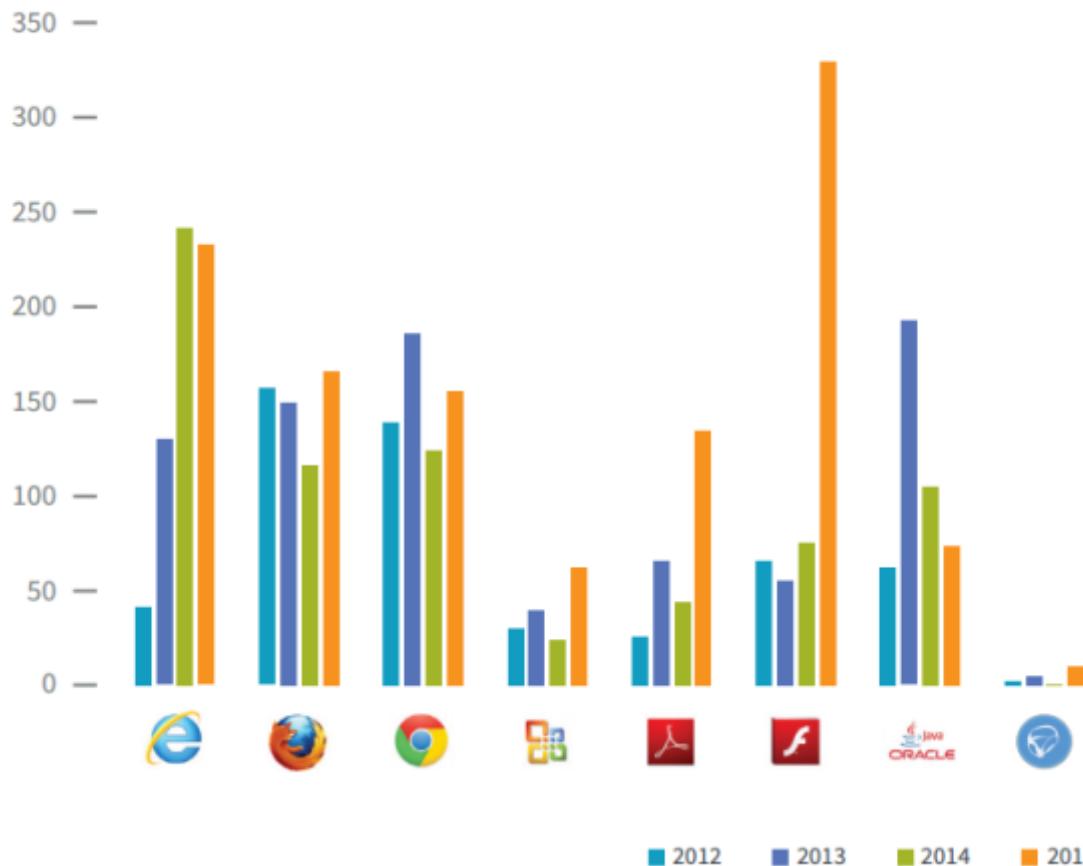
# Definitions

---

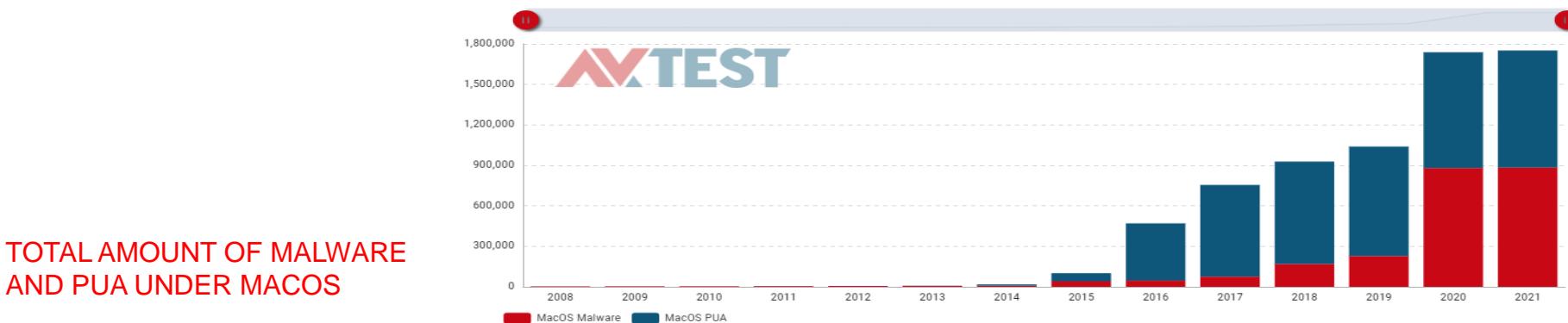
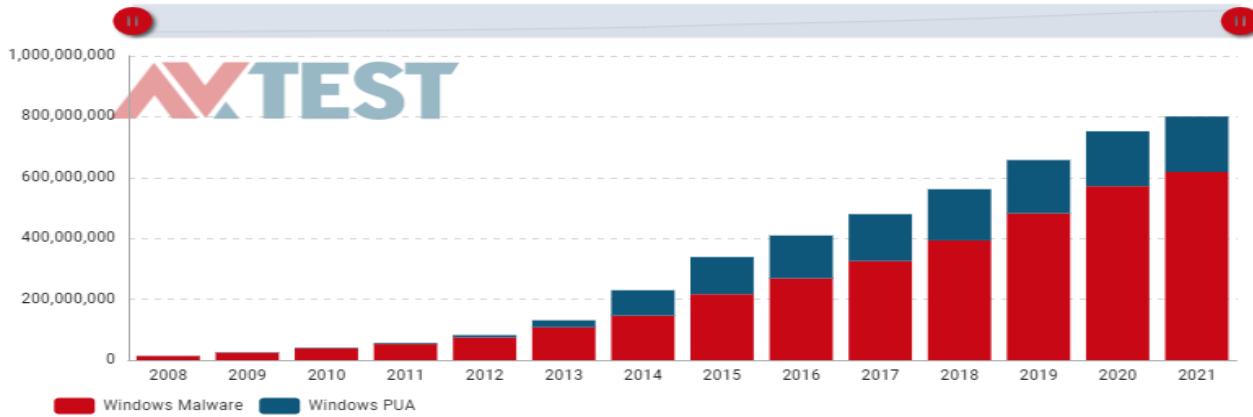
- **Computer Security:** generic name for the collection of tools designed to protect data and to thwart hackers.
  - **Network Security:** measures to protect data during their transmission
  - **Internet Security:** measures to protect data during their transmission over a collection of interconnected networks
-

# Security Trends

FIGURE 1: VULNERABILITY DYNAMICS 2012-2015

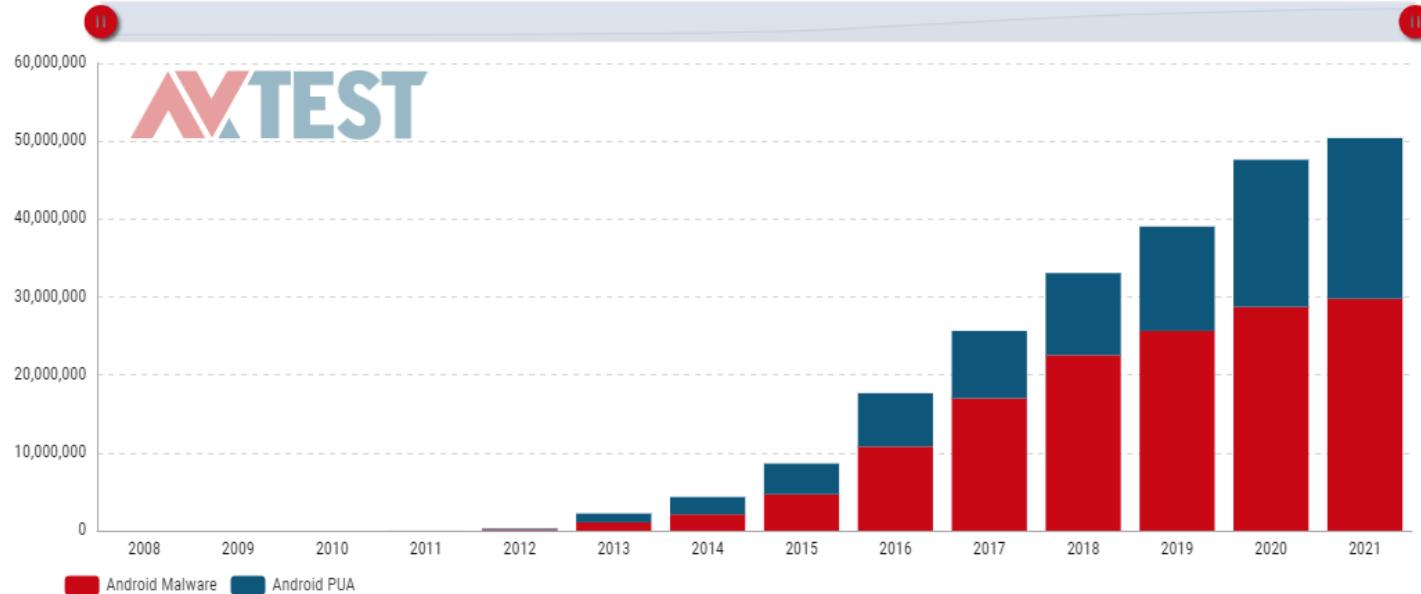


# Malware Growth (1)



Source L: <https://www.av-test.org/en/statistics/malware/>  
 Source R: <https://www.av-test.org/en/statistics/malware/>

# Malware Growth (2)



30 Jan 2018:

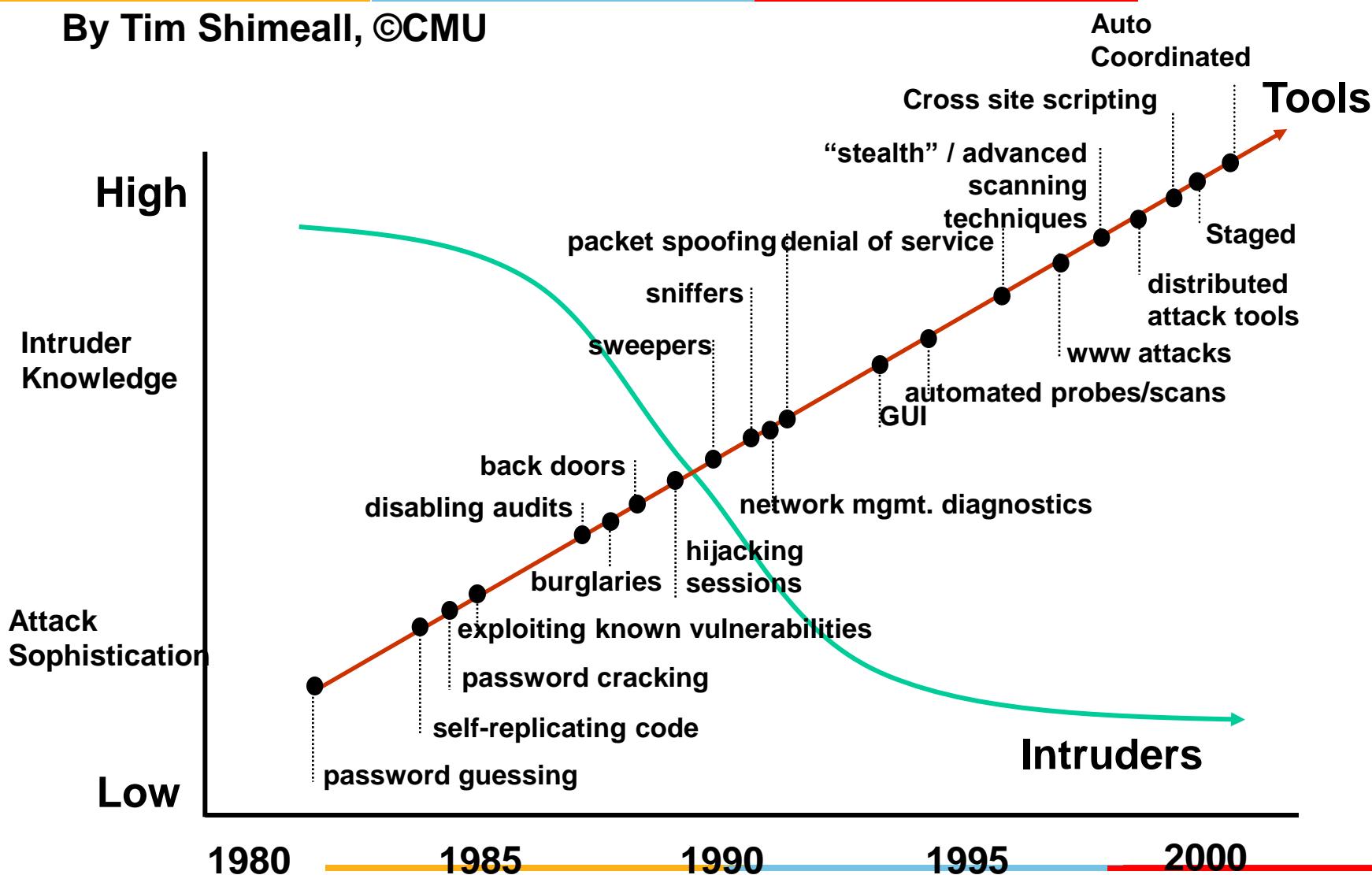
~ $7 \times 10^5$  malicious / bad apps are removed from Google Play Store in 2017, which is ~70% more than the apps taken down in 2016

Source T: <https://portal.av-atlas.org/malware/statistics>

Source B: <https://android-developers.googleblog.com/2018/01/how-we-fought-bad-apps-and-malicious.html>

# Security Trends

By Tim Shimeall, ©CMU



# CSI/FBI 2005 Computer Crime and Security Survey



---

Thanks!!!  
Queries?



# Network Security

## CSI ZG513 / ES ZG513 / SS ZG513

**BITS** Pilani  
K K Birla Goa Campus

Hemant Rathore  
Department of Computer Science and Information Systems

**Lecture Session – 2**

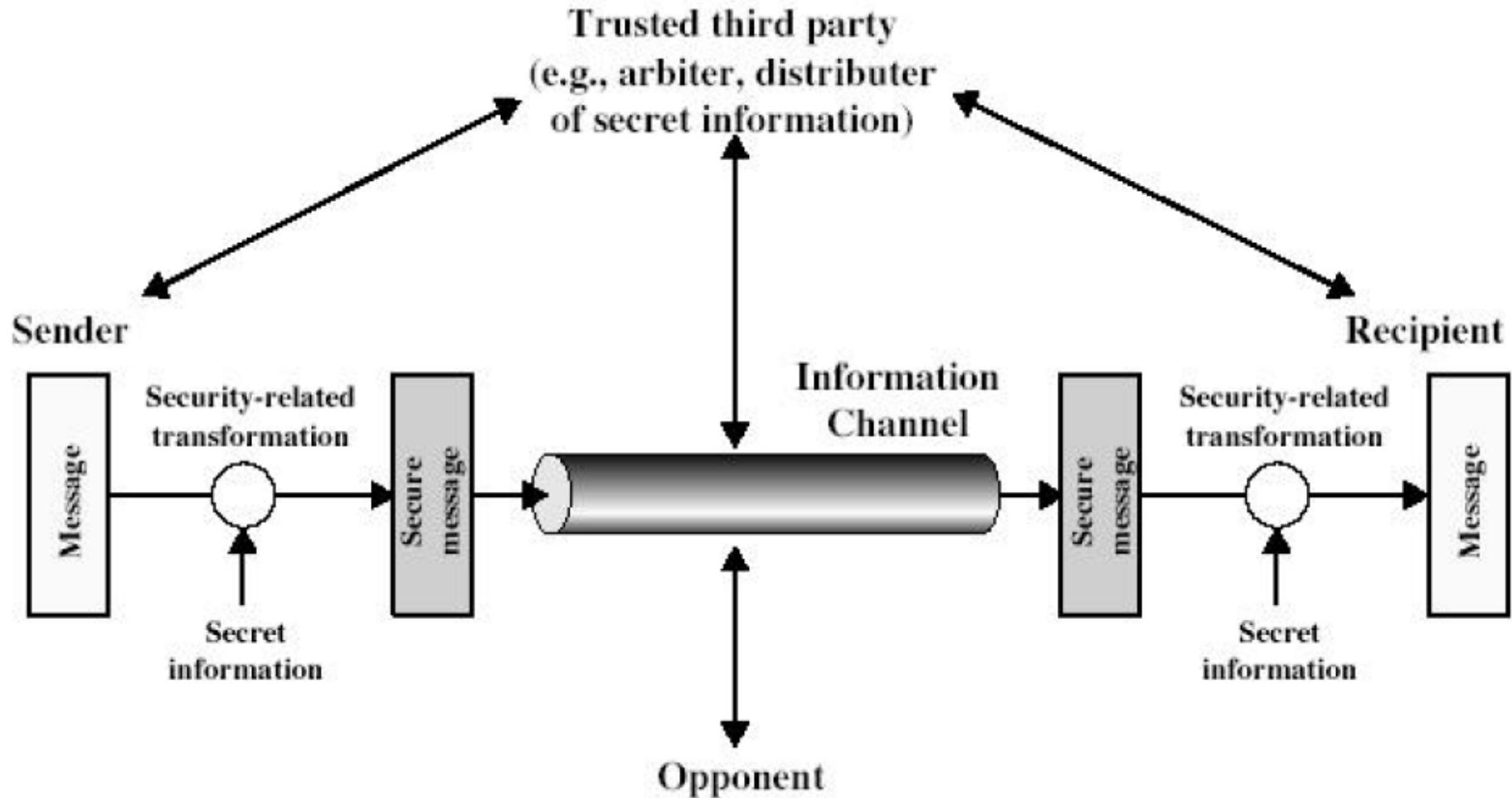
# Model for Network Security

---

- A likes to send a message to B securely.
- Insecure Channel.
- Secure Channel.



# Model for Network Security



# Cryptography: Terminology

---

- **Cryptography:** science of secret writing with the goal of hiding the meaning of a message.
  - **Cryptanalysis:** art and science to break the cryptosystem.
  - **Encryption:** method of transforming data ( $x$ ) into an unreadable format.
  - **Plaintext:** message/data before encryption.
  - **Ciphertext:** message/data after encryption.
  - **Decryption:** method to get back the ' $x$ ' from ' $y$ '.
-

# Cryptography: Terminology

---

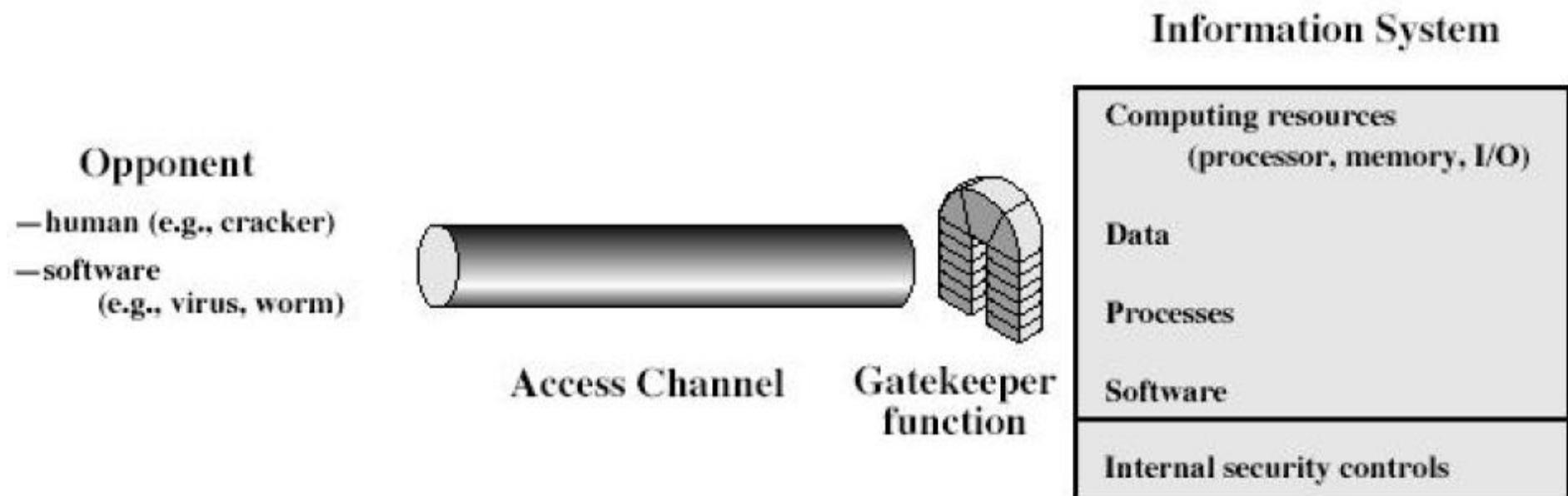
- **Cipher/EA:** set of rules/procedures that dictates how to encrypt/decrypt data.
  - **Key:** values used in encryption/decryption.
  - **Key space:** range of possible values used to construct keys.
  - **Key clustering:** when two different keys generate the same 'y' from the same 'x'.
  - **Work factor:** estimated time and resources to break a cryptosystem. **No system is unbreakable.**
-

# Model for Network Security

---

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

# Model for Network Security



# Model for Network Security

---

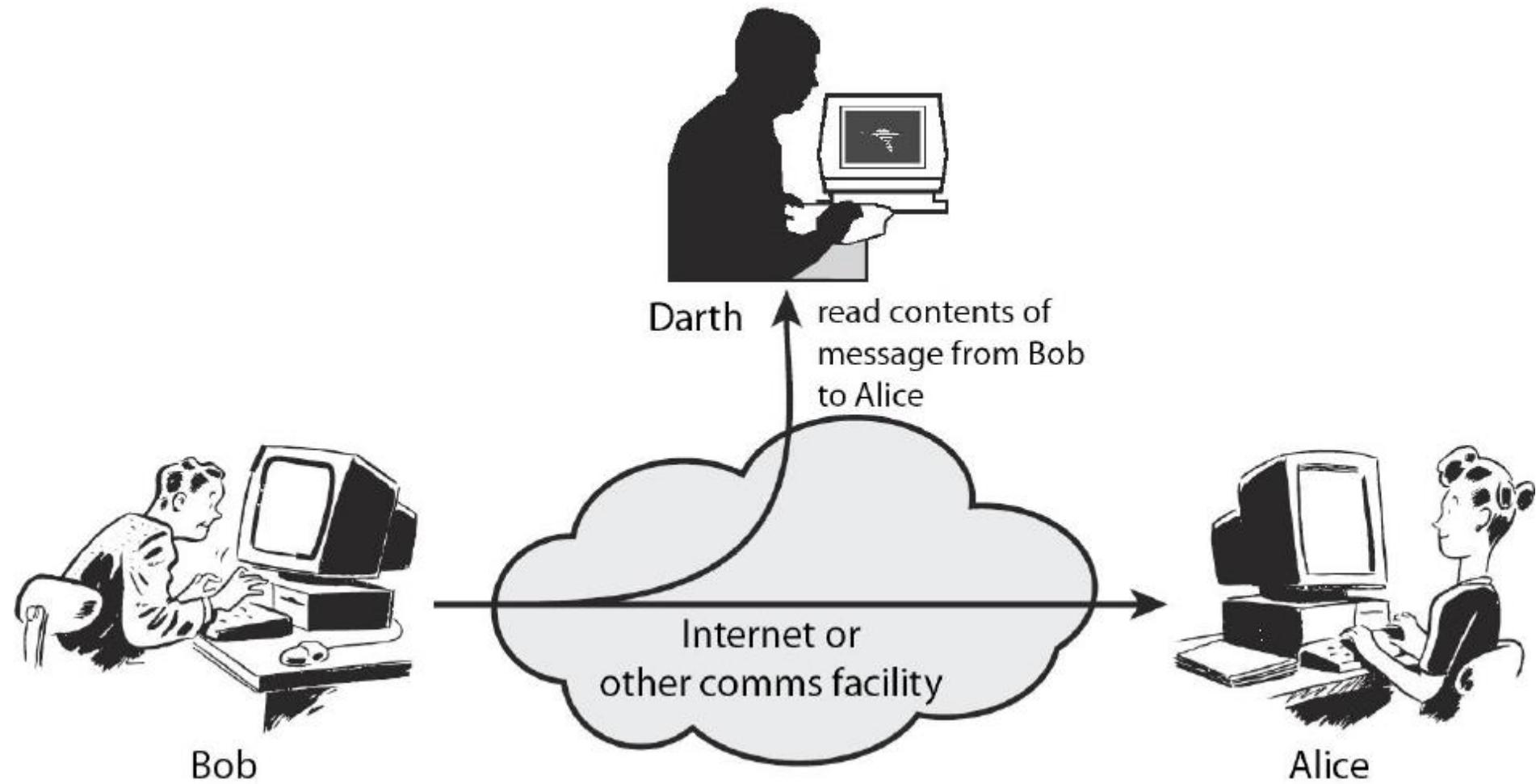
- Using this model requires us to:
    - Select appropriate gatekeeper functions to identify users
    - Implement security controls to ensure only authorised users access designated information or resources.
  - Trusted computer systems may be useful to help implement this model.
-

# Security Attacks

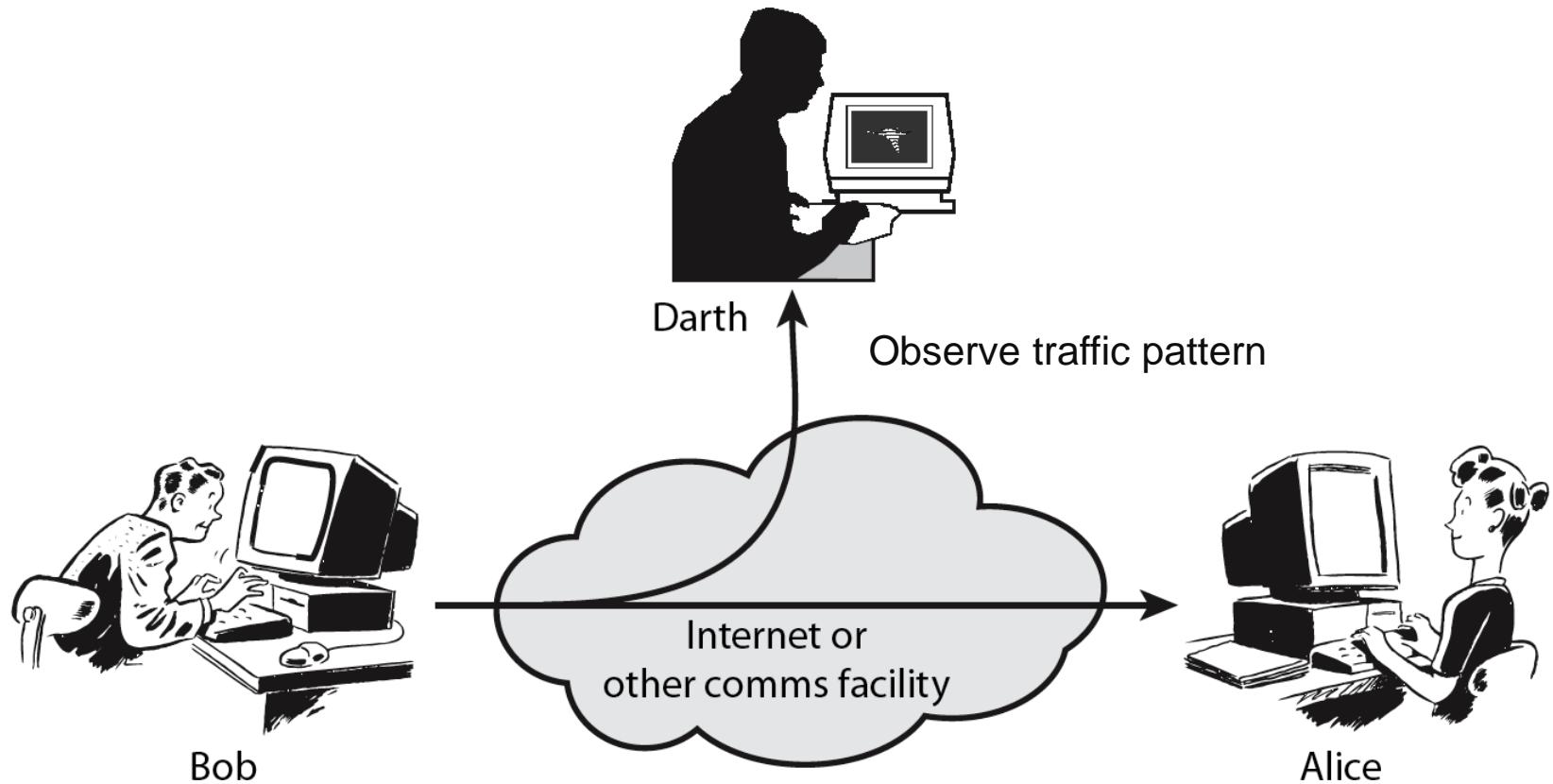
---

- Any action that compromises the security of information owned by an organization
- Information security is about how to prevent attacks, or failing that, to detect attacks on information based systems.
- Have a wide range of attacks.
- Focus on generic types of attacks
  - Passive
  - Active

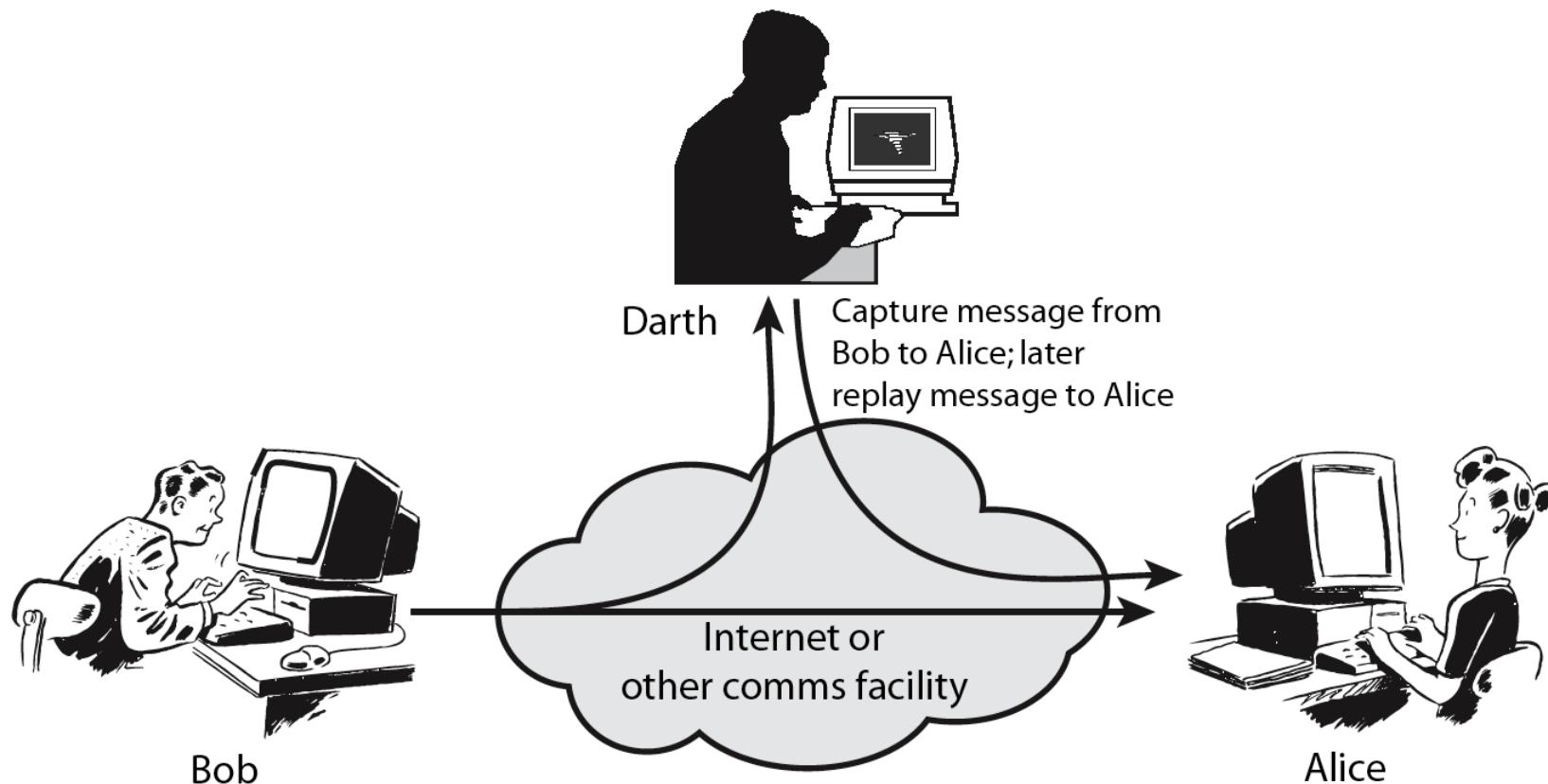
# Passive Attacks - Interception



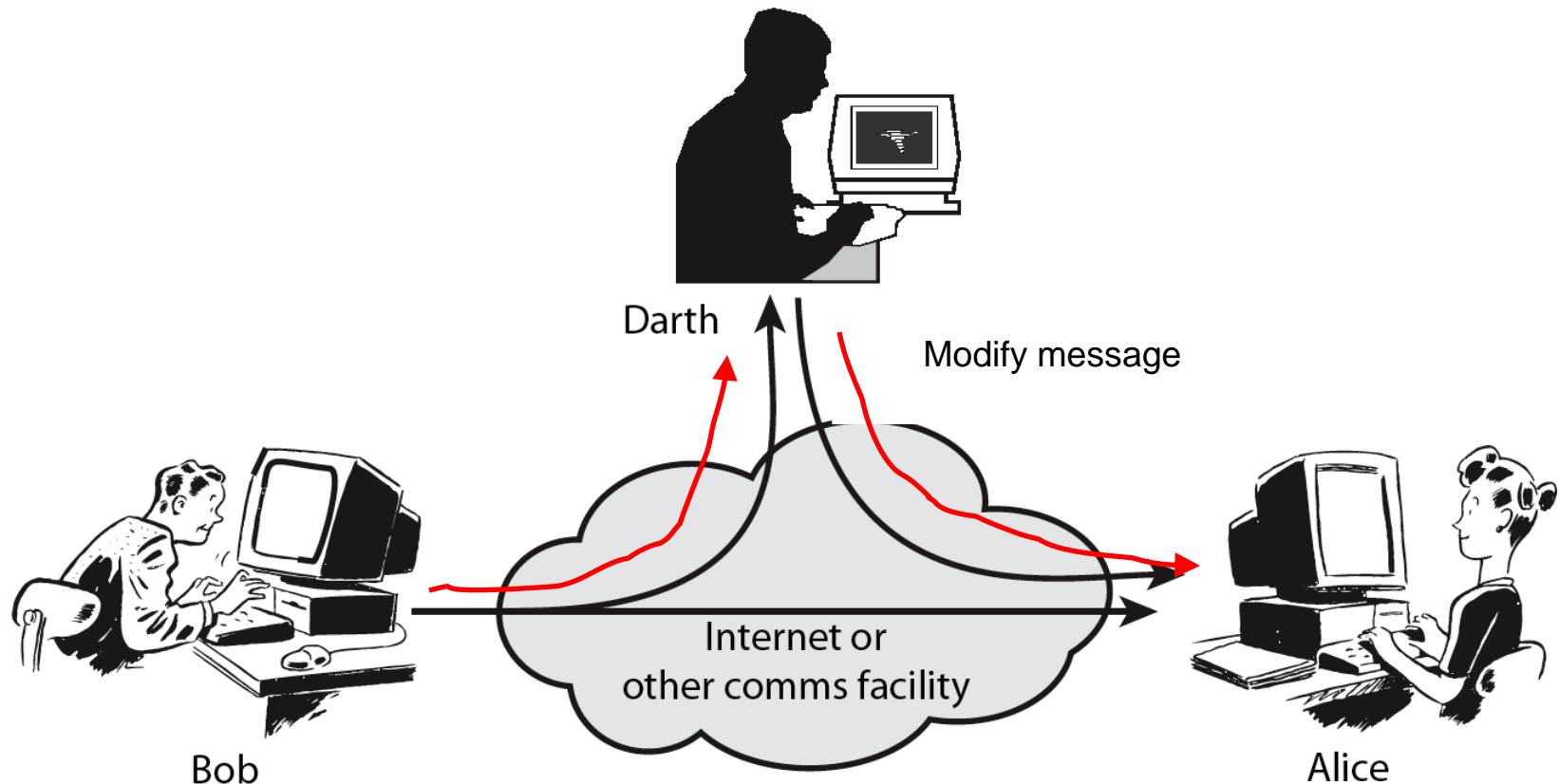
# Passive Attacks - Traffic Analysis



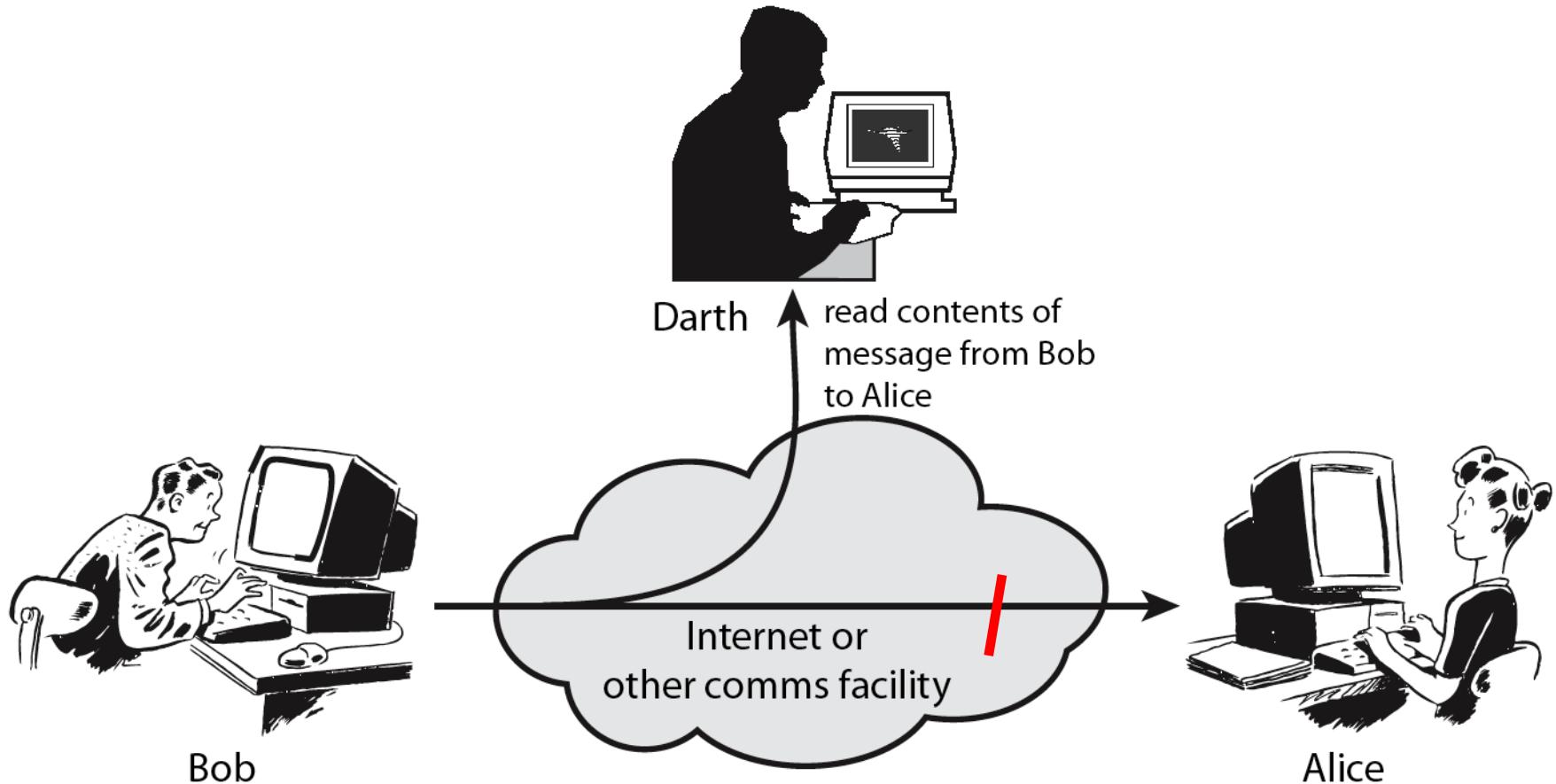
# Active Attacks - Reply



# Active Attacks - Modification



# Active Attacks - Interruption



# Handling Attacks

---

- Passive attacks – focus on Prevention
  - Easy to stop
  - Hard to detect
- Active attacks – focus on Detection and Recovery
  - Hard to stop
  - Easy to detect

# Security Policies

---

- Which is not expressly prohibited is permitted.
  - Which is not expressly permitted is prohibited.
  - What resource are you trying to protect.
  - Who is interested in attacking you.
  - How many security can you afford.
  - What action will be taken when security is compromised in the organization.
  - What action will be tolerated and what will not.
-

# Security Truism

---

- There is no such thing as absolute security.
  - Security is always a question of economics.
  - Keep the level of all your defenses at about the same height.
  - Attacker doesn't go through security, but around it.
  - Put your defenses in layers.
  - Its bad idea to rely on “Security through obscurity.”
-

# Security Truism

---

- If you don not run a program, it does not matter if it has security holes.
- A program/protocol is insecure until proven secure.
- A chain is only as strong as its weakest link.
- Security is a tradeoff with convenience.
- Don't underestimate the value of your assets.

---

Thanks!!!  
Queries?



# Network Security

## SS ZG513

**BITs** Pilani  
K K Birla Goa Campus

Hemant Rathore  
Department of Computer Science and Information Systems

# Last Class

---

- Course Objectives
- Books and Evaluation Scheme
- Course Content
- Introduction to Network Security
- Active and Passive Attacks

# Security Policies

---

- Which is not expressly prohibited is permitted.
  - Which is not expressly permitted is prohibited.
  - What resource are you trying to protect.
  - Who is interested in attacking you.
  - How many security can you afford.
  - What action will be taken when security is compromised in the organization.
  - What action will be tolerated and what will not.
-

# Security Truism

---

- There is no such thing as absolute security.
  - Security is always a question of economics.
  - Keep the level of all your defenses at about the same height.
  - Attacker doesn't go through security, but around it.
  - Put your defenses in layers.
  - Its bad idea to rely on “Security through obscurity.”
-

# Security Truism

---

- If you don not run a program, it does not matter if it has security holes.
- A program/protocol is insecure until proven secure.
- A chain is only as strong as its weakest link.
- Security is a tradeoff with convenience.
- Don't underestimate the value of your assets.

# Network Security

---

Network Security

Secure OS - Not covered

Secure N/W - Not covered  
Cryptology(Confidentiality)

Cryptography

Symmetric/Private Key

Asymmetric/Public Key Cryptography  
RSA Algo

Stream Ciphers (Bit by Bit)  
RC4

Block Ciphers  
AES(Advanced encryption standard)

Cryptanalysis -

Positive as well as Negative Impact =>  
Bad guy tries to break algo for exploiting vulnerability  
Can be used to find/investigating algo robustness by good guys

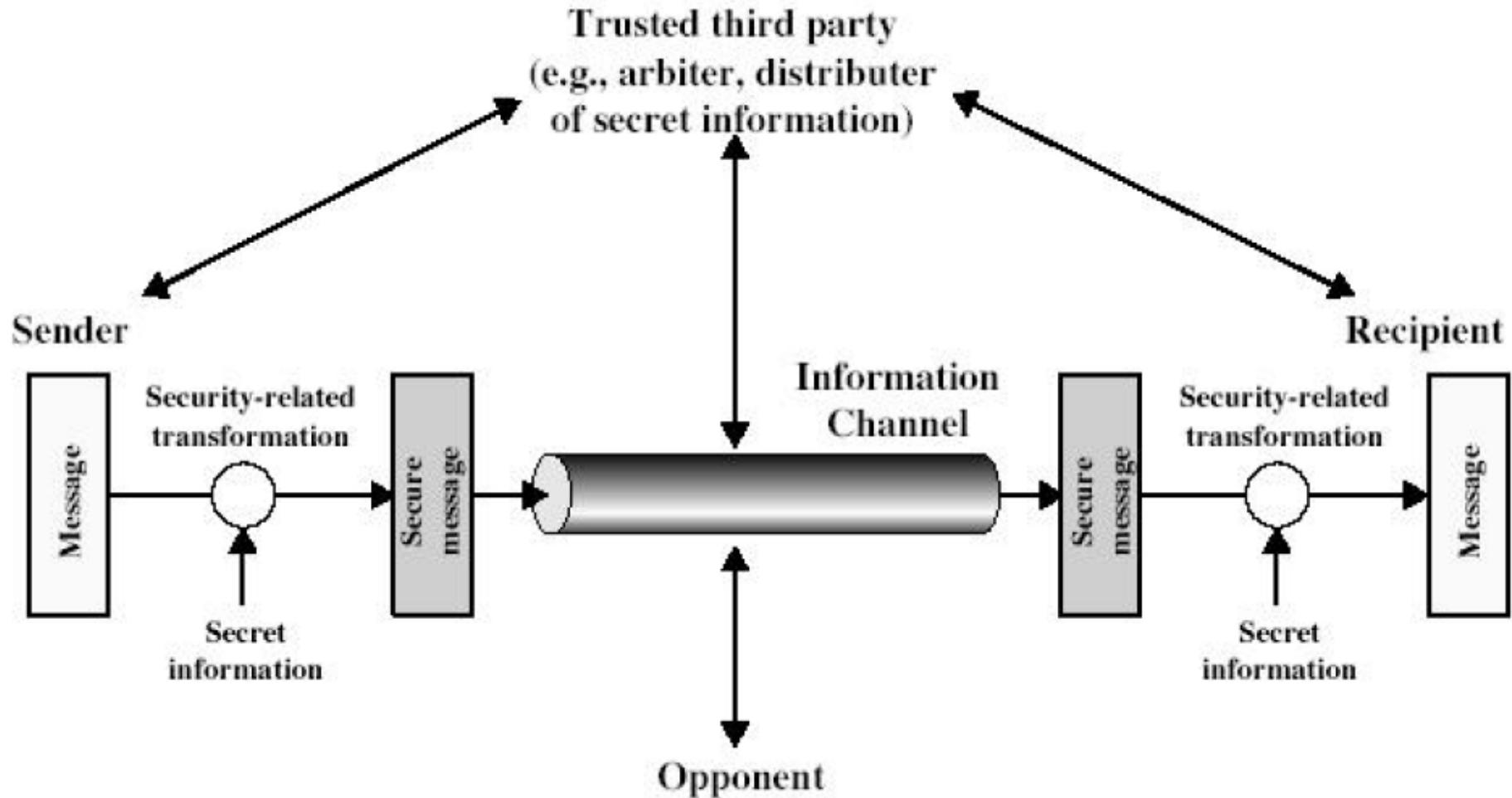
Protocols (Availability)  
Hash functions (Integrity) - SHA1/SHA2  
Digital Signatures

---

# Model (Communication)

---

# Model for Network Security



# Cryptography: Terminology

---

- **Cryptography:** science of secret writing with the goal of hiding the meaning of a message.
  - **Cryptanalysis:** art and science to break the cryptosystem.
  - **Encryption:** method of transforming data ( $x$ ) into an unreadable format.
  - **Plaintext:** message/data before encryption.
  - **Ciphertext:** message/data after encryption.
  - **Decryption:** method to get back the ' $x$ ' from ' $y$ '.
-

# Cryptography: Terminology

---

- **Cipher/EA:** set of rules/procedures that dictates how to encrypt/decrypt data.
  - **Key:** values used in encryption/decryption.
  - **Key space:** range of possible values used to construct keys.
  - **Key clustering:** when two different keys generate the same 'y' from the same 'x'.
  - **Work factor:** estimated time and resources to break a cryptosystem. **No system is unbreakable.**
-

# Kerckhoffs Principle (1883)

---

2000 BC to 1970 People thought Encryption and Decryption should be private

1880: A Cryptosystem should be secure if the attacker knows all the details except the secret key used to encrypt the message

# Substitution Cipher

---

Symmetric encryption - same key for encryption and decryption. Substitution cipher historical used 500 years back.  
Operates on letters. Today all ciphers Bits/Block not on letters.

A -> 1  
B -> 2  
C-> 3

...

Z-> 26

Suppose message is X (AACB) -> Y(1132) Letters sent as numbers  
Other end decryption numbers to letters Y(1132) -> X(AACB)

1-> A  
2->B  
3->C

26-> Z

This table acts as key

# Substitution Cipher

---

Plaintext		Ciphertext
A	->	q
B	->	w
C	->	e

Eg: ABC would be encrypted as qwe

- Ciphertext:
- qwertyuiop
- How secure is the Substitution Cipher ?

Brute force attack - Try all possible keys to get plaintext from cipher text. Number of possible combinations =  $26 * 25 * 24 \dots 1 = 26!$

e.g. A mapped to a then B can be /c/d ...25 other combo, C can be mapped to 24 other letters, D can be 23 other so  $26!$  possible keys =  $2^{88}$

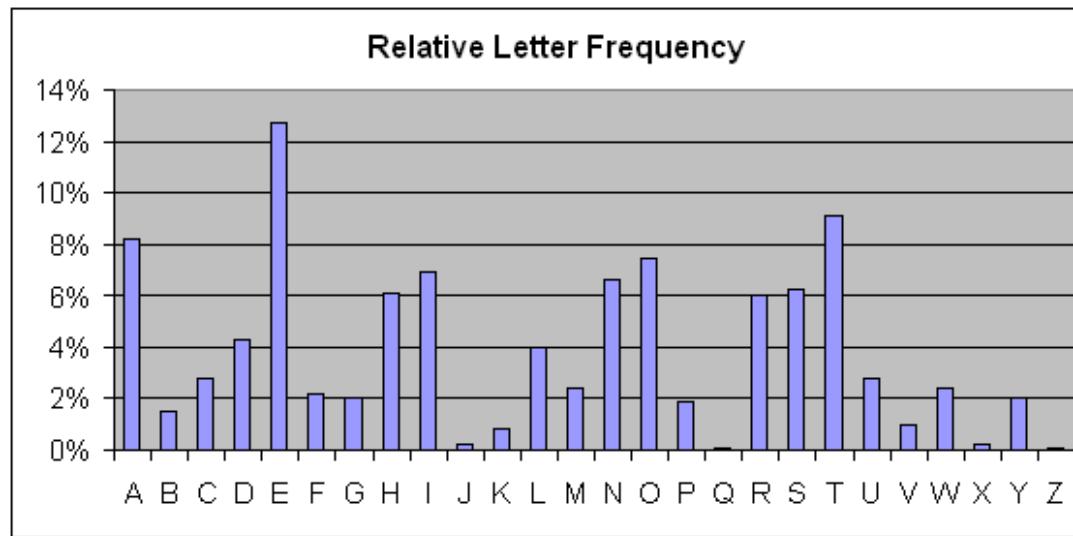
# Substitution Cipher

---

- Brute Force Attack:
  - Try every possible key
  - How many keys are there
$$26 * 25 * 24 * 23 \dots \dots 1 = 26! = 2^{88}$$
  - Not possible to solve by todays computer
  - Is Substitution Cipher unbreakable ?

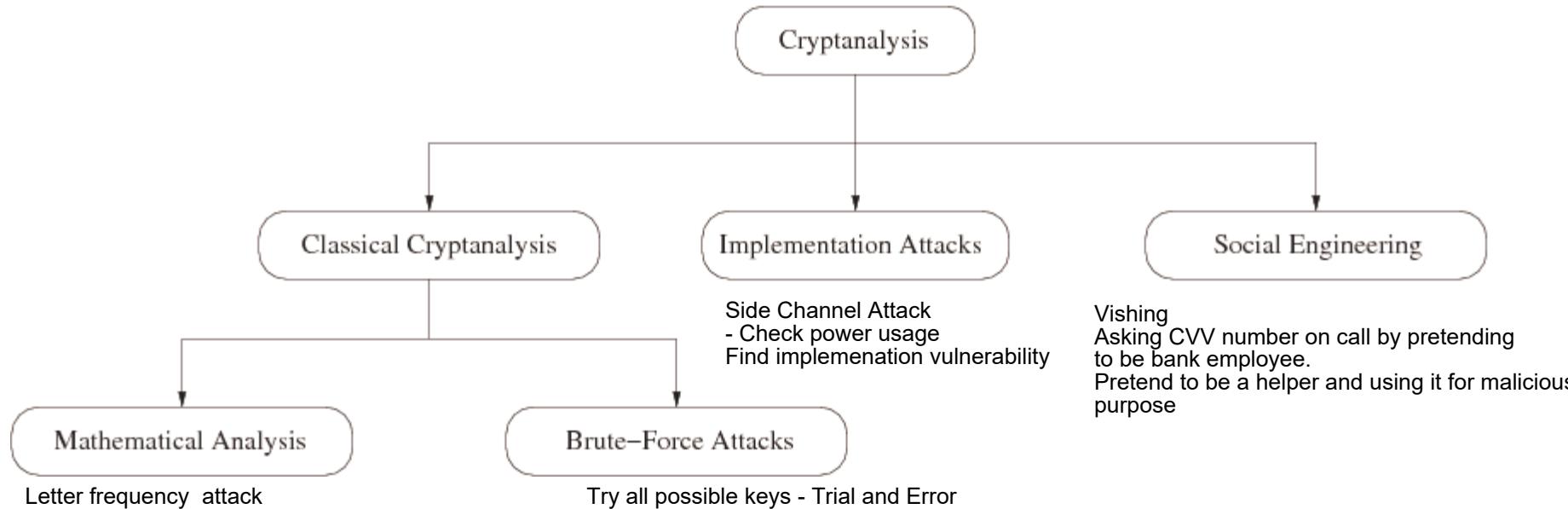
# Substitution Cipher

- Letter frequency analysis



- Count the frequency of letters in Cipher text and replace it with help of letter frequency analysis.

# Cryptanalysis



# Cryptanalysis

---

---

Thanks!!!  
Queries?



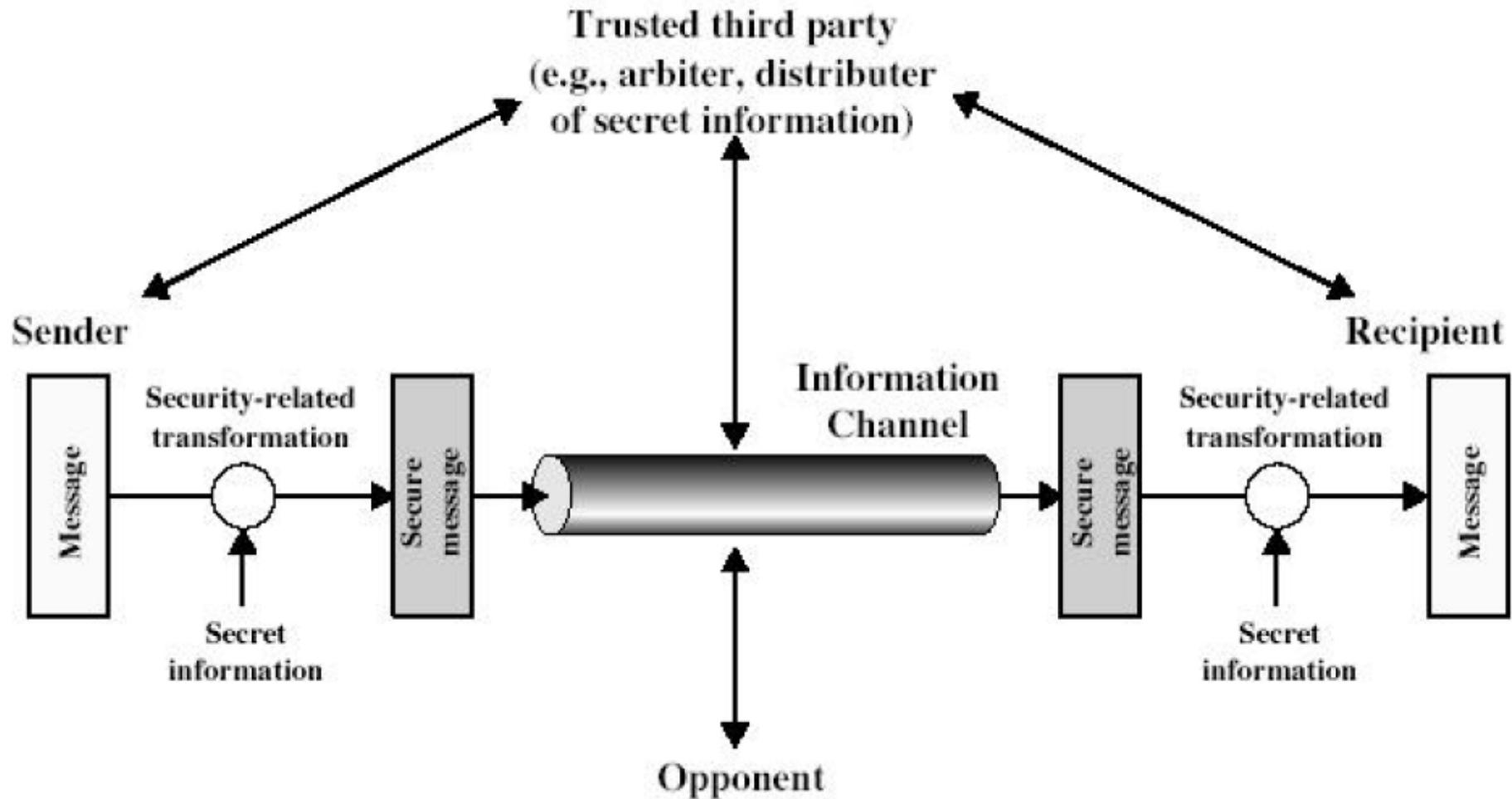
**BITS** Pilani  
K K Birla Goa Campus

# Network Security

## SS ZG513

Hemant Rathore  
Department of Computer Science and Information Systems

# Model for Network Security



# NS Goals

---

- **Data Confidentiality** - protection of data from unauthorized disclosure.
  - **Authentication** - assurance that the communicating entity is the one claimed.
  - **Integrity** - assurance that data received is as sent by an authorized entity.
  - **Non-Repudiation** - protection against denial by one of the parties in a communication.
  - **Access Control** - prevention of the unauthorized use of a resource.
-

# Modular Arithmetic

---

Most Cryptosystems are based on **FINITE** sets:

- Finite set:
  - Discrete (Sets with integers)
  - Finite (Cardinality of set)

# Modular Arithmetic

---

- Modulus Operation

Let  $a, r, m$  be integers where  $m > 0$ . Then

$$a \equiv r \pmod{m}$$

if  $(r-a)$  is divisible by  $m$

“ $m$ ” is called the modulus

“ $r$ ” is called the remainder

# Modular Arithmetic

---

## Computation of Remainder

Given  $a, m \in \mathbb{Z}$

$$a = q * m + r$$

where      $q$  = quotient

$r$  = remainder

$m$  = modulus operation

**remainder is not unique**

# Computation of the Remainder

---



# Modular Arithmetic

---

Equivalence Class:

where all the members of a set has equivalence relation.







# Ring in Modular Arithmetic

---

Definition:

- The Integer Ring,  $Z_m$  consist of
  - The set  $Z_m = \{ 0, 1, \dots, m-1 \}$
  - Two operations
    - “+” and “\*”  
for all  $a, b, \dots \in Z_m$
    - eg  $a + b = c \text{ mod } m$
    - $a * b = d \text{ mod } m$

# Integer Ring

---

- Closure: add and multiply of any two numbers and the result is always in the ring.
- Associative: Addition and Multiplication operations are associative, for all  $a, b, \dots \in Z_m$ 
$$a + (b + c) = (a + b) + c$$
$$a * (b * c) = (a * b) * c$$
- Distributive: for all  $a, b, \dots \in Z_m$ 
$$a * (b + c) = (a * b) + (a * c)$$

# Integer Ring

---

- Neutral Element: 0 with respect to addition, i.e., for all  $a \in Z_m$

$$a + 0 \equiv a \text{ mod } m$$

- Additive inverse: For  $a \in Z_m$ , there is always an additive inverse element  $-a$  such that

$$a + (-a) \equiv 0 \text{ mod } m$$

# Integer Ring

---

- Neutral Element: 1 with respect to multiplication,  
i.e., for all  $a \in \mathbb{Z}_m$

$$a * 1 \equiv a \pmod{m}$$

- Multiplicative inverse ( $a^{-1}$ )

$$a * a^{-1} \equiv 1 \pmod{m}$$

exists only for some, **but not for all**, elements in  $\mathbb{Z}_m$ .

# Modular Arithmetic: Applications



## Shift/Caesar Cipher

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

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

$$y = E_k(x) \equiv (x + k) \bmod 26$$
$$x = D_k(y) \equiv (y - k) \bmod 26$$

# Shift/Caesar Cipher

---

- If  $k = 10$  and plaintext is CRYPTO =  $x_1, x_2, x_3, x_4, x_5, x_6 = 2, 17, 24, 15, 19, 14$  then ciphertext =  $y_1, y_2, y_3, y_4, y_5, y_6 = 12, 1, 8, 25, 3, 24 = MBIZDY$
- Only 25 possible keys, hence brute force attack is trivial. Also one can apply letter frequency analysis.
- If arbitrary substitution, then key space is 26!

# Modular Arithmetic: Applications

## Affine cipher:

- If  $x, y, a, b \in \mathbb{Z}_{26}$ , then

$$y = E_k(x) \equiv (a \cdot x + b) \pmod{26}$$

$$x = D_k(y) \equiv a^{-1} \cdot (y - b) \pmod{26}$$

- If  $(a, b) = (3, 10)$  and plaintext is CRYPTO =  $x_1, x_2, x_3, x_4, x_5, x_6 = 2, 17, 24, 15, 19, 14$  then ciphertext =  $y_1, y_2, y_3, y_4, y_5, y_6 = 16, 9, 4, 3, 15, 0$  = QJEDPA
- $12 \times 26 = 312$  possible keys. Larger than caesar cipher but still brute force attack is trivial and letter frequency analysis.
- Correctness.

---

Thanks!!!  
Queries?



**BITS** Pilani  
K K Birla Goa Campus

# Network Security

## SS ZG513

Hemant Rathore  
Department of Computer Science and Information Systems



# Previous Lecture

---

- Modular Arithmetic, Integer Ring, Group.
- Substitution and Transposition Cipher.

# Modular Arithmetic: Applications

## Shift/Caesar cipher:

- If  $x, y, k \in \mathbb{Z}_{26}$ , then

$$y = E_k(x) \equiv (x + k) \pmod{26}$$
$$x = D_k(y) \equiv (y - k) \pmod{26}$$

- If  $k = 10$  and plaintext is CRYPTO =  $x_1, x_2, x_3, x_4, x_5, x_6 = 2, 17, 24, 15, 19, 14$  then ciphertext =  $y_1, y_2, y_3, y_4, y_5, y_6 = 12, 1, 8, 25, 3, 24$  = MBIZDY
- Only 25 possible keys, hence brute force attack is trivial. Also one can apply letter frequency analysis.
- If arbitrary substitution, then key space is 26!

# Modular Arithmetic: Applications

## Affine cipher: ✓

- If  $x, y, a, b \in \mathbb{Z}_{26}$ , then

$$y = E_k(x) \equiv (ax + b) \pmod{26}$$

$$x = D_k(y) \equiv a^{-1} \cdot (y - b) \pmod{26}$$

- If  $(a, b) = (3, 10)$  and plaintext is CRYPTO =  $x_1, x_2, x_3, x_4, x_5, x_6 = 2, 17, 24, 15, 19, 14$  then ciphertext =  $y_1, y_2, y_3, y_4, y_5, y_6 = 16, 9, 4, 3, 15, 0 = QJEDPA$
- $12 \times 26 = 312$  possible keys. Larger than caesar cipher but still brute force attack is trivial and letter frequency analysis.
- Correctness.



# Breaking an Algorithm

- **Total break:** the attacker deduces the secret key.
- **Global deduction:** the attacker discovers a functionally equivalent algorithm for encryption and decryption, but without learning the key.
- **Instance (local) deduction:** the attacker discovers additional plaintexts (or ciphertexts) not previously known.
- **Information deduction:** the attacker gains some Shannon information about plaintexts (or ciphertexts) not previously known.

Pilani  
KKB

# Security of Cipher

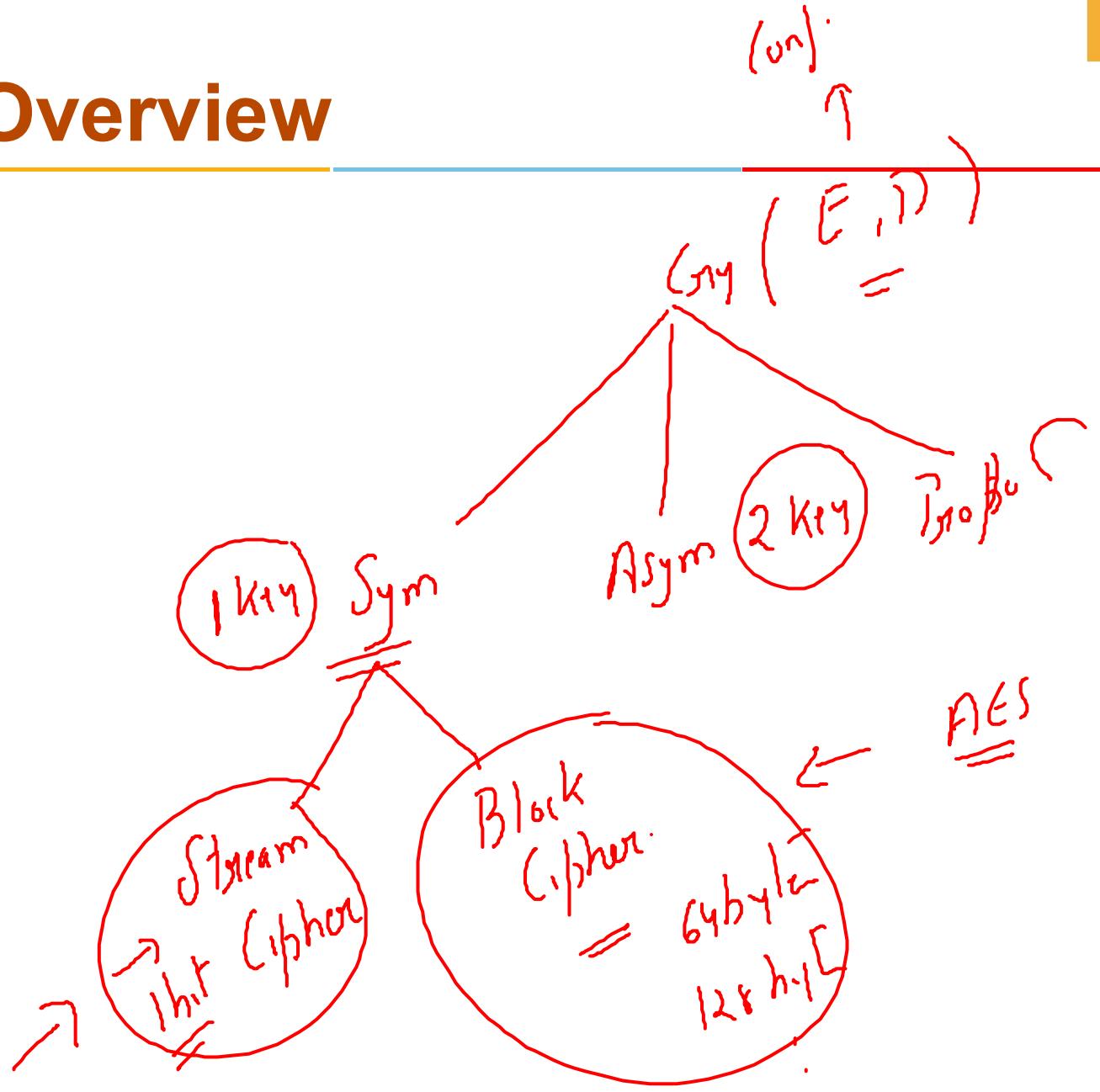
- Unconditional secure ✓
- Computationally secure ✓
- Degree of security: how hard to break.
- Peer-review.
- Decoding by reverse engg.
- Data Complexity: Breaking cost  $\gg$  Encrypted data cost.
- Time Complexity: Time require to break  $\gg$  Time the data is useful.
- Storage requirement: Amount of data required to break  $\gg$  Amount of available 'x', 'y'.
  - $2^{128}$
- An algorithm is said to have a **security level of  $n$  bit** if the best known attack requires  $2^n$  steps.



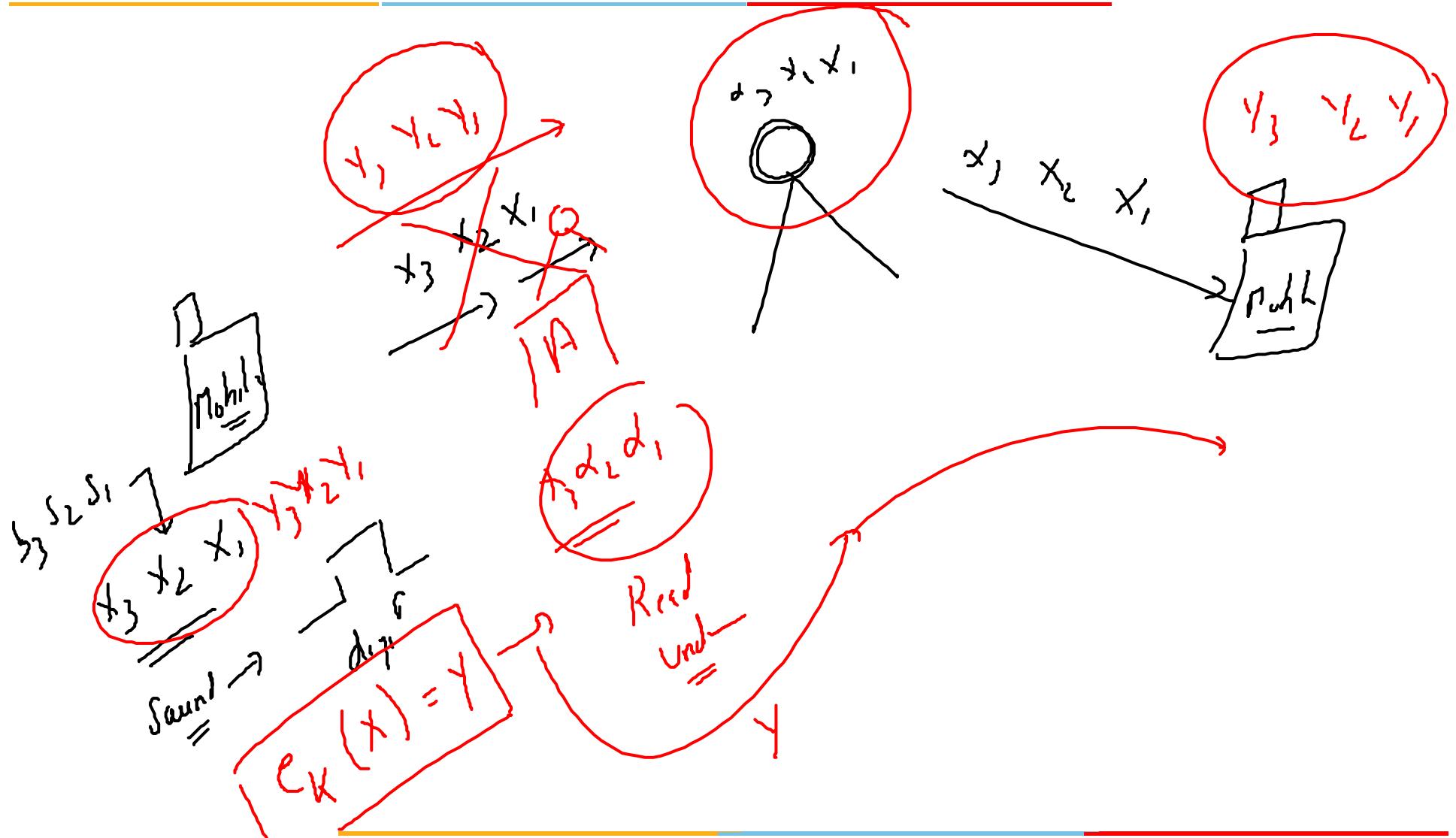
# Today's Agenda

- Stream Cipher ✓
- Random Number ✓
- TRNG, PRNG, CSPRNG ✓
- OTP ✓
- RC4 ✓
- Slides:
  - Cryptography and Network Security by William Stallings.

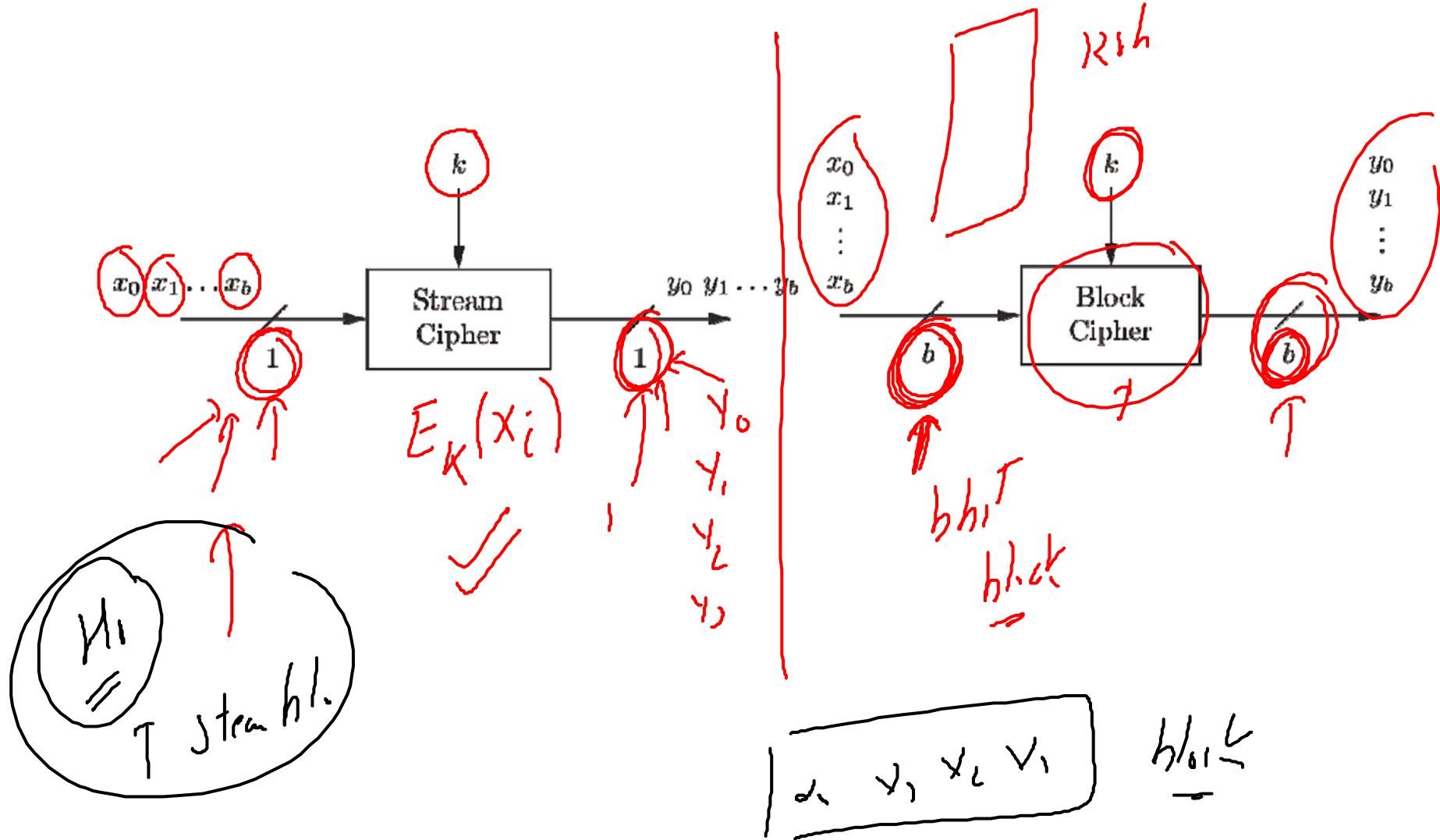
# Overview



# Stream Cipher (Example)



# Stream Cipher vs Block Cipher



# Stream Cipher vs Block Cipher

## Stream Ciphers

- Encrypt bits individually ✓✓
- Usually small and fast common in embedded devices (e.g. A5/1 for GSM phones)



## Block Ciphers:

- Always encrypt a full block (several bits)
- Are common for Internet applications



$\alpha_i, \gamma_i \in \{0, 1\}$



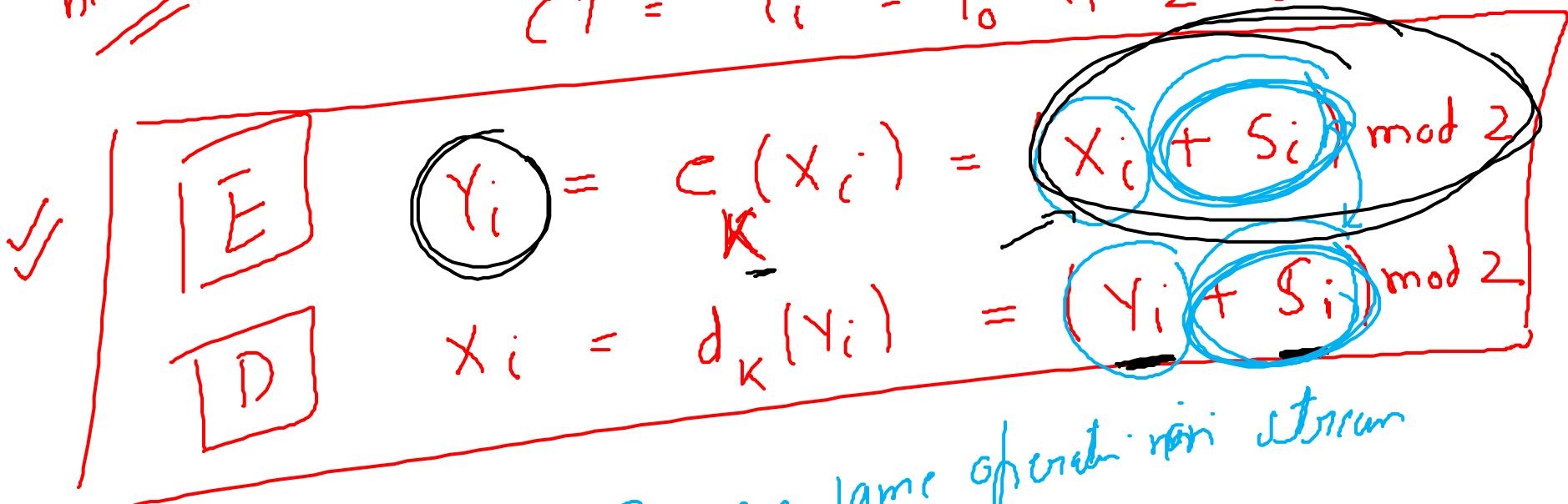
## Stream Cipher (Definition)

one bit  
at a time  
bit initial

$$PT = x_i = \alpha_0, x_1, x_2, \dots, x_n$$

$$S_i = s_0, s_1, s_2, s_3$$

$$CT = y_i = y_0, y_1, y_2, y_3, \dots$$



E&P are same ciphering stream  
cipher

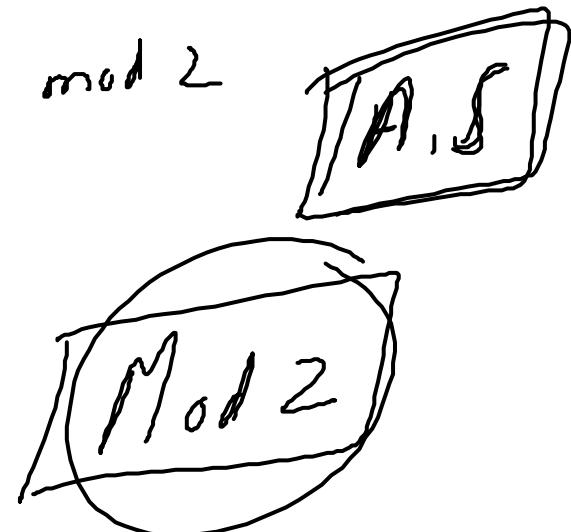
# Why E and D are same operation?

$$d(y_i) = (\underline{y_i} + s_i) \bmod 2$$

$$= (\underline{x_i} + \underline{s_i}) \bmod 2 + \underline{s_i} \bmod 2$$

$$= x_i + 2s_i \bmod 2$$

$d(y_i) = \boxed{x_i} \bmod 2$

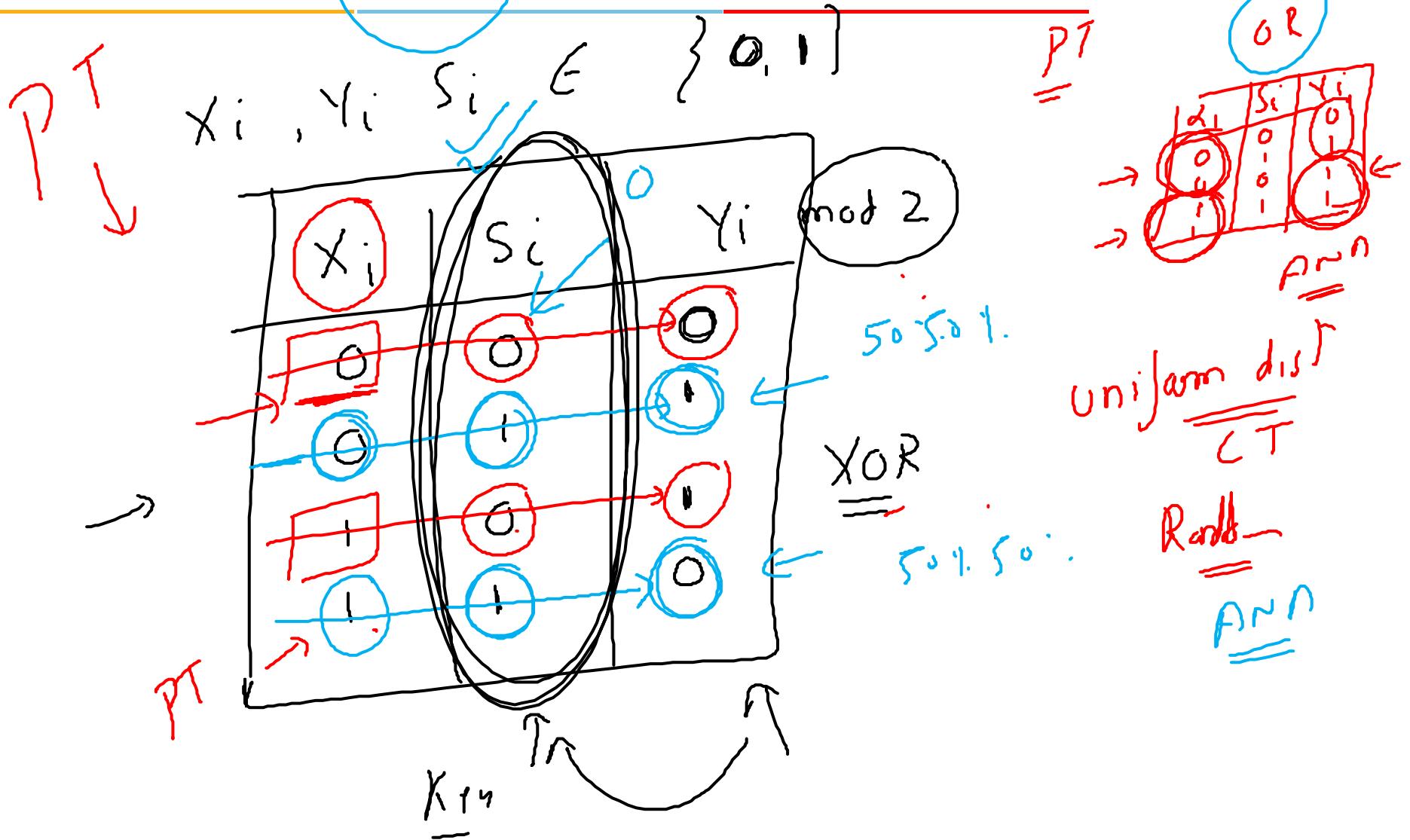


$$d(y_i) = \checkmark (y_i - s_i) \bmod 2$$

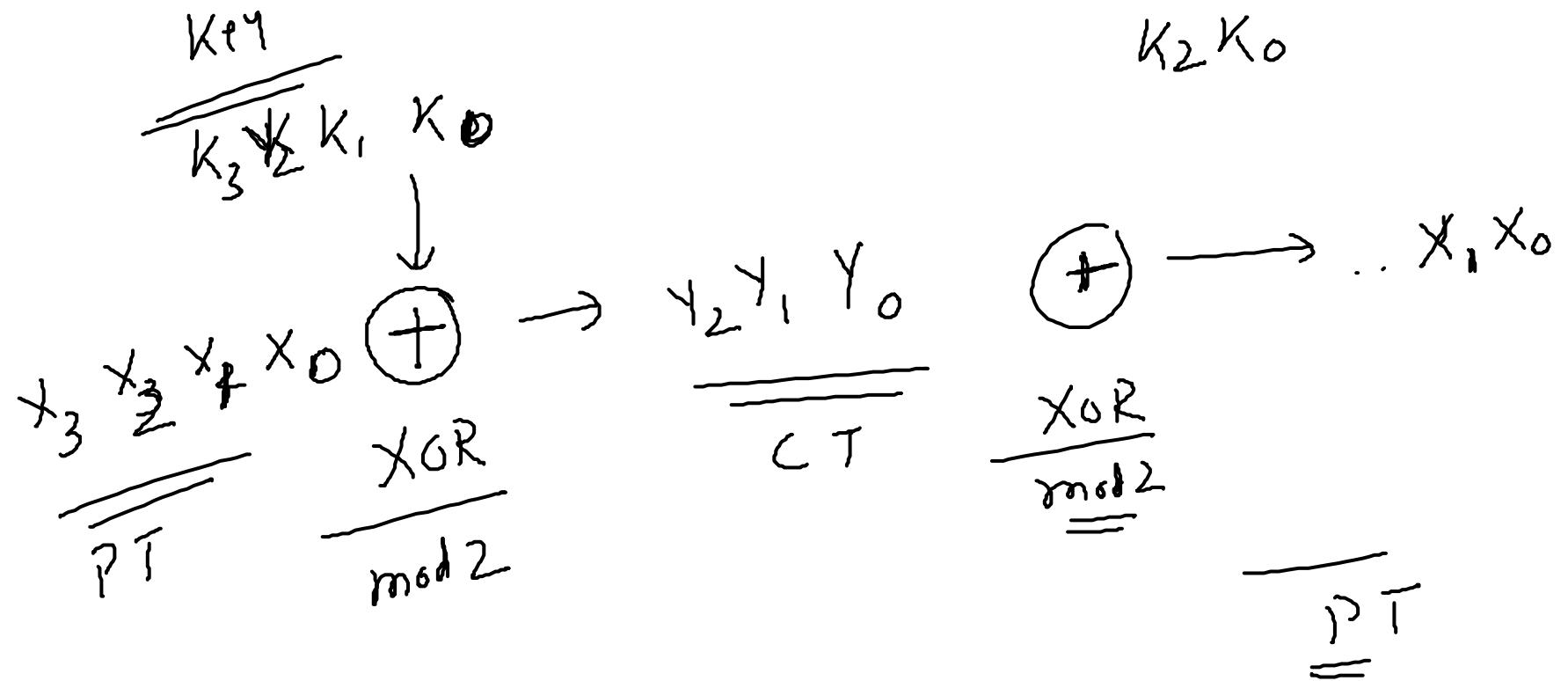
$$= \cancel{x_i} (x_i + s_i - s_i) \bmod 2$$

~~$x_i \bmod 2$~~

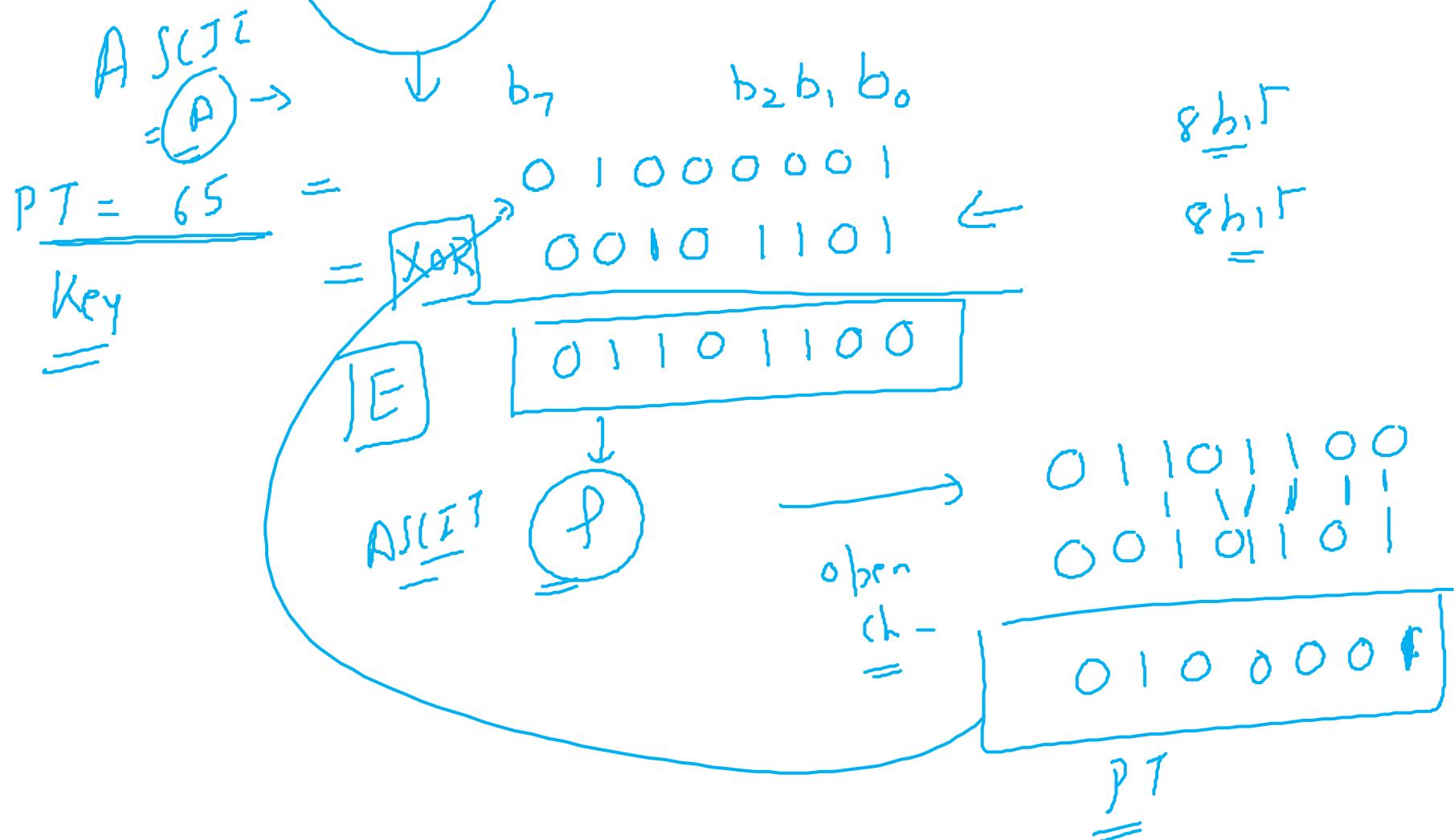
# Mod2 vs XOR



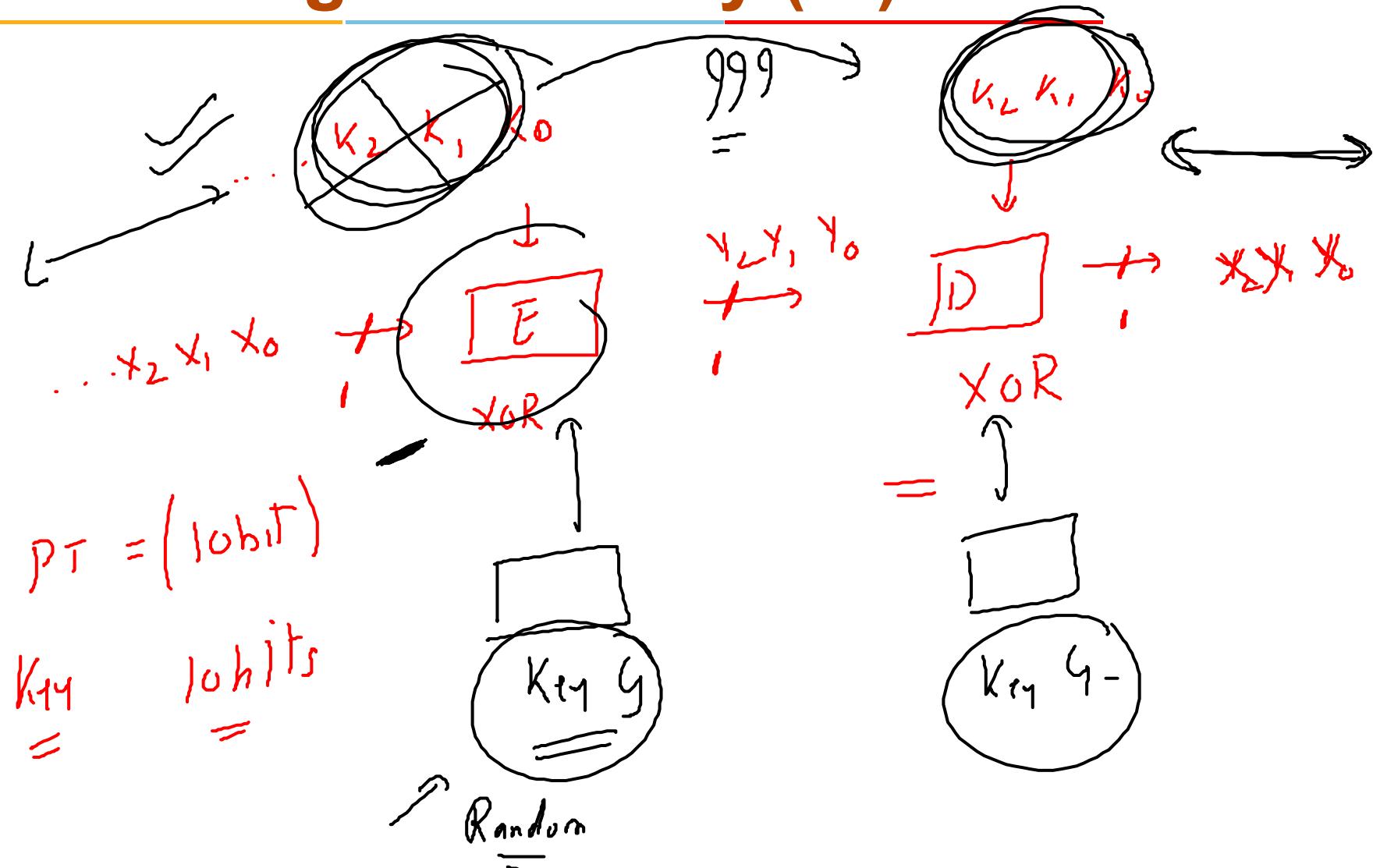
# Stream Cipher (Example Diagram)



# Encrypt "A"



# How do generate key (Si)



# Random Numbers

- Random Numbers in cryptography
  - nonces in authentication protocols to prevent replay
  - session keys
  - public key generation
  - keystream for a one-time pad
- Properties of Random Number
  - statistically random
  - uniform distribution
  - independent
  - unpredictability of future values from previous values



# True Random Number Generator (TRNG)



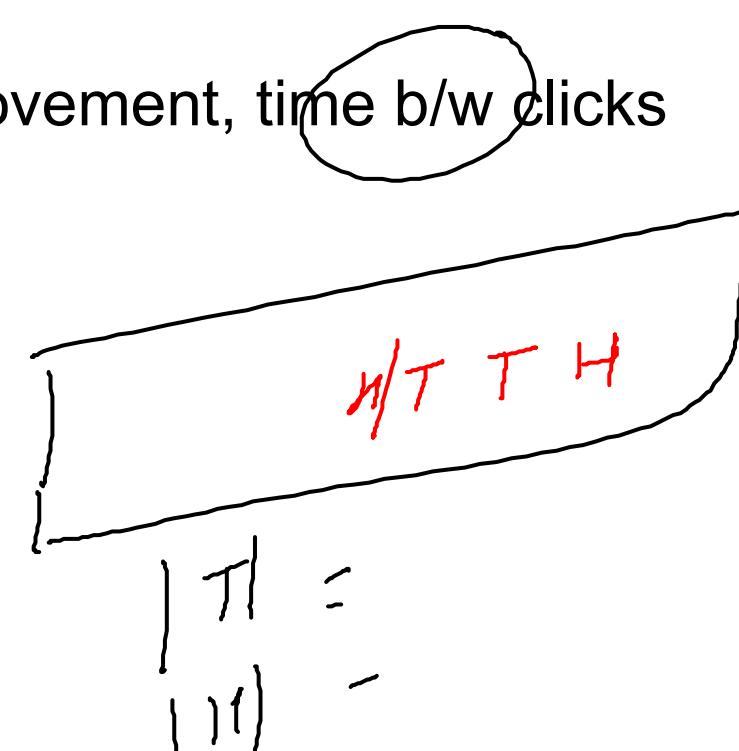
- Generate "TRUE RANDOM NUMBER"
- True Random Number stem from random physical process
- eg: coin flipping, mouse movement, time b/w clicks
- +ve:
  - True Random Number
- -ve
  - Cannot recreate them

(1) SR

(2) V P

(3) g n

↓  
F



# Pseudo Random Number Generators (PRNG)



- Use deterministic algorithmic techniques to create “random numbers”
- can be recreated
- although are not truly random
- can pass many tests of “randomness”
- known as “pseudorandom numbers”
- created by “Pseudorandom Number Generators (PRNGs)”

# Pseudorandom Number Generators



- PNG computation:

$$\left. \begin{array}{l} s_0 = \text{seed} \\ s_{i+1} = f(s_i, s_{i-1}, \dots, s_{i-t}) \end{array} \right\}$$

- ~~rand() function in ANSI C~~

$$s_0 = 12345$$

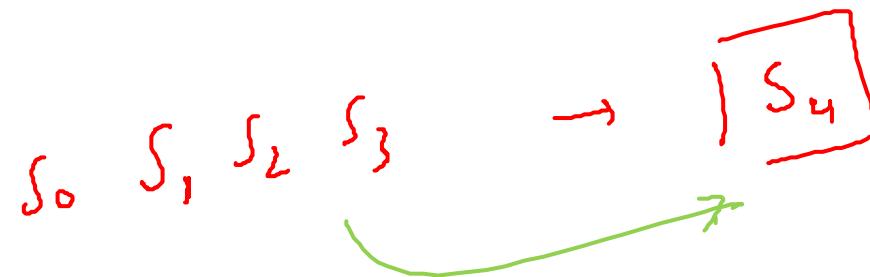
$$s_{i+1} = 1103515245s_i + 12345 \bmod 2^{31}$$

- Most PRNGs have bad cryptographic properties

$$s_{i+1} = As_i + B \bmod 2^m$$

# Cryptographically Secure PRNG

- Given n consecutive bits of output  $S_n$ , the following output bits  $s_{n+1}$  cannot be predicted



# One Time Pad



$\text{TRNG}$

- A cryptosystem is unconditionally secure if it cannot be broken even with infinite computational resources.

- OTP in stream cipher:
  - The key steam  $S_i$  from TRNG
  - Each key stream is used only once
- Eg: Let plaintext, ciphertext and key

$$x_i, y_i, k_i \in \{0, 1\}$$

$$\text{Encryption: } e_k(x) = x \oplus k$$

$$\text{Decryption: } d_k(y) = y \oplus k$$

- OTP is unconditionally secure if key  $k_i$  is used once

$\text{TRNG}$

# One Time Pad

---

- Unconditionally secure cryptosystem:

$$y_0 = x_0 \oplus k_0$$

$$y_1 = x_1 \oplus k_1$$

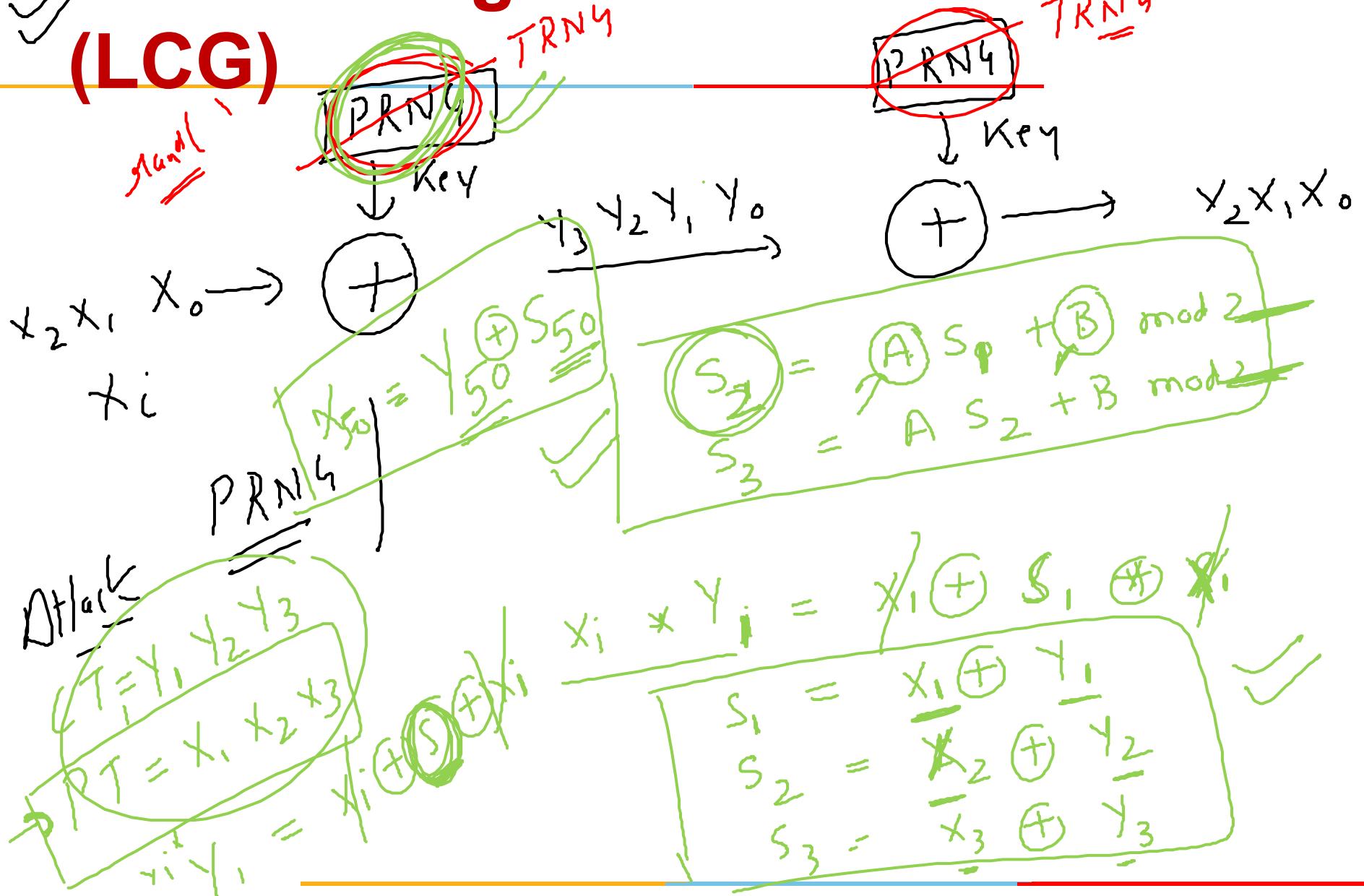
- Every equation is a linear equation with two unknowns
  - For every  $y_i$  are  $x_i = 0$  and  $x_i = 1$  equi-probable
  - Since  $k_0, k_1, \dots$  are independent, thus  $k_i$  will be Truly Random number
  - It can proved that this systems can not be solved.



# One Time Pad (Limitation)

---

# Linear Congruential Generator (LCG)

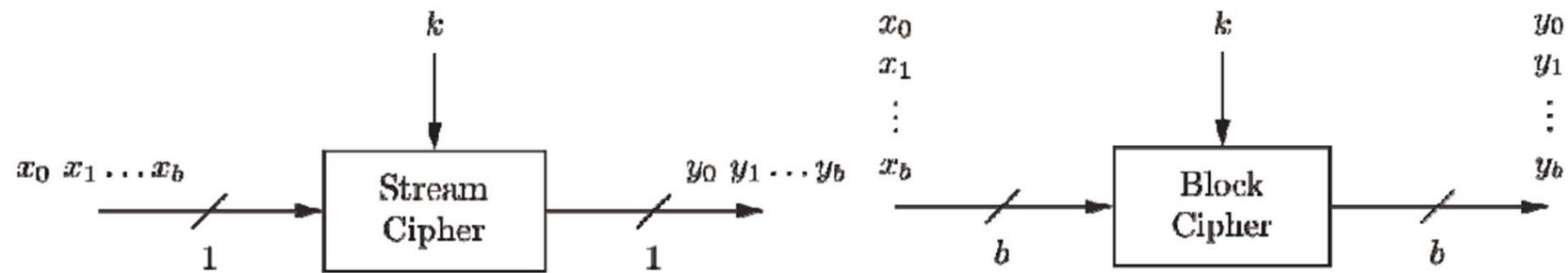


# Linear Congruential Generator (LCG)

---



# Stream Cipher vs Block Cipher

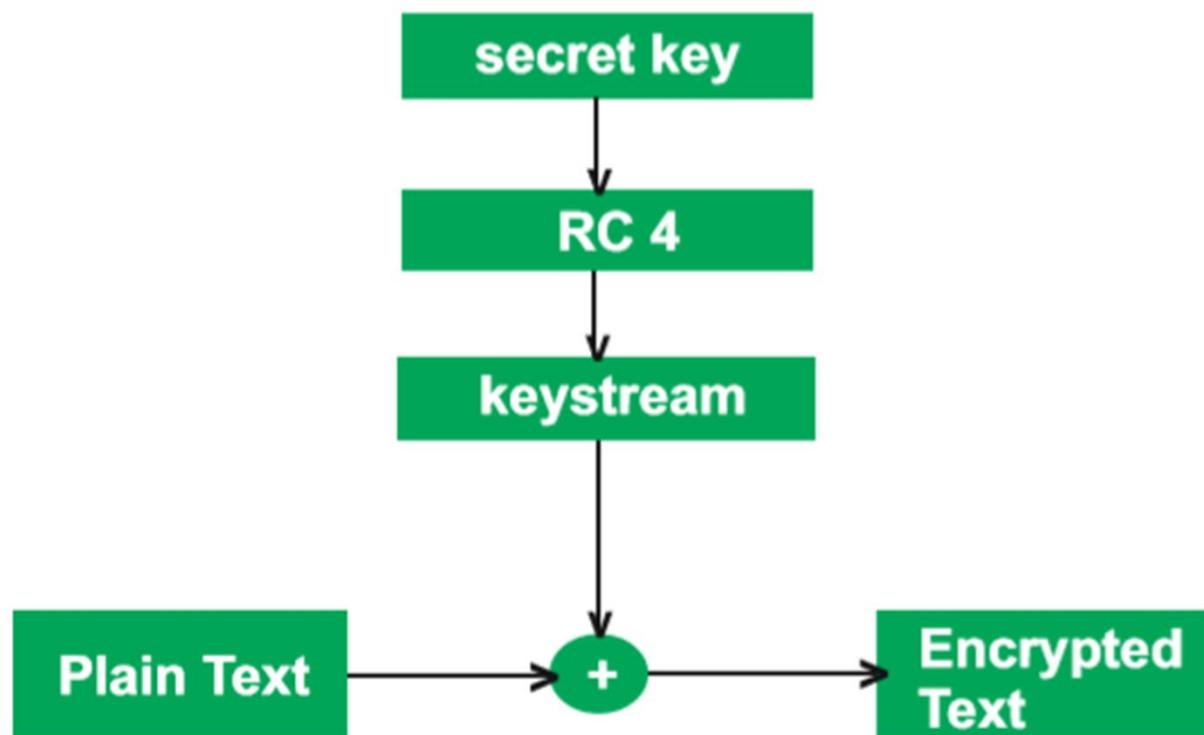




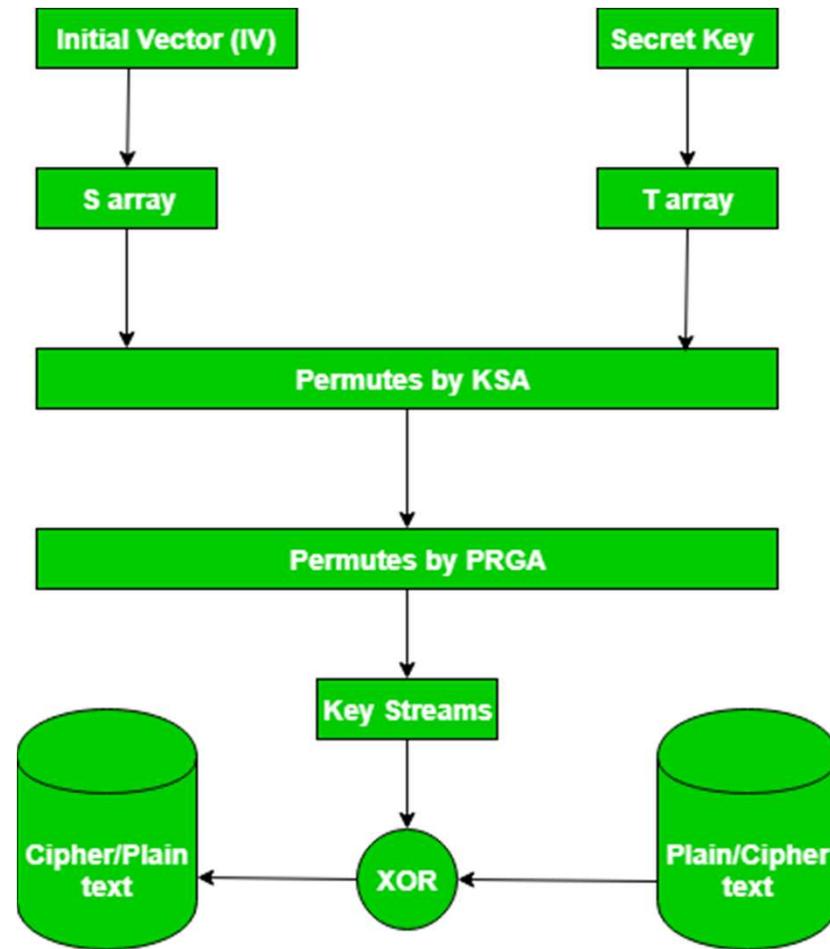
# RC4 (Rivest Cipher 4)

- A proprietary cipher owned by RSA Security (Ron Rivest, Adi Shamir and Leonard Adleman).
- RC4 was designed by Ron Rivest.
- Simple but effective.
- Variable key size, byte-oriented stream cipher.
- Widely used (Web SSL/TLS, Wireless WEP).
- Key formed by the random permutations of all 8-bit values.
- Permutation is used to scramble input info.
- One byte is processed at a time.

## RC 4 BLOCK DIAGRAM



# RC4





# RC4

---

- Starts with an array S of numbers: 0.....255.
- Uses key to well and truly shuffle.
- S forms the internal state of the cipher.
- Encryption continues shuffling array values.
- Sum of shuffled pair selects “stream key” value from permutation

```
/* Initialization */
for i = 0 to 255 do
    S[i] = i;
    T[i] = K[i mod keylen];
```

---

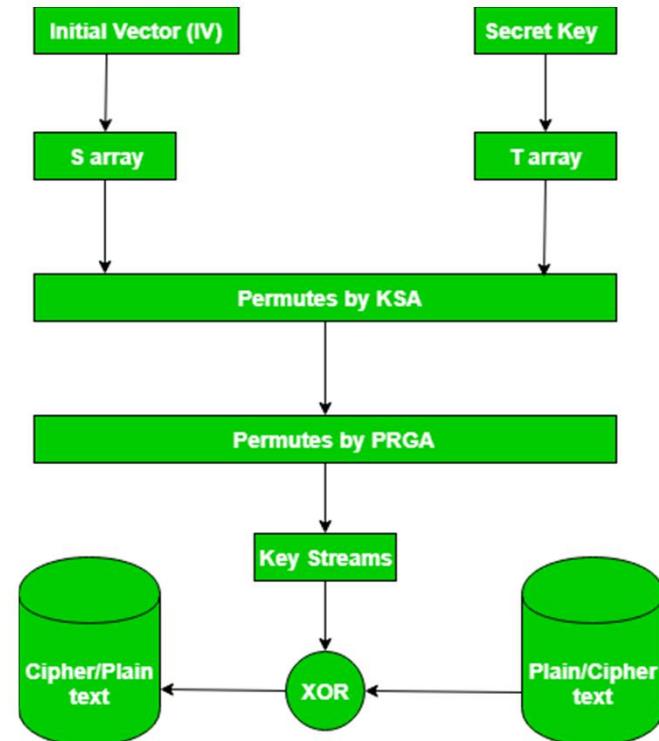
# RC4

```

/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap (S[i], S[j]);
}

/* Stream Generation */
i, j = 0;
while (true)
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 256;
    k = S[t];
}

```

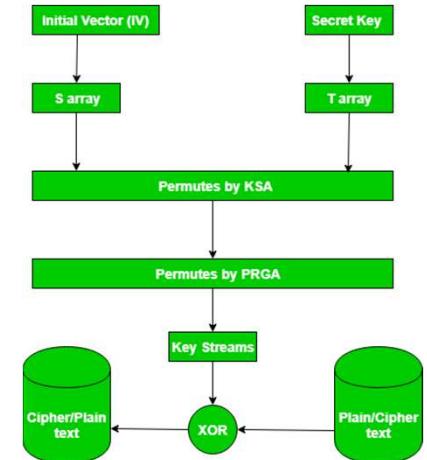


# Simplified RC4 (Initialization)

- Suppose the length of S-array is 8

```
/* Initialization */
for i = 0 to 255 do
    S[i] = i;
    T[i] = K[i mod keylen];
```

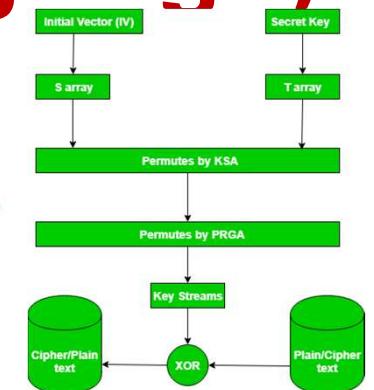
- $[S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7]$
- Initialize S-array = [0, 1, 2, 3, 4, 5, 6, 7]
- Let the Key [3, 1, 4, 1, 5]
- Initialize T-array = [3, 1, 4, 1, 5, 3, 1, 4]





# Simplified RC4 (Permute by Key Scheduling Algo)

```
/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap (S[i], S[j]);
```

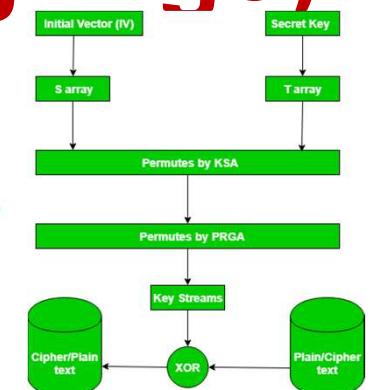


		$T_0=3$	$T_1=1$	$T_2=4$	$T_3=1$	$T_4=5$	$T_5=3$	$T_6=1$	$T_7=4$
i	j	$S_0$	$S_1$	$S_2$	$S_3$	$S_4$	$S_5$	$S_6$	$S_7$
0	0	0	1	2	3	4	5	6	7
1									
2									
3									
4									
5									
6									
7									



# Simplified RC4 (Permute by Key Scheduling Algo)

```
/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap (S[i], S[j]);
```



		$T_0=3$	$T_1=1$	$T_2=4$	$T_3=1$	$T_4=5$	$T_5=3$	$T_6=1$	$T_7=4$
i	j	$S_0$	$S_1$	$S_2$	$S_3$	$S_4$	$S_5$	$S_6$	$S_7$
	0	0	1	2	3	4	5	6	7
0	3	3	1	2	0	4	5	6	7
1	5	3	5	2	0	4	1	6	7
2	3	3	5	0	2	4	1	6	7
3	6	3	5	0	6	4	1	2	7
4	7	3	5	0	6	7	1	2	4
5	3	3	5	0	1	7	6	2	4
6	6	3	5	0	1	7	6	2	4
7	6	3	5	0	1	7	6	4	2



# Simplified RC4

---

- The final S-array will be  
[3, 5, 0, 1, 7, 6, 4, 2]



# Simplified RC4 (Generate Key Stream)

```
/* Stream Generation */
i, j = 0;
while (true)
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 256;
    k = S[t];
```

i	j	t	k	S <sub>0</sub>	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>	S <sub>4</sub>	S <sub>5</sub>	S <sub>6</sub>	S <sub>7</sub>
0	0			3	5	0	1	7	6	4	2
1	5	3	1	3	6	0	1	7	5	4	2
2	5	5	0	3	6	5	1	7	0	4	2
3	6	5	0	3	6	5	4	7	0	1	2
4	5	7	2	3	6	5	4	0	7	1	2

# Simplified RC4 (Generate Key Stream)



```
/* Stream Generation */
i, j = 0;
while (true)
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 256;
    k = S[t];
```

i	j	t	k	S <sub>0</sub>	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>	S <sub>4</sub>	S <sub>5</sub>	S <sub>6</sub>	S <sub>7</sub>
0	0			3	5	0	1	7	6	4	2

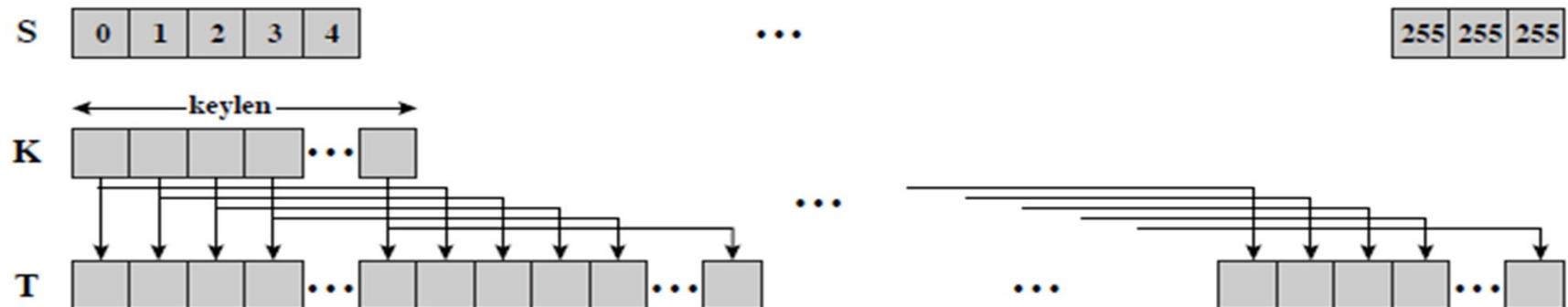


# Simplified RC4 (Encrypt)

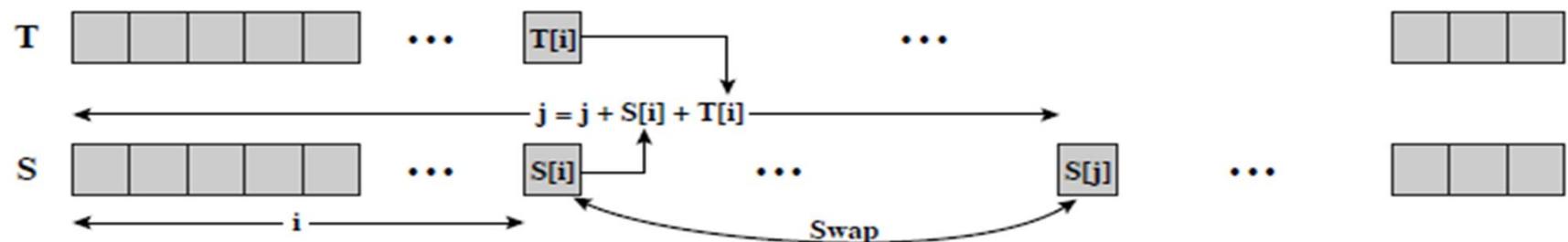
---

- PT = [6, 1, 5, 4]
- Key = [1, 0, 0, 2]
- CT = PT XOR Key
- PT = 0110 0001 0101 0100
- Key = 0001 0000 0000 0010
- CT = 0111 0001 0101 0110
- CT =   7       1       5       6

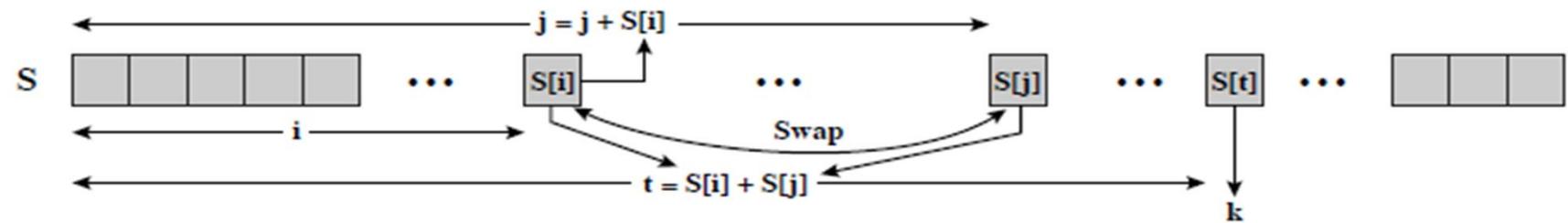
# RC4



(a) Initial state of S and T



(b) Initial permutation of S





# RC4 Security

---

- Claimed secure against known attacks.
- Result is very non-linear.
- Key shall not be reused as RC4 is a stream cipher.
- Have a concern with WEP, but due to key handling rather than RC4 itself.



---

Thanks!!!  
Queries?



**BITS** Pilani  
K K Birla Goa Campus

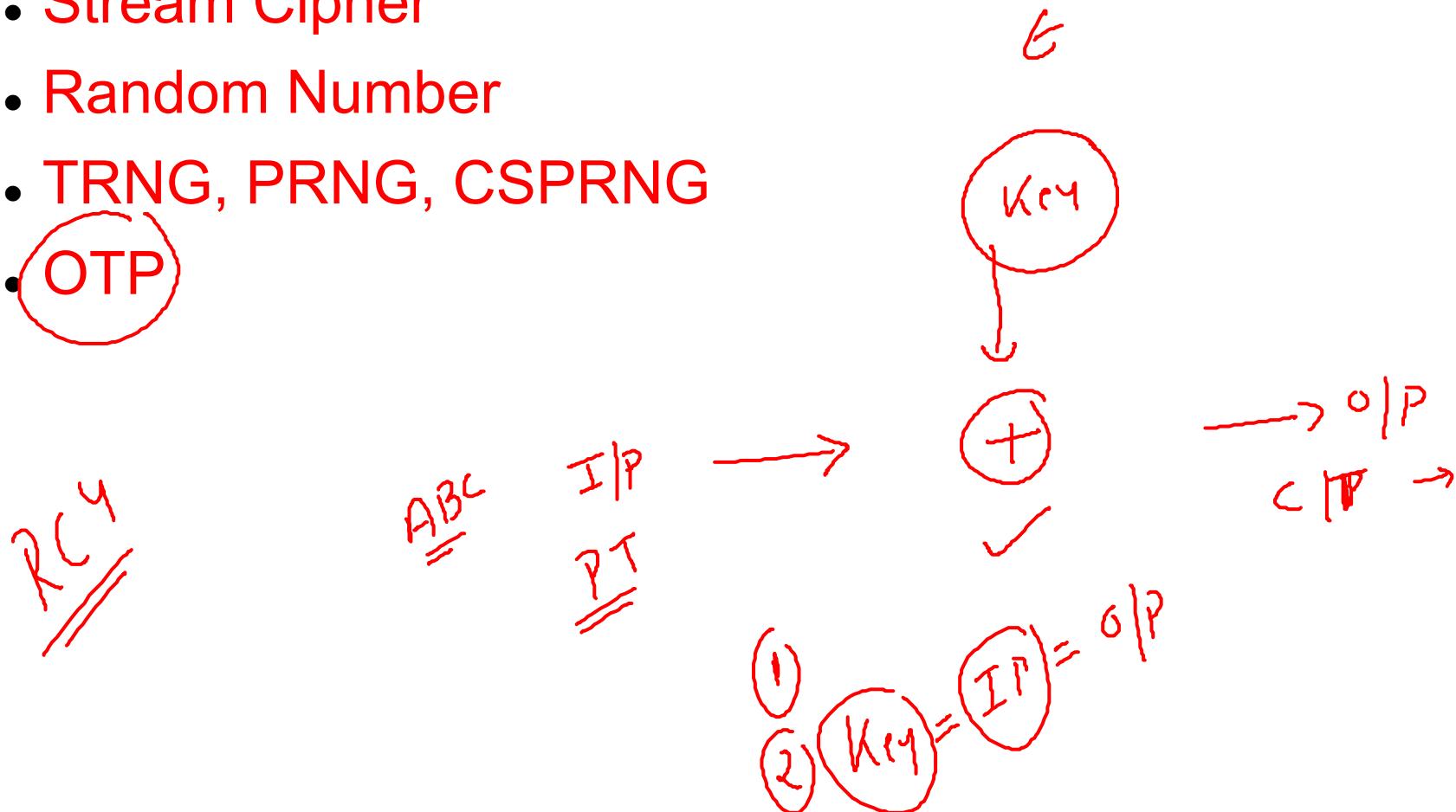
# Network Security

## SS ZG513

Hemant Rathore  
Department of Computer Science and Information Systems

# Previous Lecture

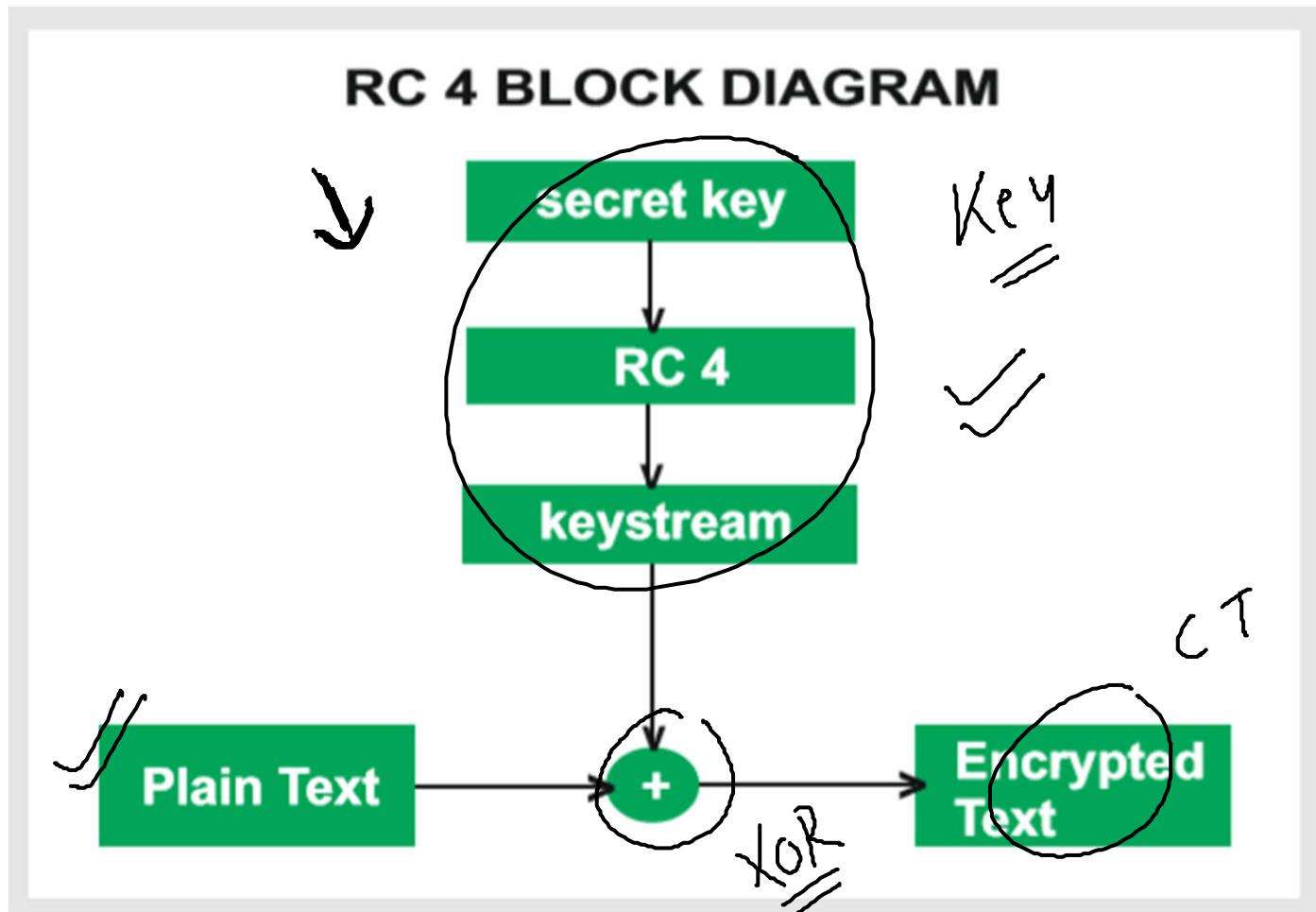
- Stream Cipher
- Random Number
- TRNG, PRNG, CSPRNG
- **OTP**



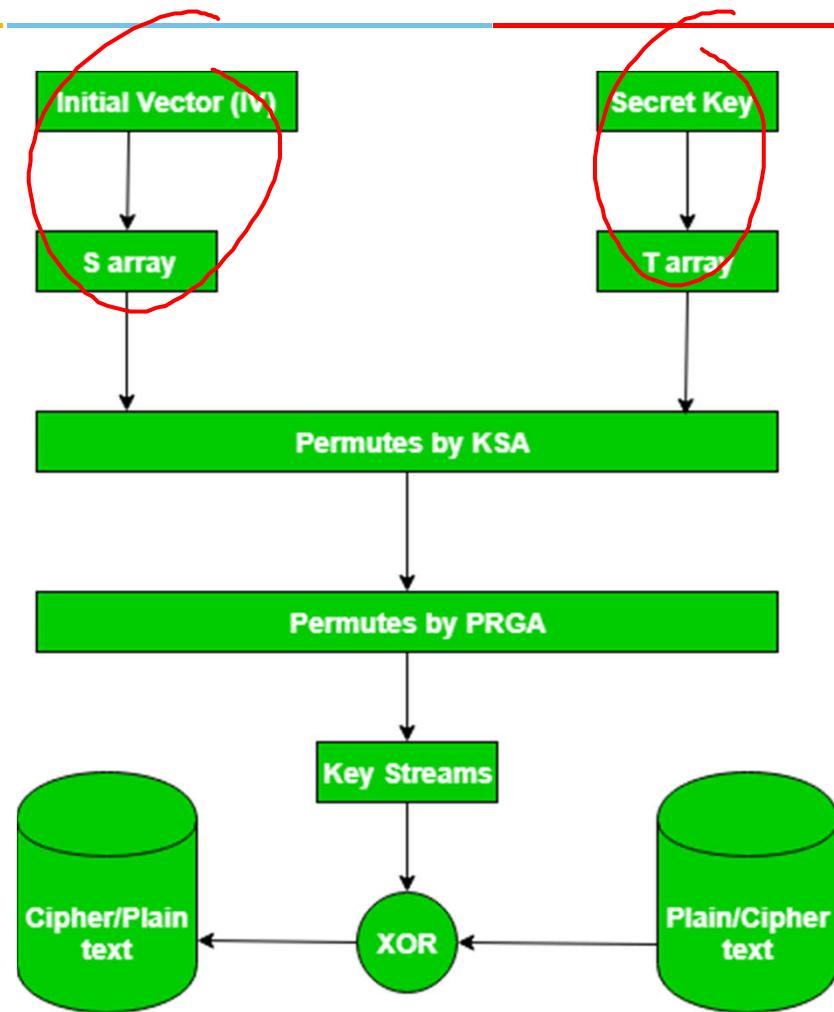


# RC4 (Rivest Cipher 4)

- A proprietary cipher owned by RSA Security (Ron Rivest, Adi Shamir and Leonard Adleman).
- RC4 was designed by Ron Rivest.
- Simple but effective.
- Variable key size, byte-oriented stream cipher. (+ + + +)
- Widely used (Web SSL/TLS, Wireless WEP).
- Key formed by the random permutations of all 8-bit values.
- Permutation is used to scramble input information.
- One byte is processed at a time.



# RC4



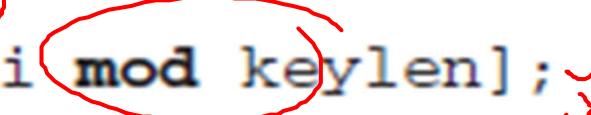
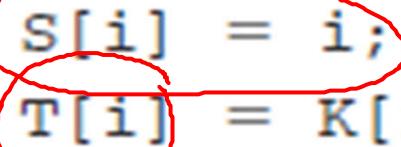
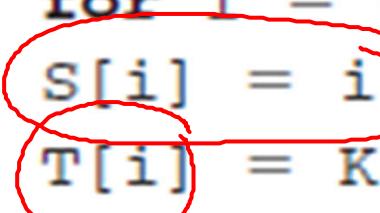
# RC4

6

- Starts with an array S of numbers: 0.....255.
- Uses key to well and truly shuffle.
- S forms the internal state of the cipher.
- Encryption continues shuffling array values.
- Sum of shuffled pair selects “stream key” value from permutation

```
/* Initialization */  
for i = 0 to 255 do  
    S[i] = i;  
    T[i] = K[i mod keylen];
```

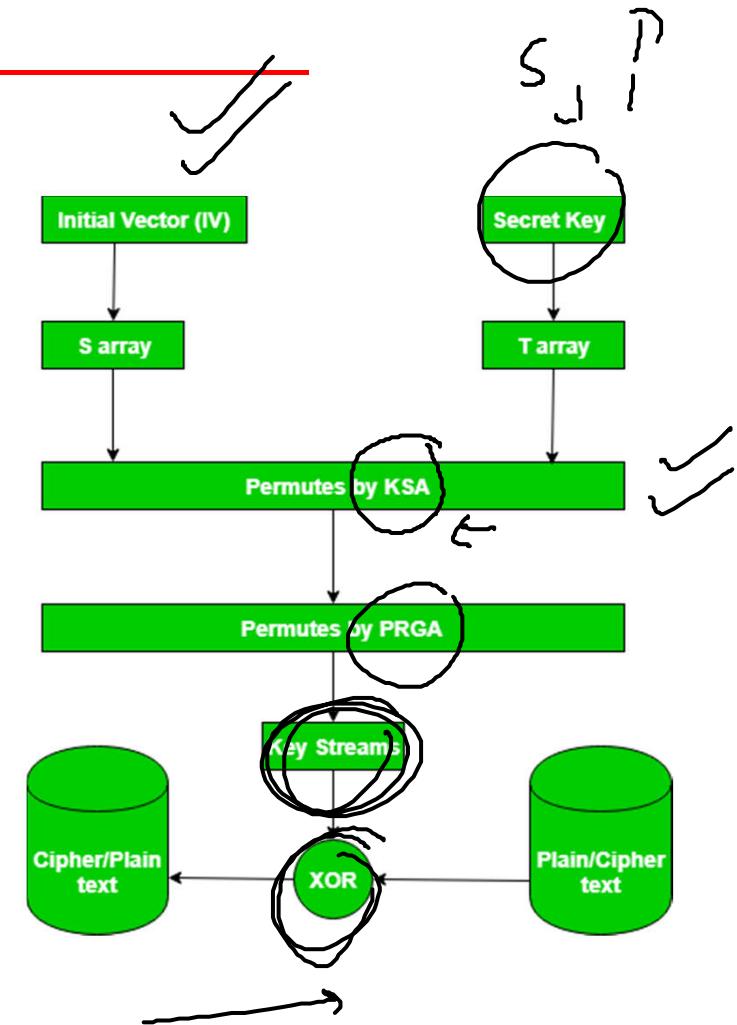
Variable key size ✓



# RC4

```
/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + K[i]) mod 256;
Swap (S[i], S[j]);
```

```
/* Stream Generation */
i, j = 0;
while (true)
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 256;
    k = S[t];
```



# Simplified RC4 (Initialization)

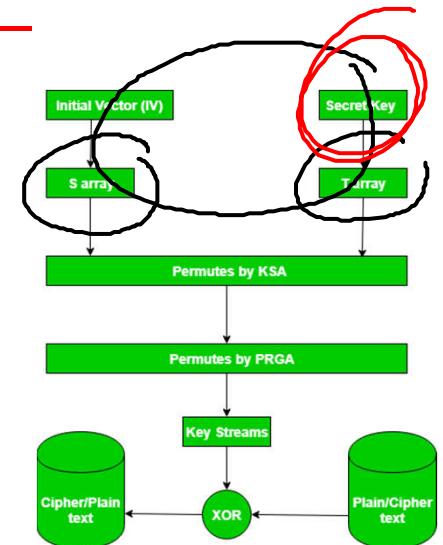
5 20  
1 25  
8 6  
8 8  
1 2  
1 2  
1 2  
1 2

- Suppose the length of S-array is 8

```
/* Initialization */
for i = 0 to 255 do
    S[i] = i;
    T[i] = K[i mod keylen];
```

- $[S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7]$
- Initialize S-array = [0, 1, 2, 3, 4, 5, 6, 7]
- Let the Key [3, 1, 4, 1, 5] ✓ ✓ ✓ ✓ ✓
- Initialize T-array = [3, 1, 4, 1, 5, 3, 1, 4]
 

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---



# Simplified RC4

## (Permute by Key Scheduling Algo)



```

j = 0
i = 0
j = 0 + s[0] + T[0]
= 0 + 0 + 3
j = 3
swap { s[0], s[3] }

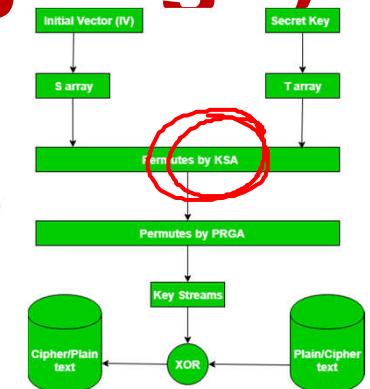
j = 1
i = 1
j = 3 + s[1] + T[1] mod 8
= 3 + 1 + 1
= 5 mod 8
j = 5
swap { s[1], s[5] }

```

```

/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
    j = (j + s[i] + T[i]) mod 256;
    Swap (s[i], s[j]);

```



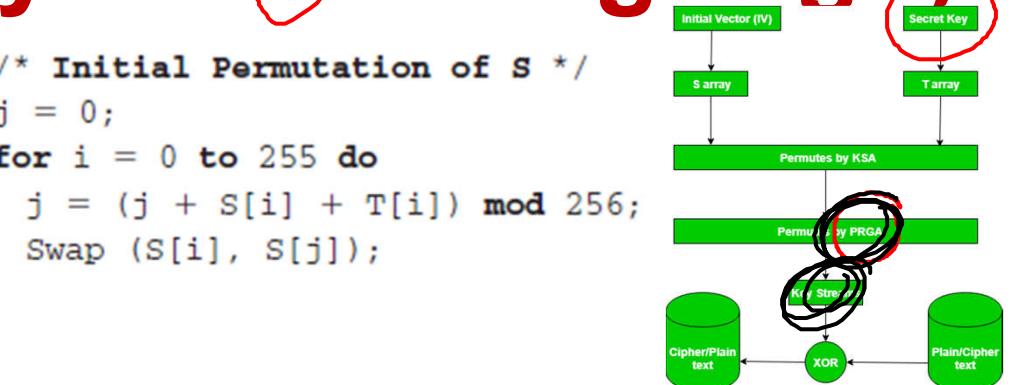
	<del>T<sub>0</sub>=3</del>	T <sub>1</sub> =1	T <sub>2</sub> =4	T <sub>3</sub> =1	T <sub>4</sub> =5	T <sub>5</sub> =3	T <sub>6</sub> =1	T <sub>7</sub> =4
i	j <del>s<sub>0</sub></del>	s <sub>1</sub>	s <sub>2</sub>	s <sub>3</sub>	s <sub>4</sub>	s <sub>5</sub>	s <sub>6</sub>	s <sub>7</sub>
0	3	1	2	3	4	5	6	7
1	3	5	2	0	4	1	6	7
2								
3								
4								
5								
6								
7								

# Simplified RC4

## (Permute by Key Scheduling Algo)



```
/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap (S[i], S[j]);
```



i	j	T <sub>0</sub> =3	T <sub>1</sub> =1	T <sub>2</sub> =4	T <sub>3</sub> =1	T <sub>4</sub> =5	T <sub>5</sub> =3	T <sub>6</sub> =1	T <sub>7</sub> =4	S <sub>0</sub>	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>	S <sub>4</sub>	S <sub>5</sub>	S <sub>6</sub>	S <sub>7</sub>
0	0	3	5	3	1	2	0	4	5	6	7	0	1	2	3	4	5
1	5	3	3	5	2	0	4	1	6	7	0	1	2	3	4	5	6
2	3	3	3	5	0	2	4	1	6	7	1	2	3	4	5	6	7
3	6	3	3	5	0	6	4	1	2	7	0	1	2	3	4	5	6
4	7	3	3	5	0	6	7	1	2	4	1	2	3	0	1	2	3
5	3	3	3	5	0	1	7	6	4	5	2	3	0	1	2	3	4
6	6	3	3	5	0	1	7	6	2	4	1	2	3	0	1	2	3
7	7	6	3	5	0	1	7	6	4	2	0	1	2	3	4	5	6



# Simplified RC4

---

- The final S-array will be

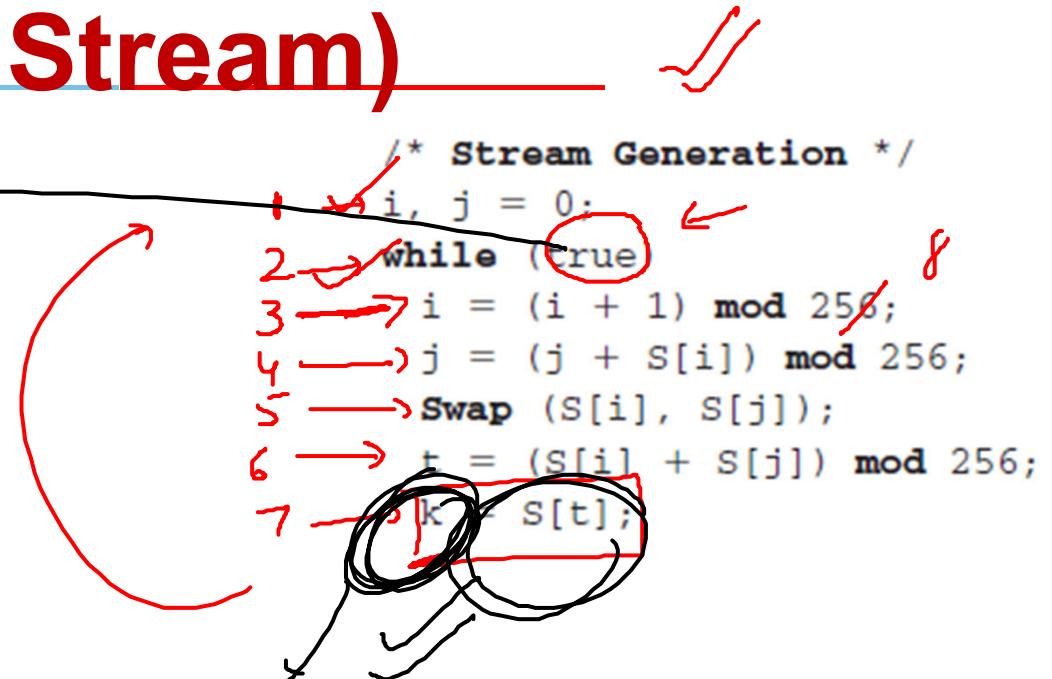
[3, 5, 0, 1, 7, 6, 4, 2]



# Simplified RC4 (Generate Key Stream)

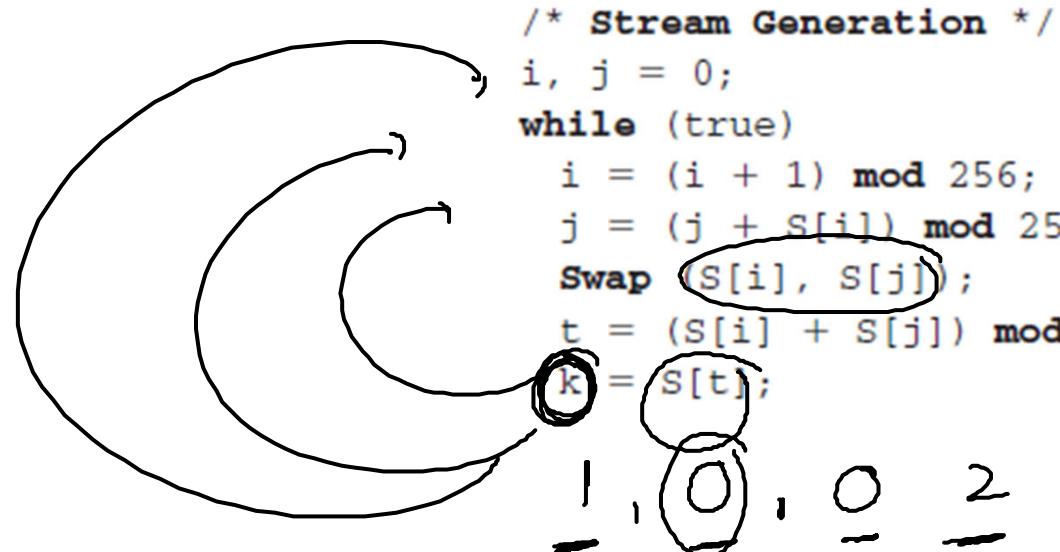


$D_4(a)_{17}^2$   
 $R^u$   
 $x = S[3]$   
 $= (6 + 5 \text{ mod } 8)$   
 $= 11 \text{ mod } 8$   
 $x = S[3]$   
 $x = S[3]$   
 $K = S[3]$   
 $K = 1$



i	j	t	k	S <sub>0</sub>	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>	S <sub>4</sub>	S <sub>5</sub>	S <sub>6</sub>	S <sub>7</sub>
0	0			3	5	0	1	7	6	4	2
				3	6	0	1	7	5	4	2

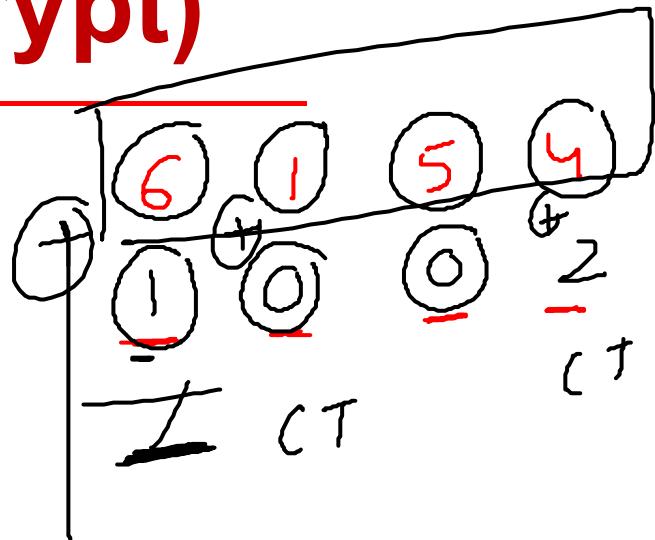
# Simplified RC4 (Generate Key Stream)



i	j	t	k	S <sub>0</sub>	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>	S <sub>4</sub>	S <sub>5</sub>	S <sub>6</sub>	S <sub>7</sub>
0	0			3	5	0	1	7	6	4	2
1	5	3	1	3	6	0	1	7	5	4	2
2	5	5	0	3	6	5	1	7	0	4	2
3	6	5	0	3	6	5	4	7	0	1	2
4	5	7	2	3	6	5	4	0	7	1	2

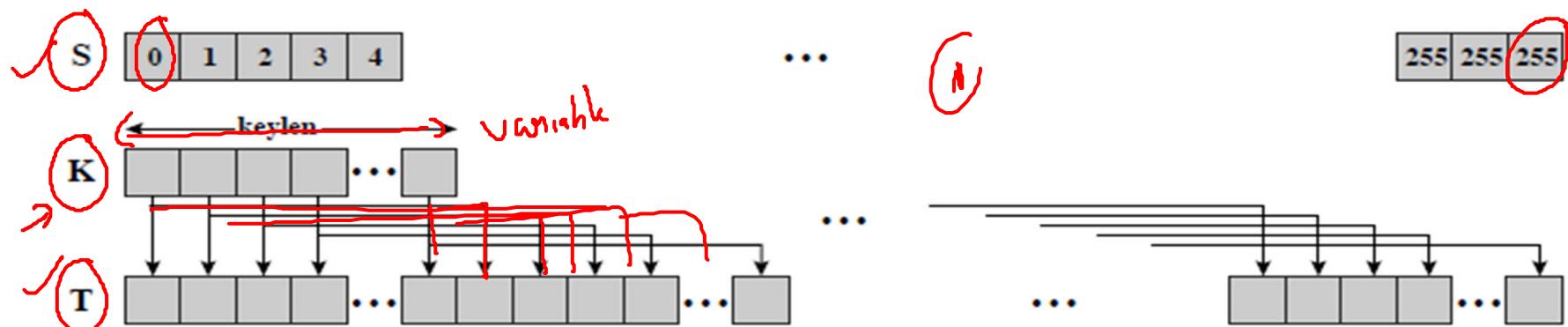
# Simplified RC4 (Encrypt)

- PT = [6, 1, 5, 4]
- Key = [1, 0, 0, 2]
- CT = PT XOR Key
- PT = 0110 0001 0101 0100
- Key = 0001 0000 0000 0010
- CT = 0111 0001 0101 0110
- CT = 7 1 5 6

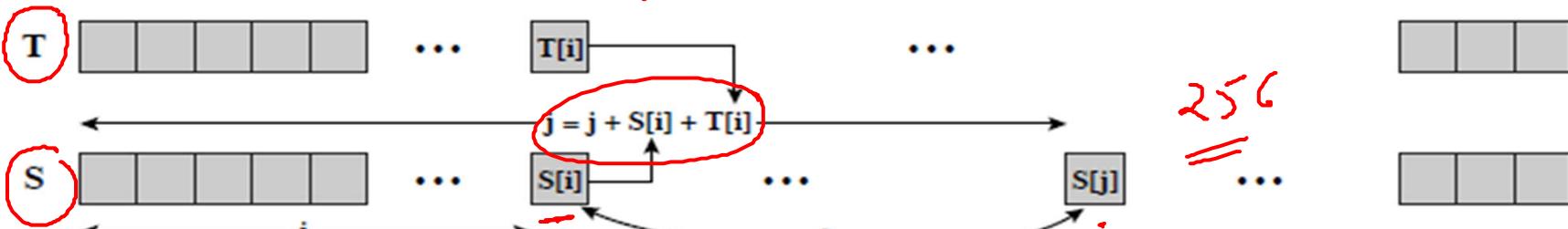


# RC4

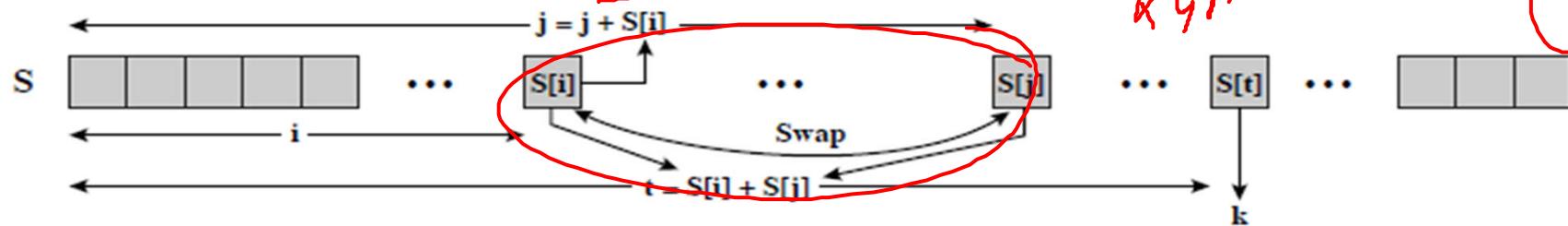
256



(a) Initial state of S and T



(b) Initial permutation of S



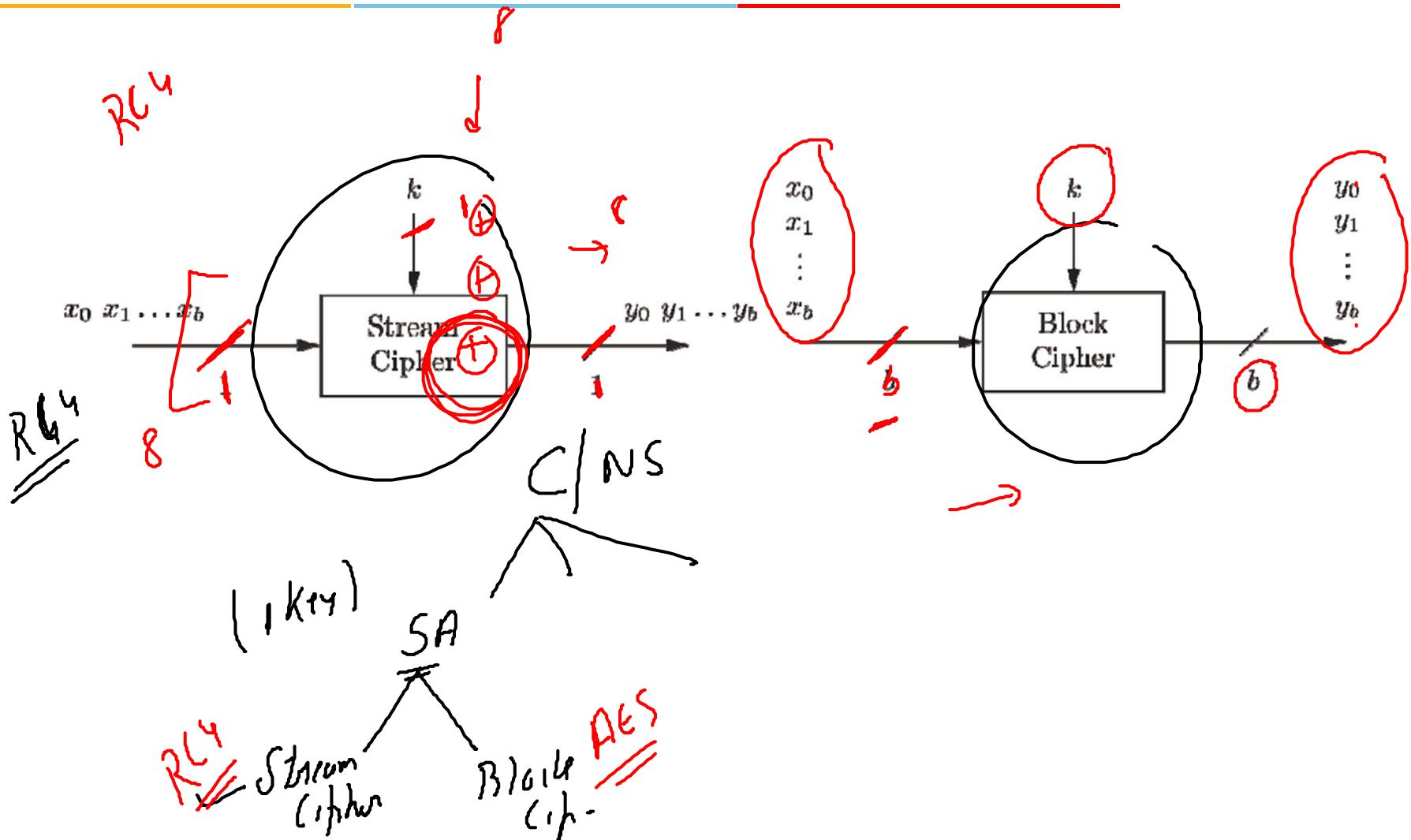


# RC4 Security

---

- Claimed secure against known attacks.
- Result is very non-linear.
- Key shall not be reused as RC4 is a stream cipher.
- Have a concern with WEP, but due to key handling rather than RC4 itself.

# Introduction



# AES History

- Call for encryption algorithm by NBS in 1973
- DES (Data Encryption Standard, IBM) - 1976 published as standard
  - Block Size : 64 bits
  - Key Size : 56 bits
- Replacement for DES was needed by 1995
  - theoretical attacks
  - exhaustive key search attacks
- Can use Triple-DES – but slow, has small blocks

# Introduction to AES

- US NIST issued call for ciphers in 1997
- 22 submissions
  - 7 did not satisfy all requirements
  - 15 submissions (August 1998)
  - 5 finalists (August 1999)
    - Mars - IBM Corporation
    - RC6 - RSA Laboratories
    - Rijndael - J. Daemen & V. Rijmen
    - Serpent - Eli Biham et al.
    - Twofish - B. Schneier et al..
  - Winner (October 2000): Rijndael.
- Published FIPS PUB 197 standard by NIST in 2001.

# Introduction to AES

- AES is based on a competition, won by Rijmen and Daemen (Rijndael) from Belgium.

AES

- Block Size: 128 bits
- Key Size: 128, 192, 256 bits  
(AES-128, AES-192, AES-256)
- Larger Key size

More secure (More Rounds)

Slower

AES



key lengths	# rounds = $n_r$
128 bit	10
192 bit	12
256 bit	14

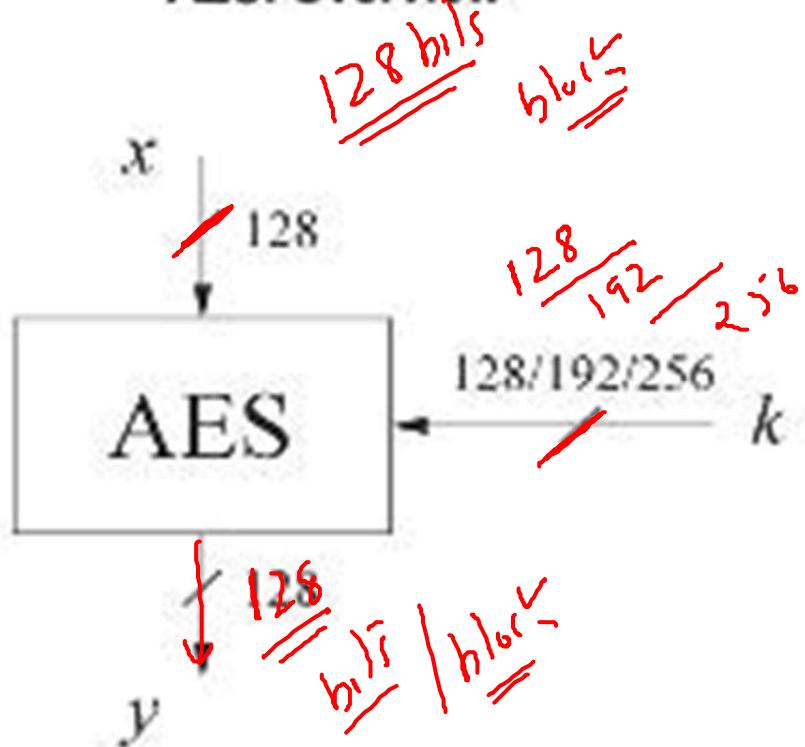
AES

AES



$$2^{4^1} = \cancel{16}$$

## AES: Overview



The number of rounds depends on the chosen key length:

Key length (bits)	Number of rounds
128	10
192	12
256	14

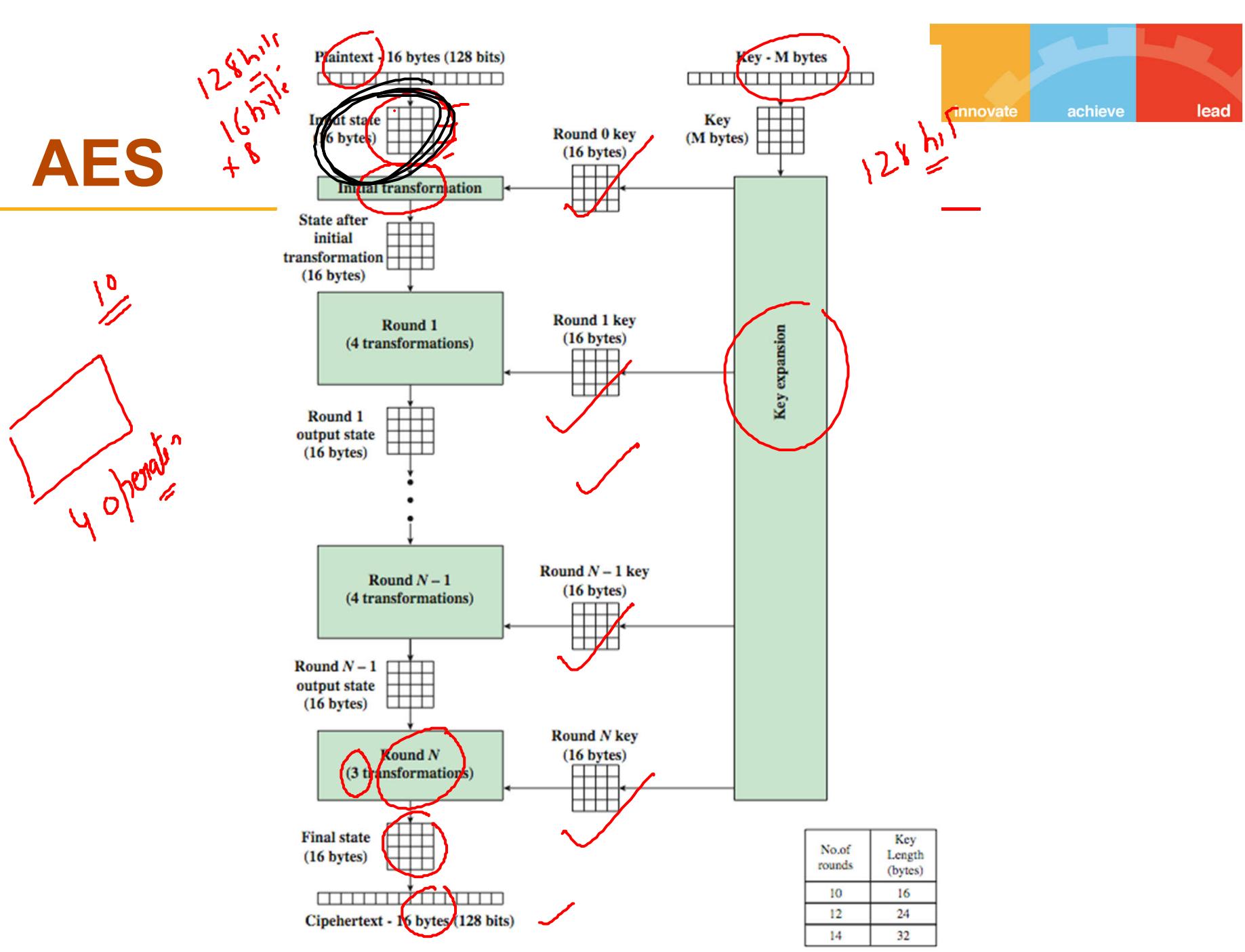


# Introduction to AES

---

- An iterative rather than Feistel cipher.
  - Operates on entire data block in every round.

# AES

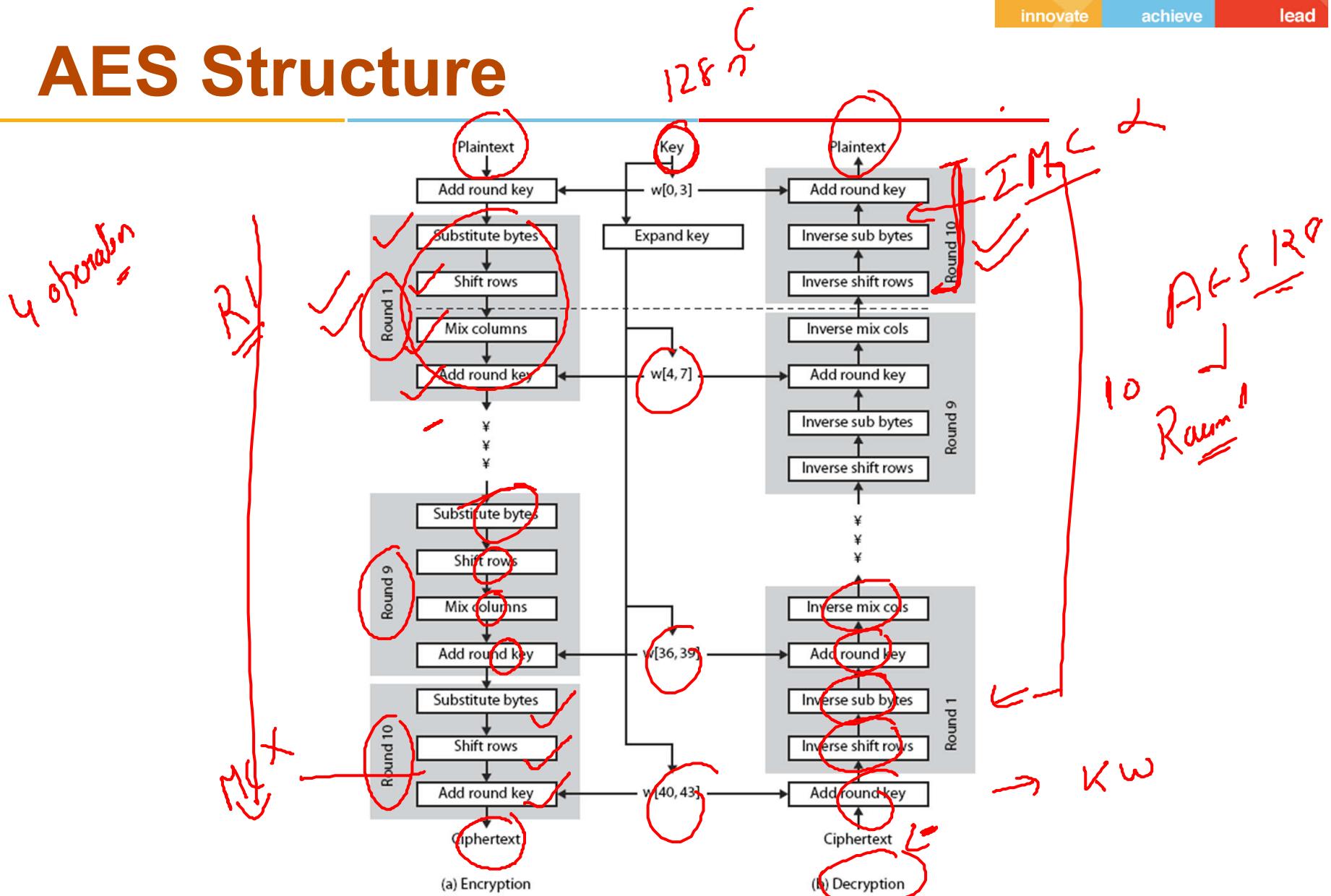




# Structure of AES

---

# AES Structure



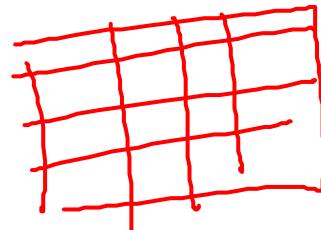
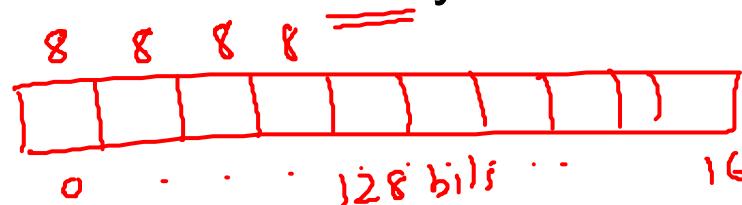
# Overview of AES

AES consists of:

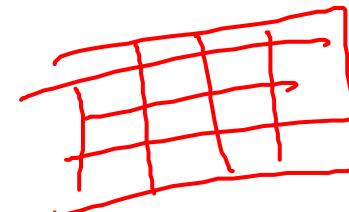
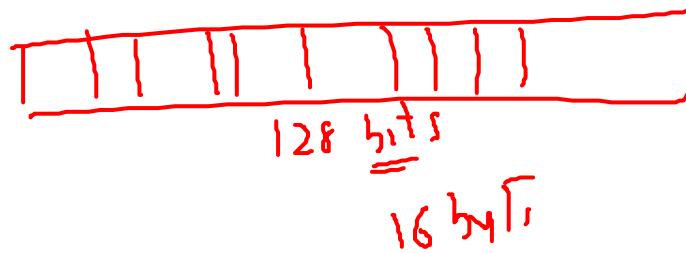
- Confusion Layer
  - Confusion: each bit of the cipher text should depend on several parts of the key.
  - Byte substitution layer (S-box):  
Introduces confusion to the data by transforming the data non-linearly and assures that changes in individual state bits propagate quickly across the data path.
- Diffusion layer
  - Diffusion: if we change a single bit in the plaintext, then (statistically) half of the bits in the cipher text should change.
  - Provides diffusion over all the state, consists two sublayers ShiftRows and MixColumn layer.

# Internal Structure of AES

- The 128-bit data path consists of 16 bytes is arranged in a 4-by-4 byte matrix

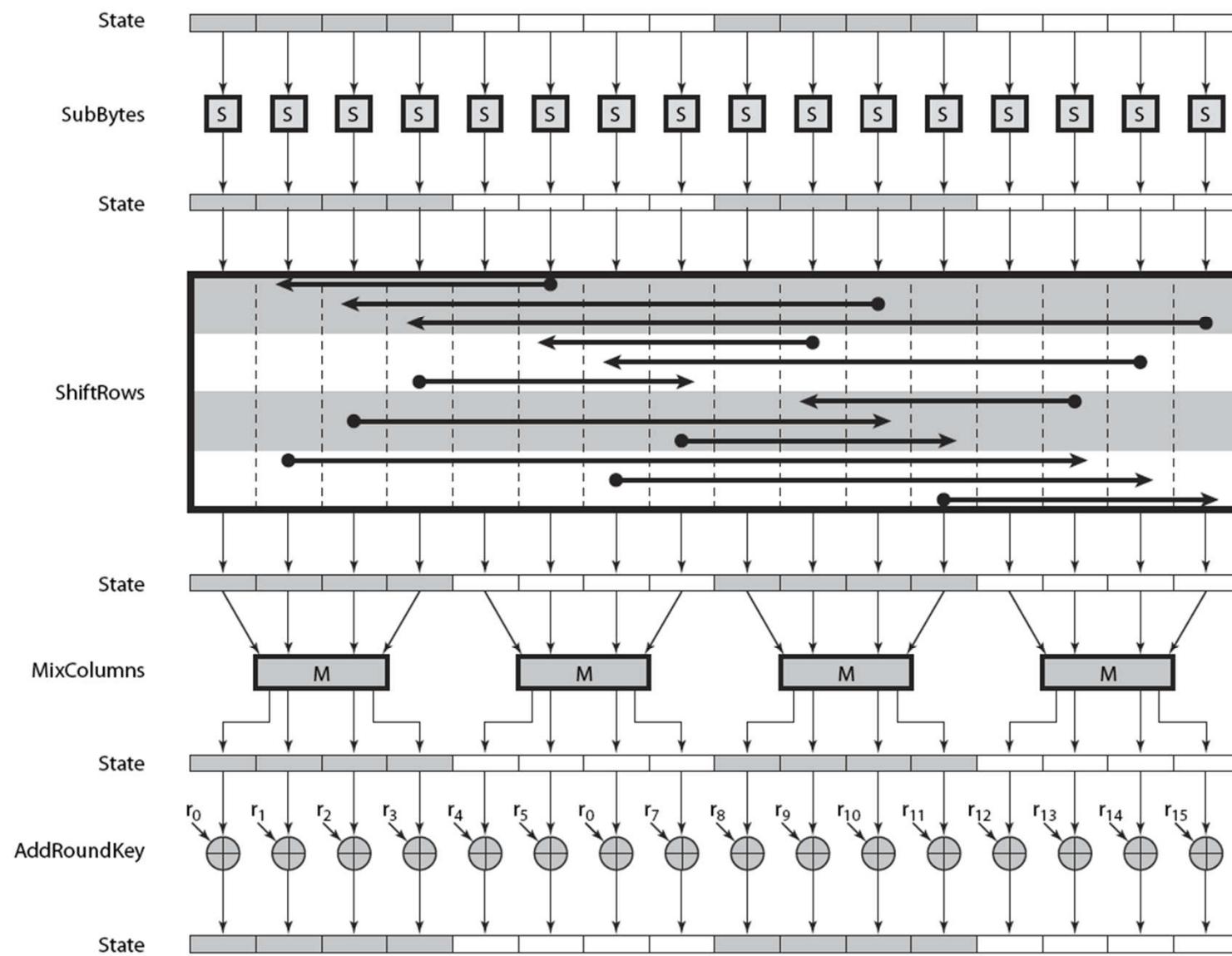


- The key bytes are arranged into a matrix with 4 rows and 4 (128-bit key), 6 (192-bit key), 8 (256-bit key) columns.

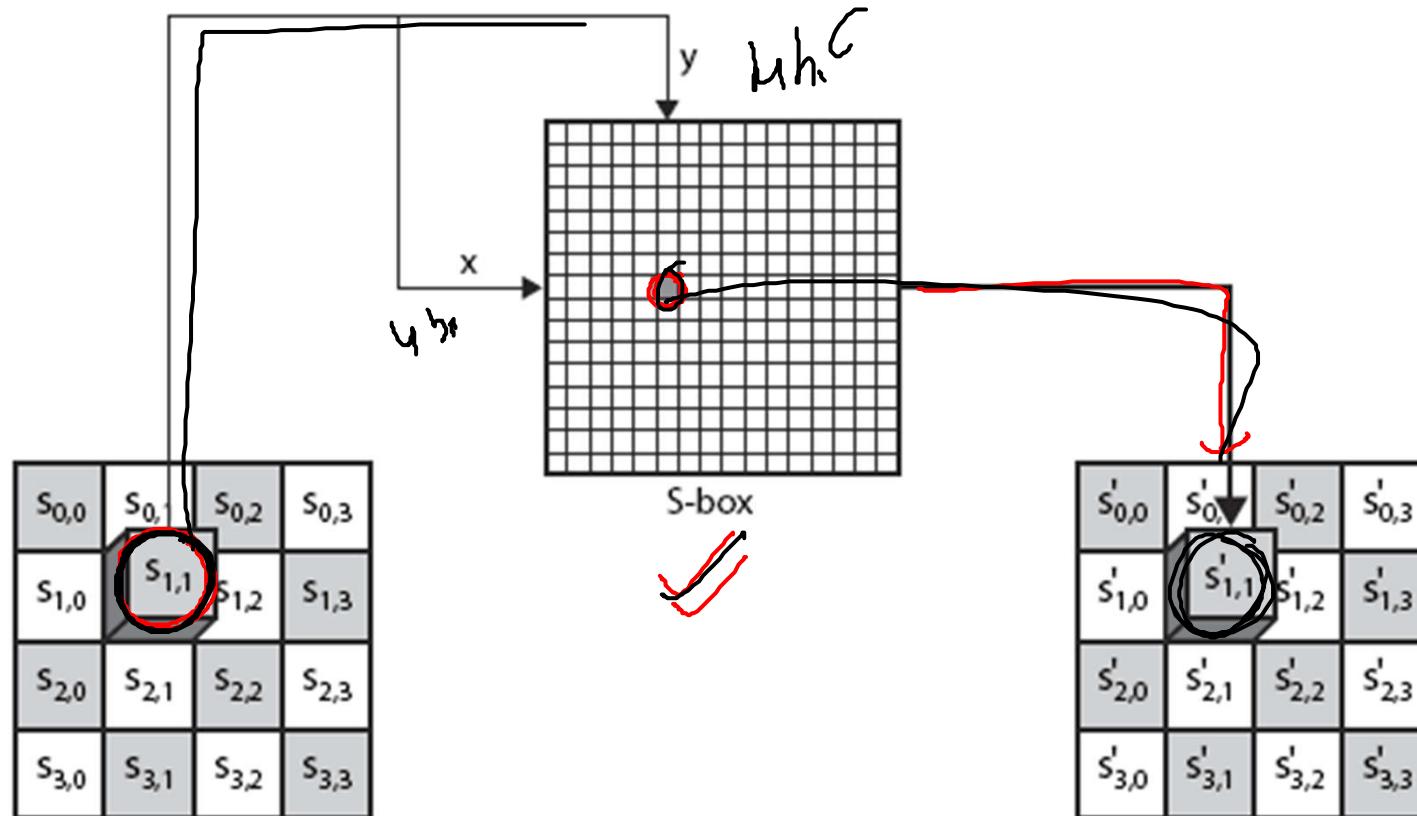




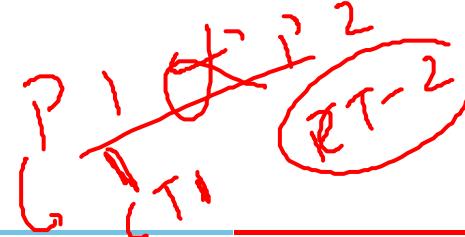




# SubBytes



# SubBytes



- Simple substitution of each byte
- uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
- Each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)
- S-box constructed using defined transformation of values in  $GF(2^8)$
- Designed to be resistant to all known attacks

# SubBytes

Handwritten notes on the left side of the table:

- Top-left: "1 byt", "4 bits", "y bits", "y bits", "bits", "bits", "bits", "bits".
- Middle-left: "X" with a circled "A".
- Bottom-left: "AES".

	y															
x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
X	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
8	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
9	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
A	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
B	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
C	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
D	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
E	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16
F																

Handwritten arrow from the bottom-left points to the value "06" in the 5th column of the 1st row.

Handwritten text "06" is written below the table.

Handwritten text "AES S-Box" is written to the right of the table.

# Shift Rows

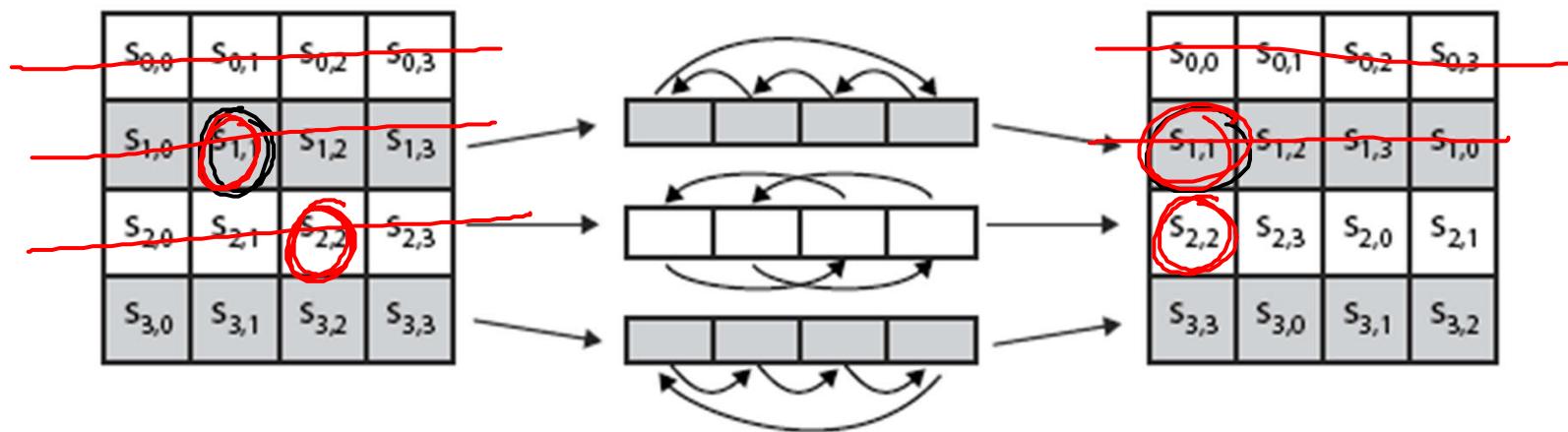
→ Row  
 → 1 LS  
 → 2 LS  
 → 3 LS  
 = by

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

Left shift  
by 1

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

# Shift Rows





# Shift Rows

---

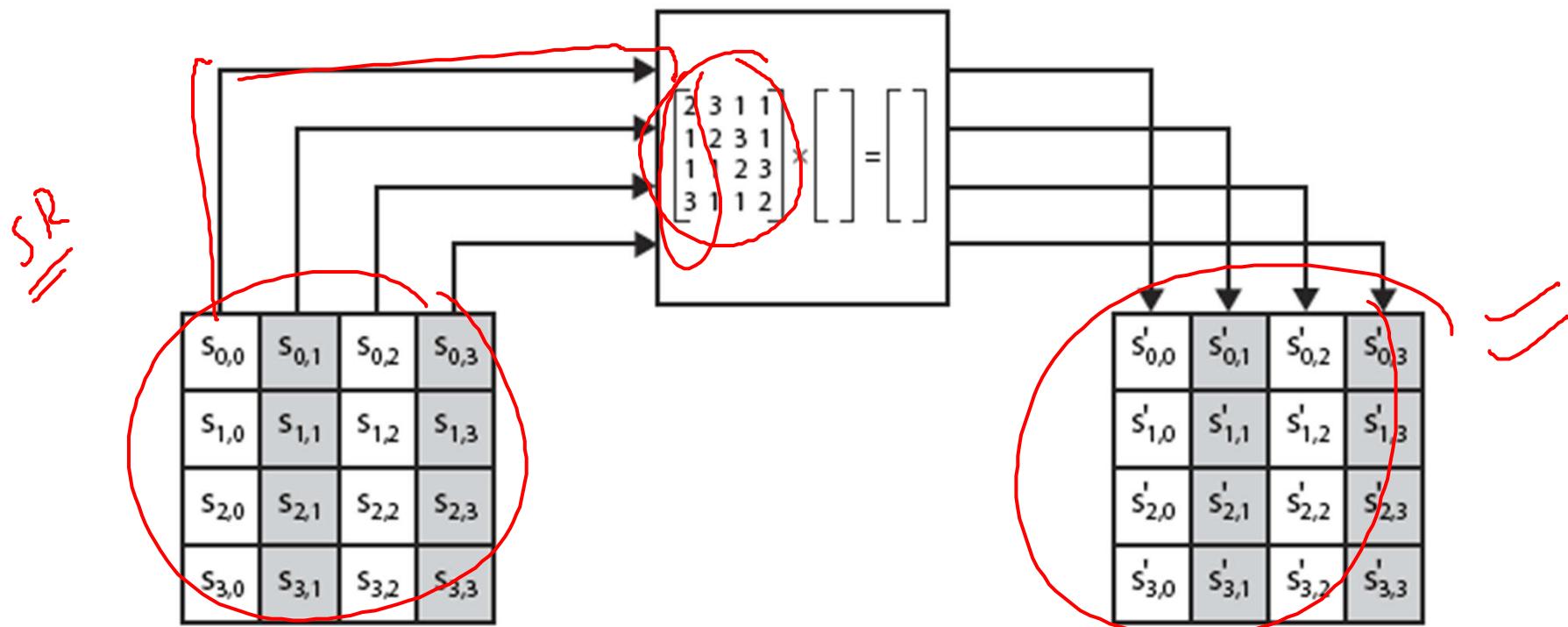
- a circular byte shift in each
  - 1st row is unchanged ✓
  - 2nd row does 1 byte circular shift to left ✓
  - 3rd row does 2 byte circular shift to left ✓
  - 4th row does 3 byte circular shift to left ✓
- decrypt inverts using shifts to right
- since state is processed by columns, this step permutes bytes between the columns

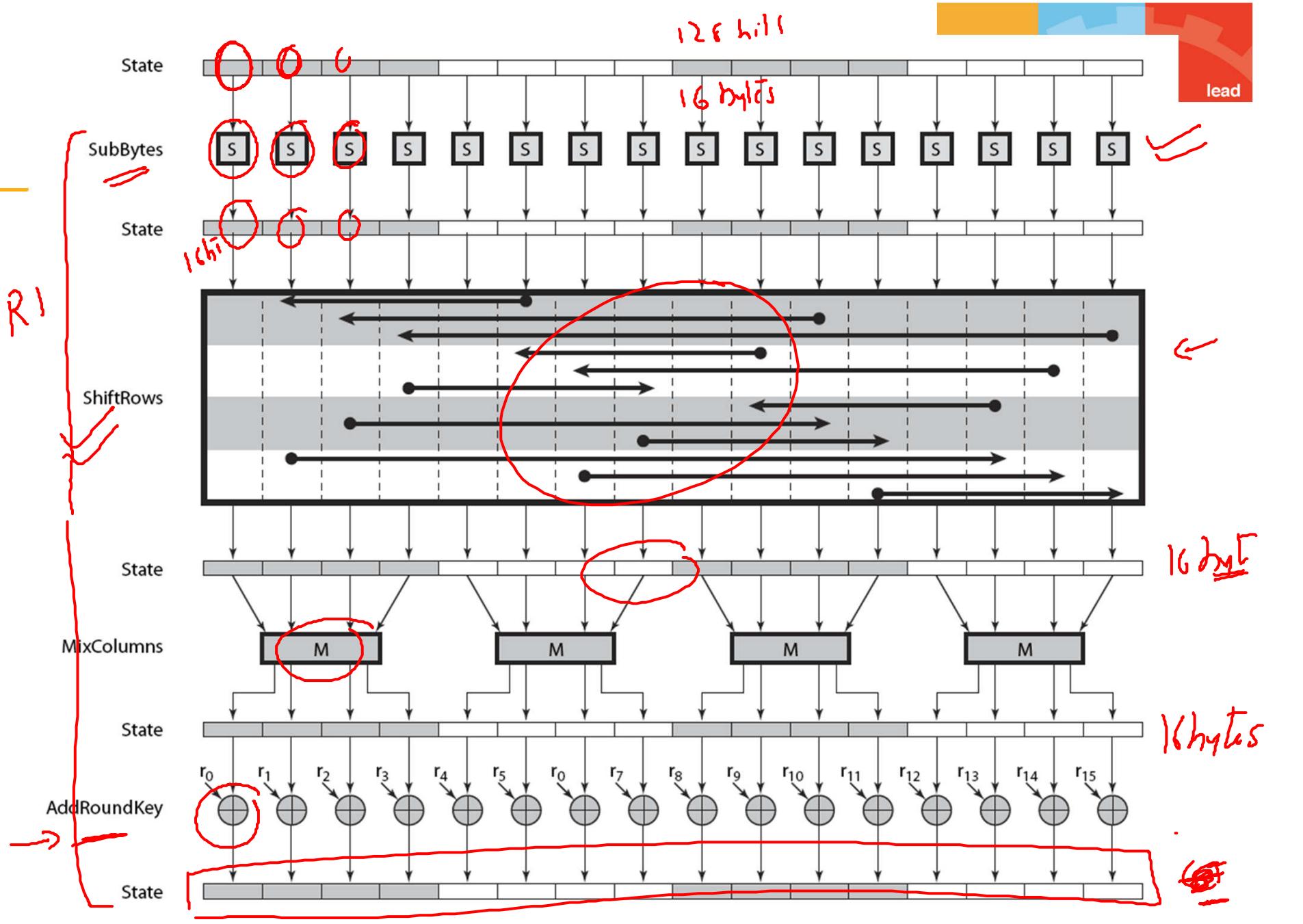


# Mix Columns

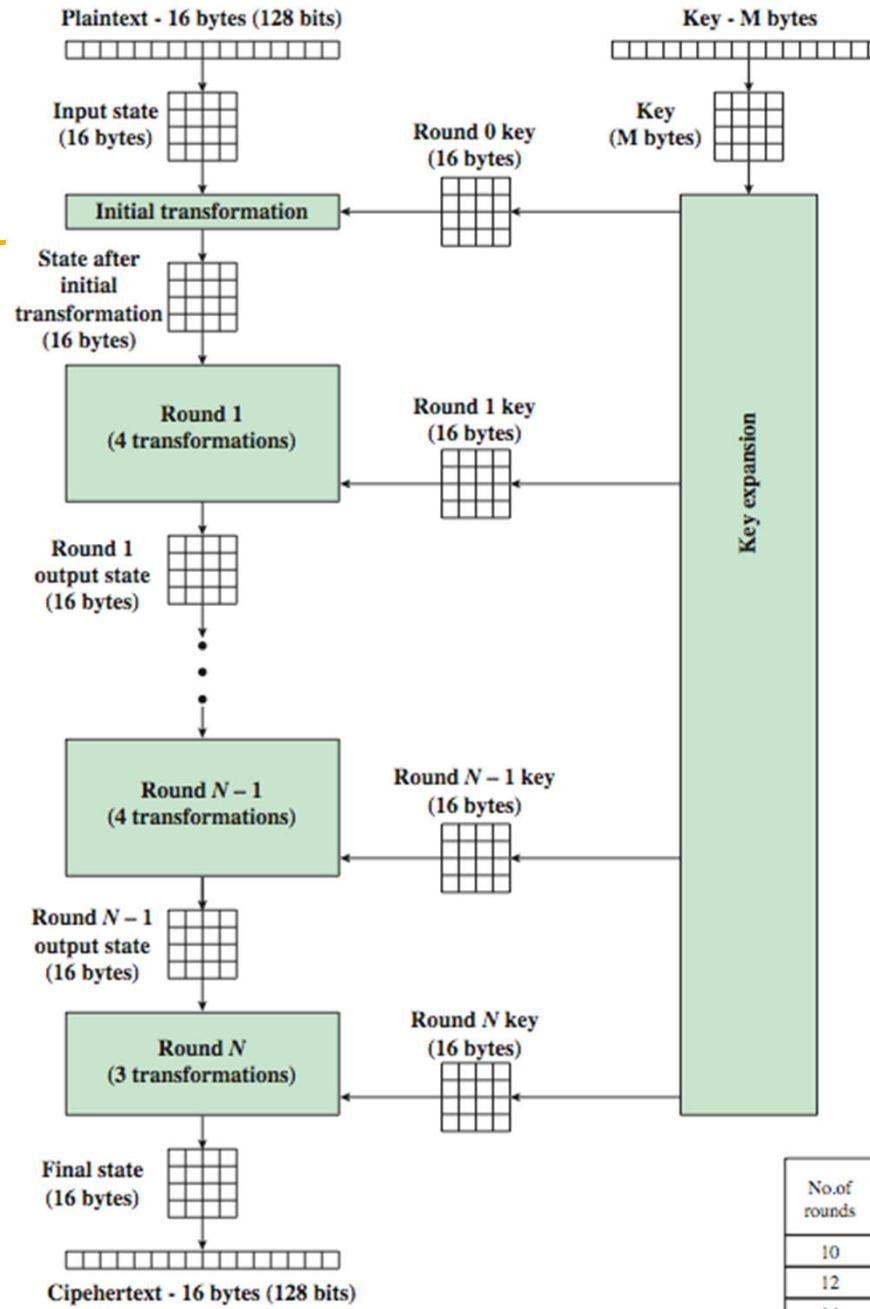
---

# Mix Columns





# AES



No.of rounds	Key Length (bytes)
10	16
12	24
14	32



# Implementation and Security

- Efficient in software and hardware.
- AES is part of numerous open standard such as ~~Ipsec/TLS~~.
- Will dominant encryption algorithm for many years.
- Not based on Feistel networks. Its basic operations use Galois field arithmetic and provide strong diffusion and confusion.
- Provides excellent long-term security against brute-force attack (128/192/256 bits key).
- Studied intensively in 1990s and no attacks have been found that are better than brute-force.  
*2<sup>128</sup>*
- Pre-computations adds a small latency to the decryption operation relative to encryption.



---

Thanks!!!  
Queries?



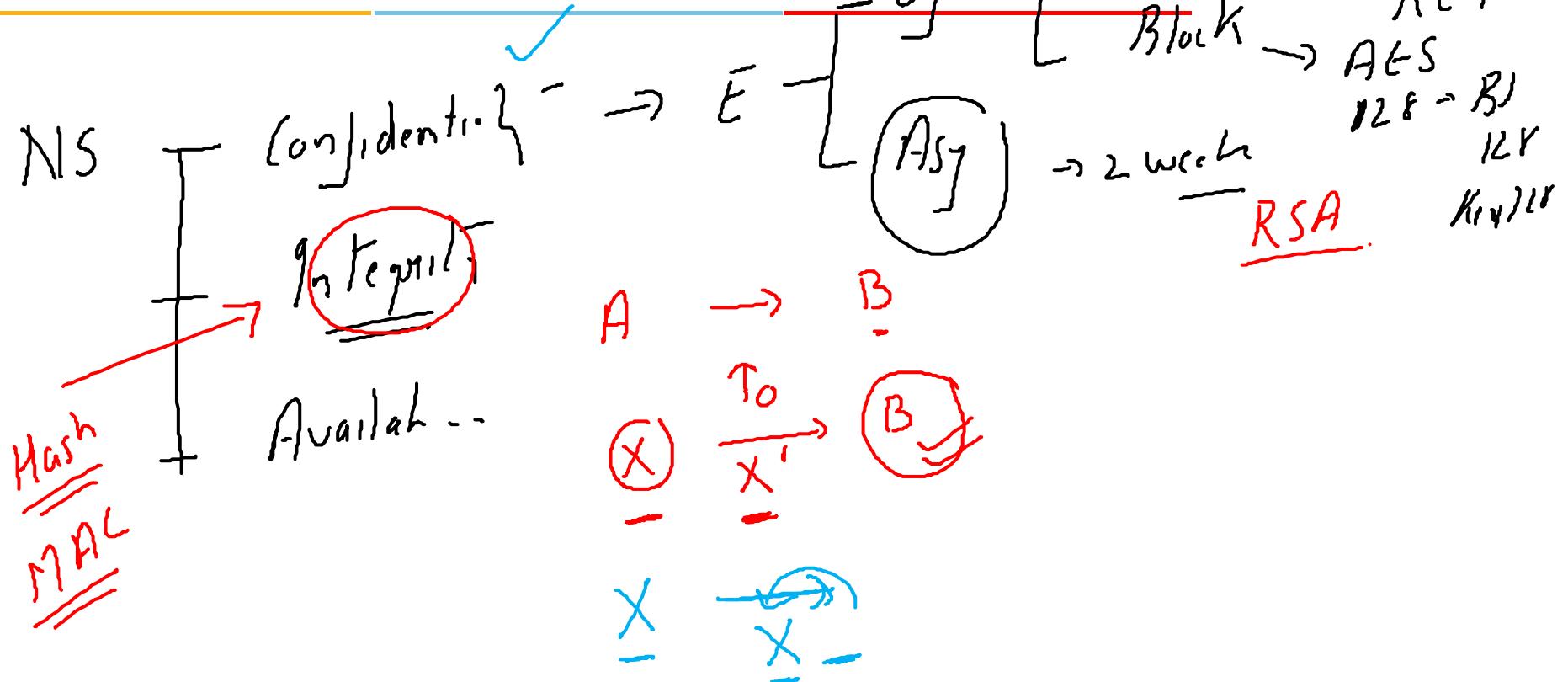
**BITS** Pilani  
K K Birla Goa Campus

# Network Security

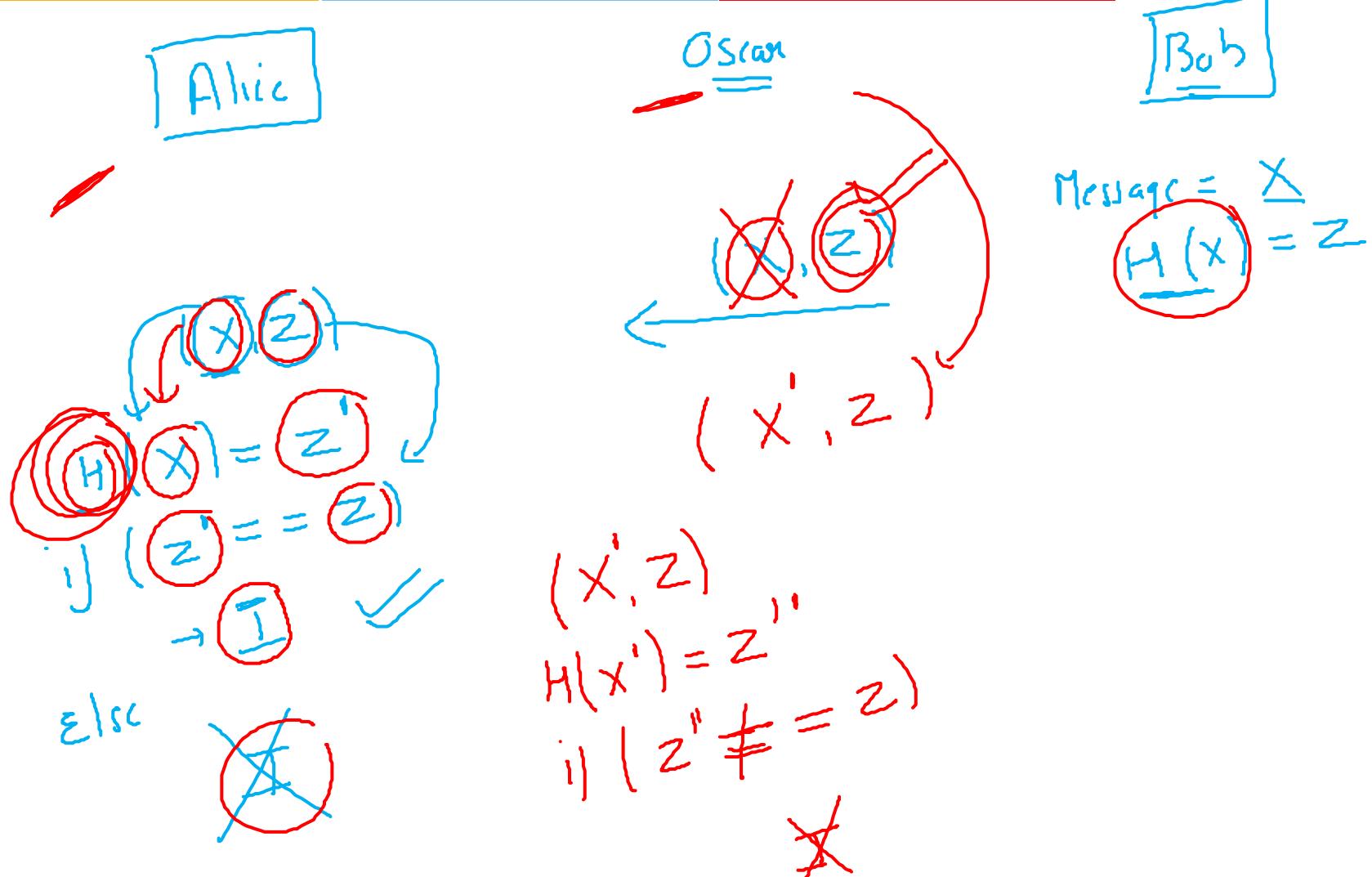
## SS ZG513

Hemant Rathore  
Department of Computer Science and Information Systems

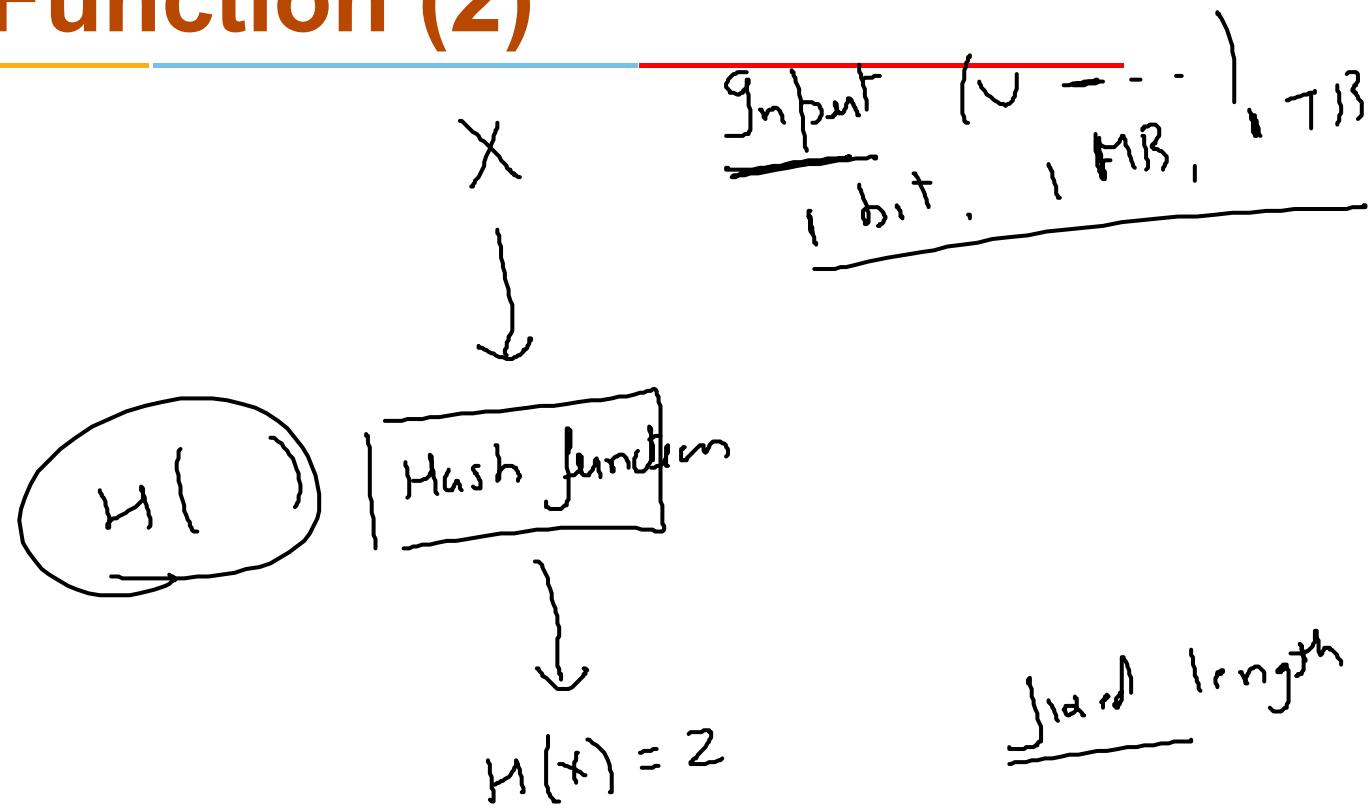
# Introduction



# Hash Function (1)



# Hash Function (2)



# Hash Function

---

- A widely used protocol to compute a unique representation of message (message digest / finger print / hash value).  
$$h(x)$$
- No key.
- Digital signatures, key derivation, data traffic, checksum, speed up the lookup tables, duplicates.

# Hash Function

Input

- Principle: Takes a string of arbitrary length and maps it to a fixed-length output, referred to as hash value.  
*(160 bits)*
- Practical Requirements: Arbitrary message size, Fixed output length, Finger print should be highly sensitive, Efficiency.
- Security Requirements: Pre-image resistance, Second pre-image resistance, collision resistance, Non-correlation, Near collision resistance, partial-image resistance.

# Types of Hash Function

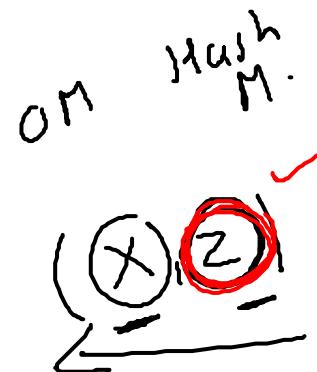
- Block cipher based hash functions:
  - Use block cipher such as AES to construct hash function.
- Dedicated Hash functions:
  - Specifically designed to serve as a hash function.
  - Fact: Any compression function 'f' which is collision resistant can be extended to a collision resistant hash function.
  - Merkle-Damgård Construction: Blocks are processed sequentially by the hash function, which has a compression function at its heart.

# Block Cipher based Hash Functions

Alice

$$\begin{aligned}
 & (x, z) \\
 & \downarrow \\
 & \text{AES}_K(x) = z \\
 & \text{if } (z) == z \\
 & \quad \quad \quad \boxed{z} \quad \checkmark
 \end{aligned}$$

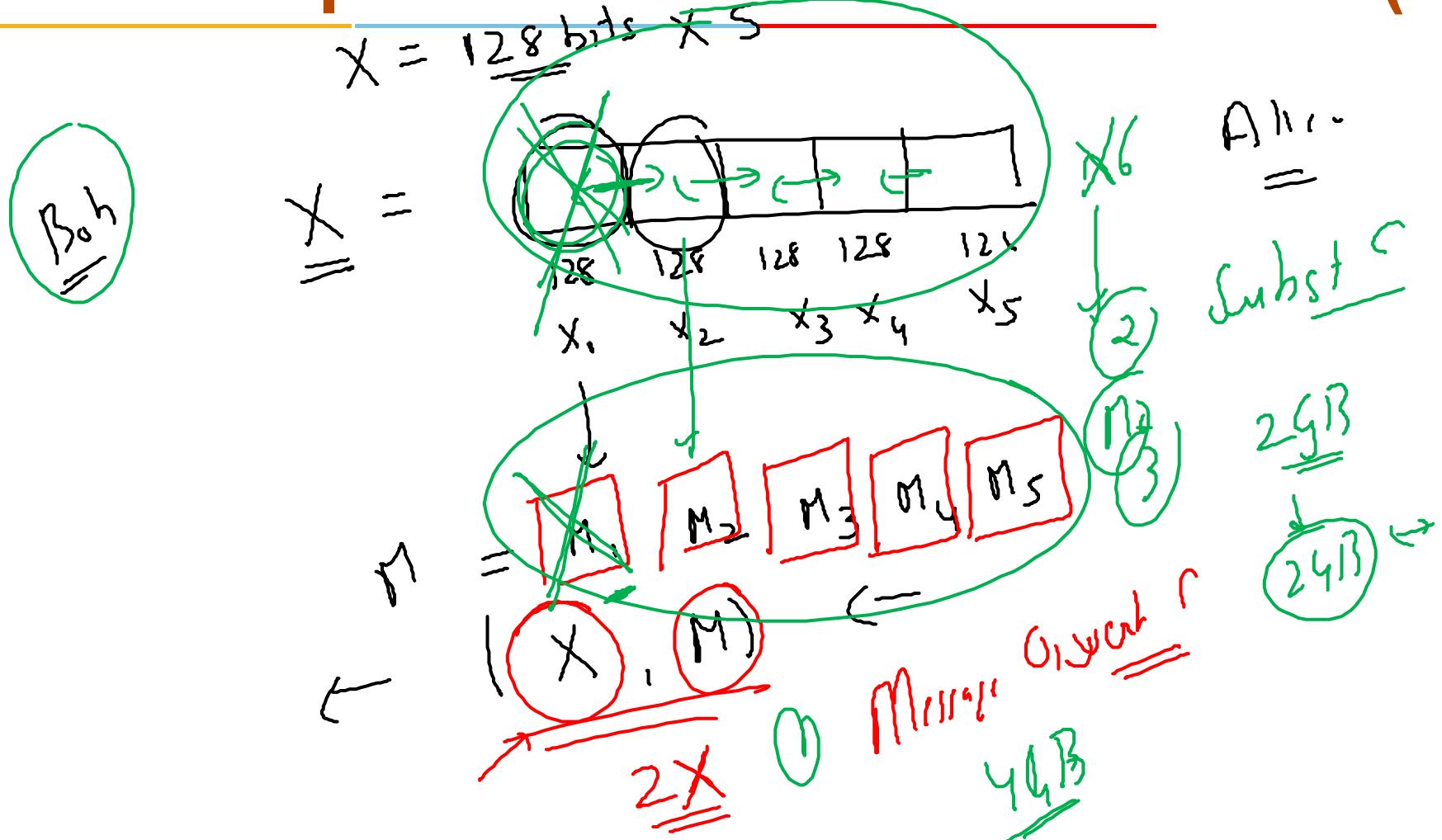
Eve ~~x~~



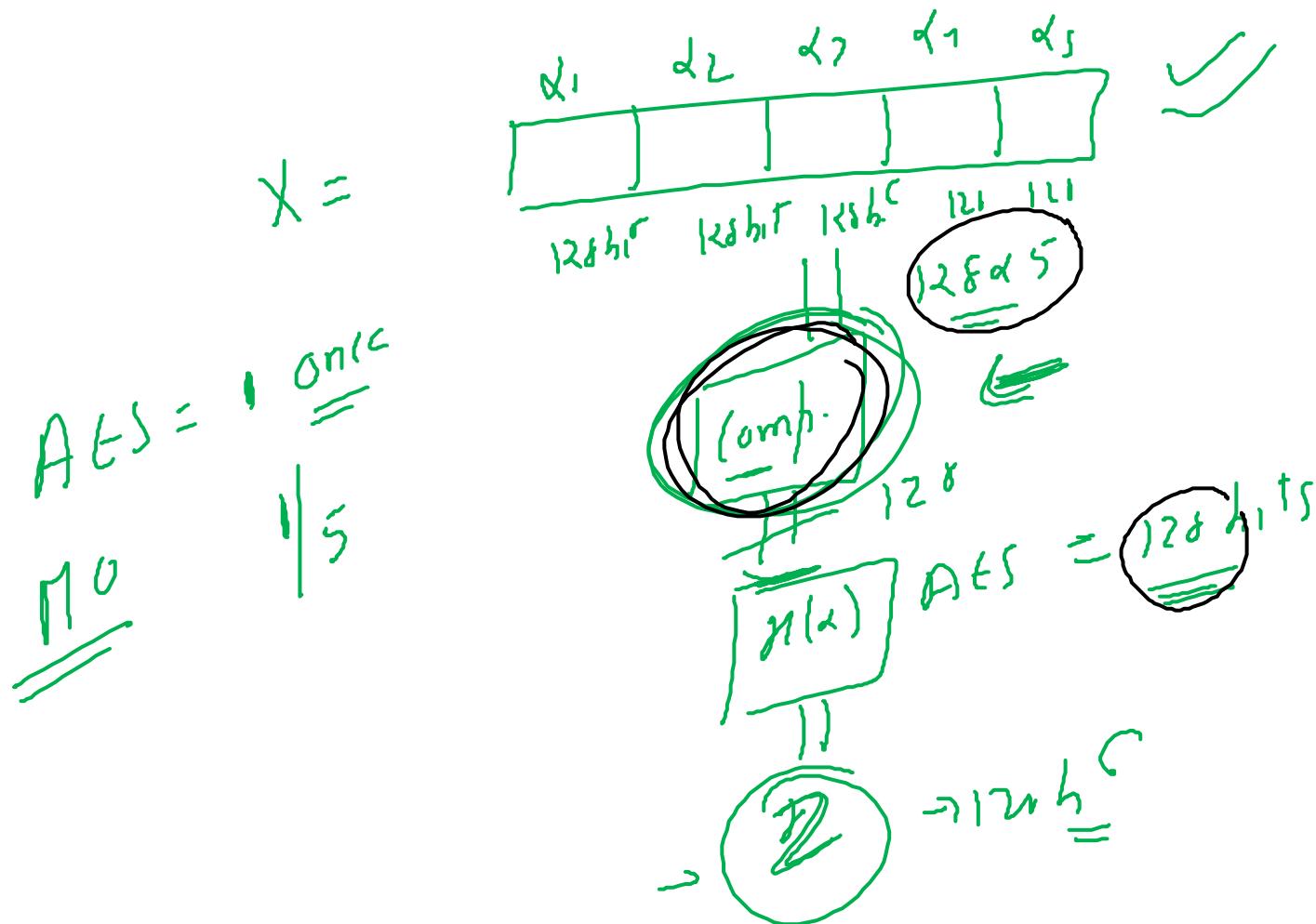
Bob

$$\begin{aligned}
 \text{Message} &= x \\
 \text{AES}_K(x) &= z \\
 (x, z)
 \end{aligned}$$

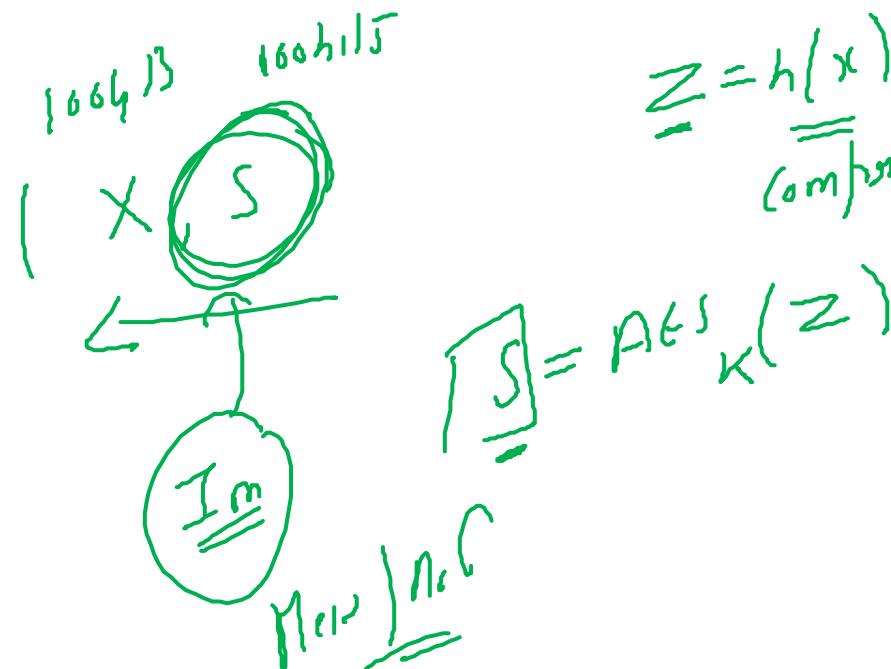
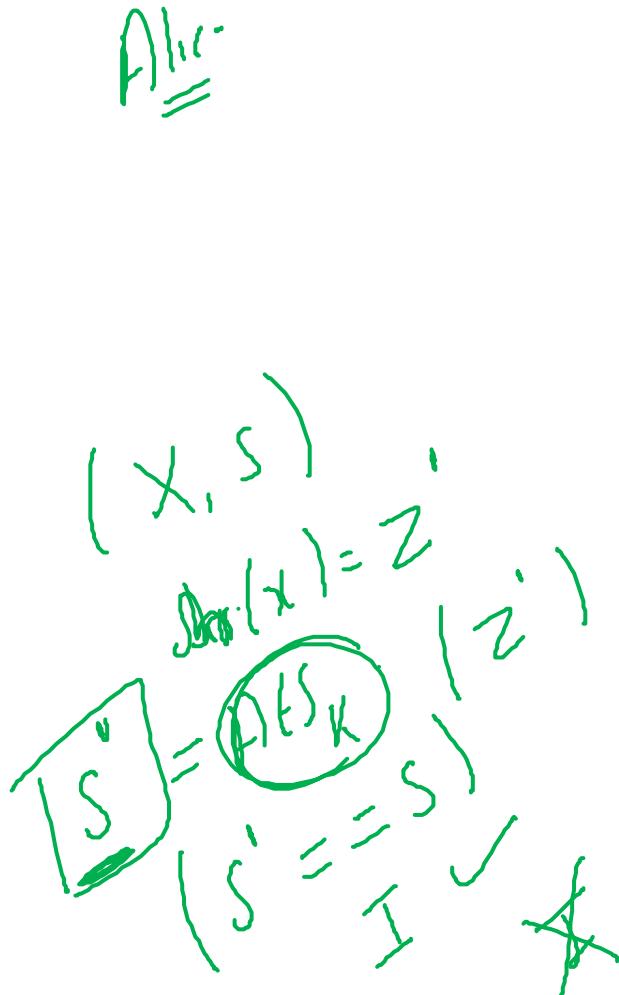
# Block Cipher based Hash Functions (2)



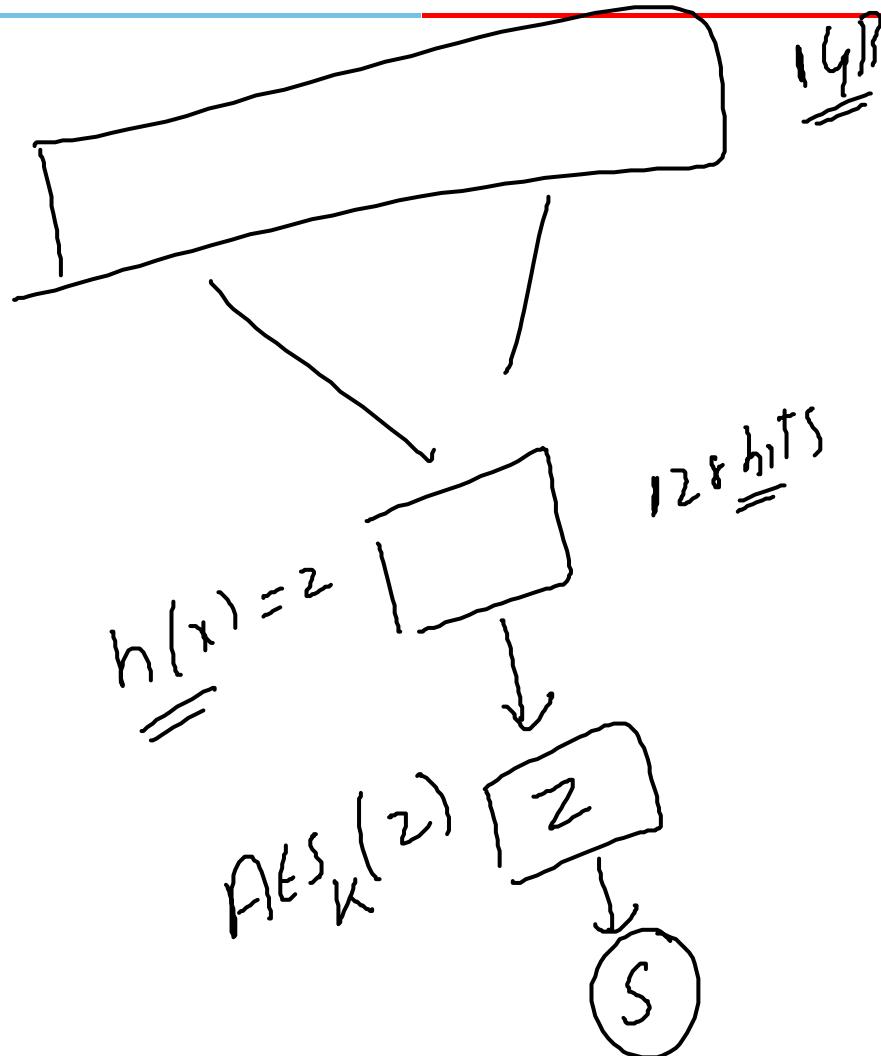
# Block Cipher based Hash Functions (3)



# Block Cipher based Hash Functions (4)



# Compress





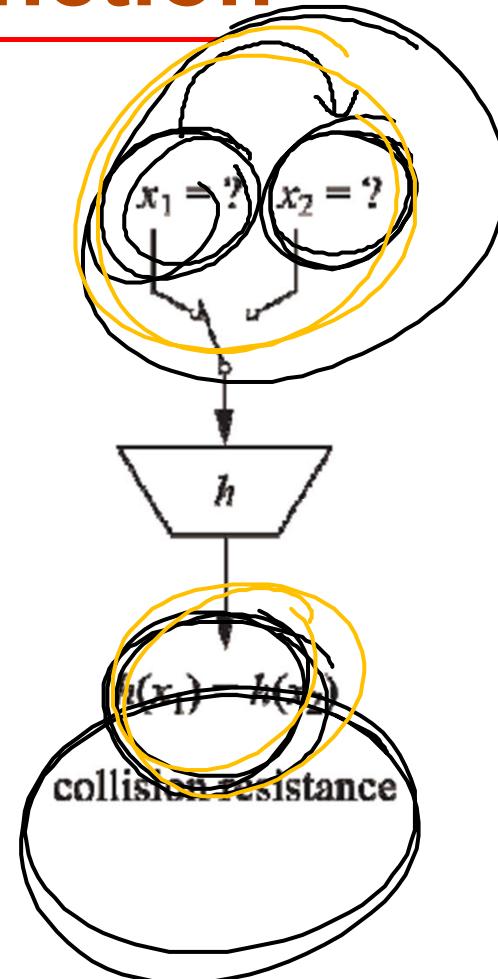
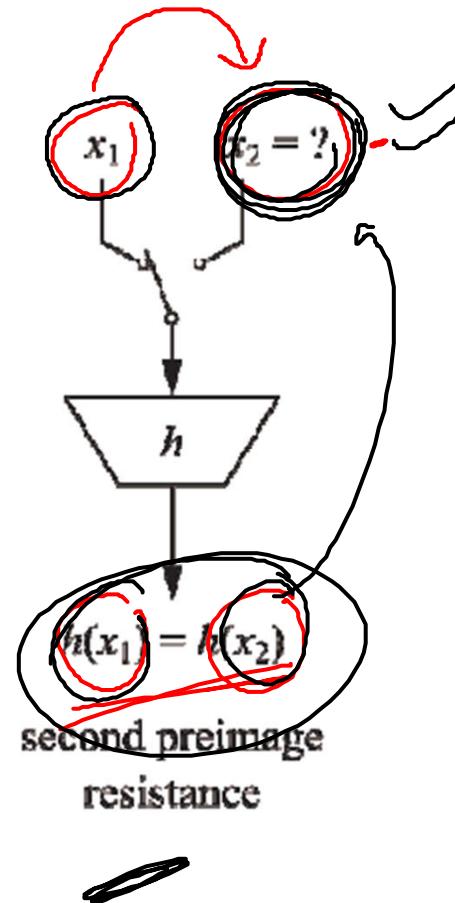
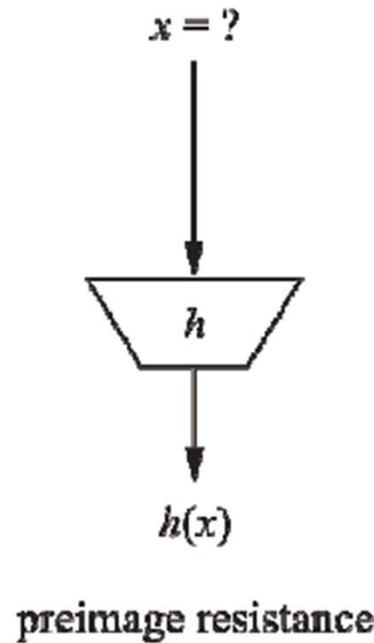
# Compress (2)

---

# Requirements of Hash Function

- ① Input  $\rightarrow$   $UL$
  - ②  $OIP \rightarrow FL$
  - ③  $E | Fast$  } Small change in  $\underline{IIP}$
- 
- ④ Pre image Reversal
  - ⑤ 2<sup>nd</sup> Pre image Reversal
  - ⑥ Collision Reversal

# Requirements of Hash Function



# Requirements of Hash Function



- Preimage resistance / One way function:

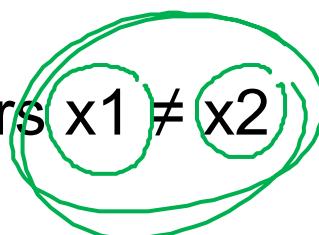
Given  $x$  it should be easy to calculate  $h(x)$  -> one way  
but given  $h(x)$  it should be computationally infeasible to  
compute  $x$

- Second preimage resistance / Weak Collision Resistance

Given  $x_1$ , and thus  $h(x_1)$ , it is computationally infeasible  
to find any  $x_2$  such that  $h(x_1) = h(x_2)$ .

- Collision resistance:

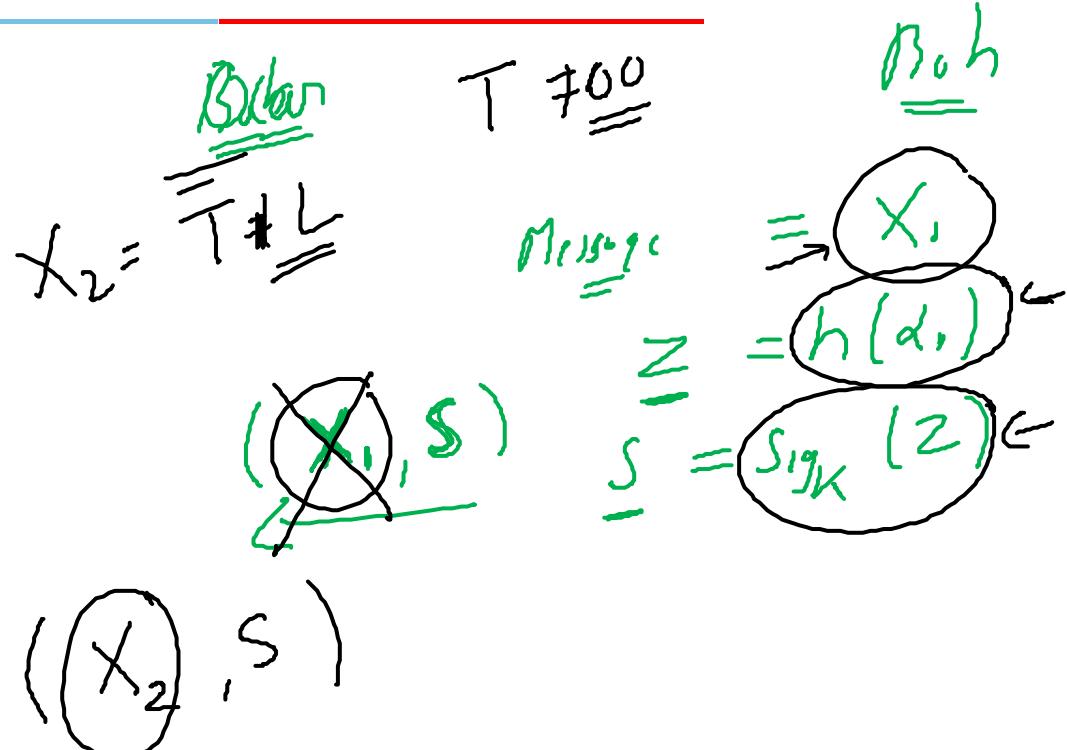
It is computationally infeasible to find any pairs  $x_1 \neq x_2$   
such that  $h(x_1) = h(x_2)$ .



## 2<sup>nd</sup> pre-image attack ✓

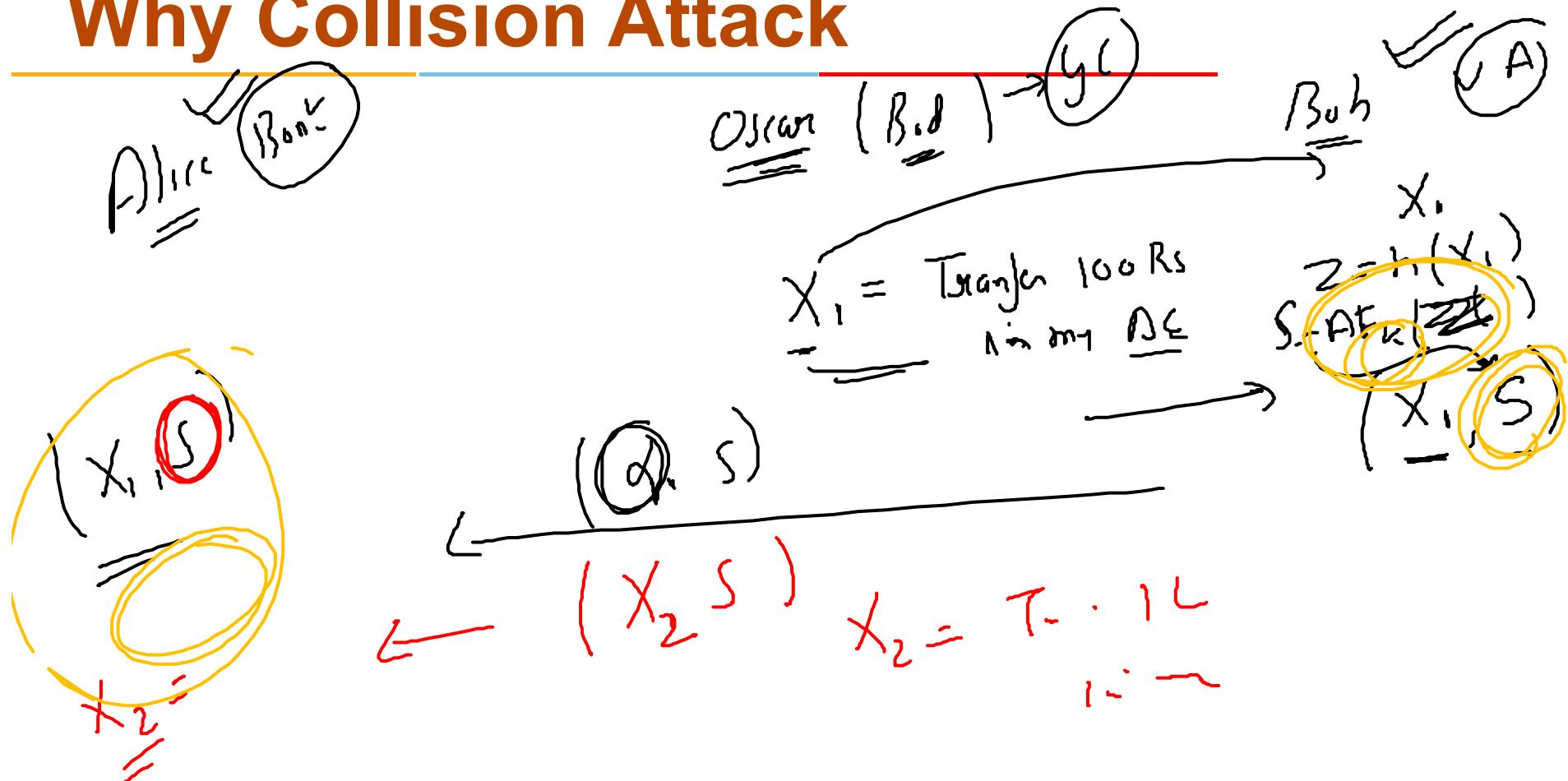
Alice

$$\begin{aligned} & (x_2, s) \\ z' &= h(x_2) \\ z' &= \text{Sig}_K(z') \\ & (s' == s) \end{aligned}$$

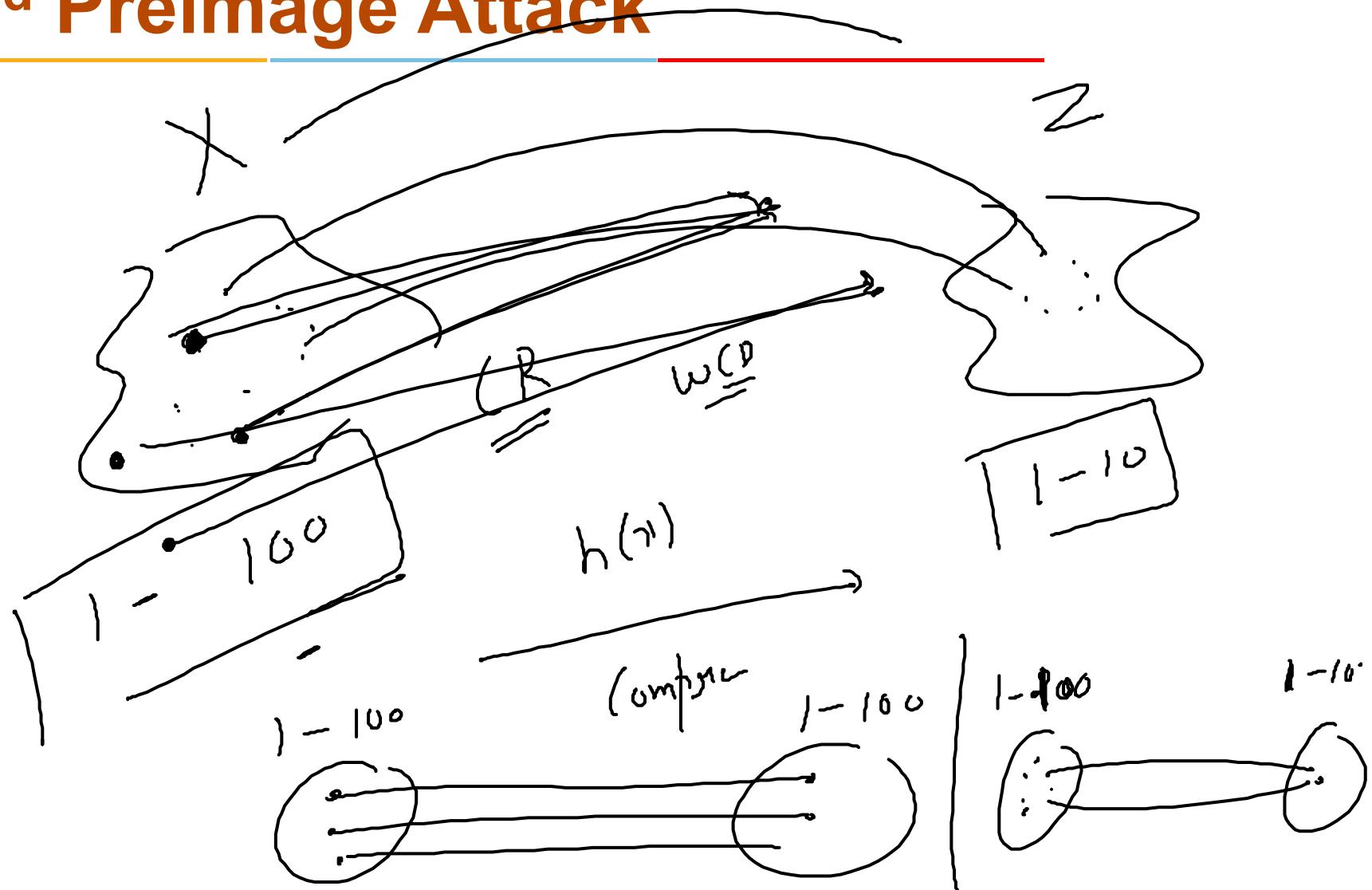




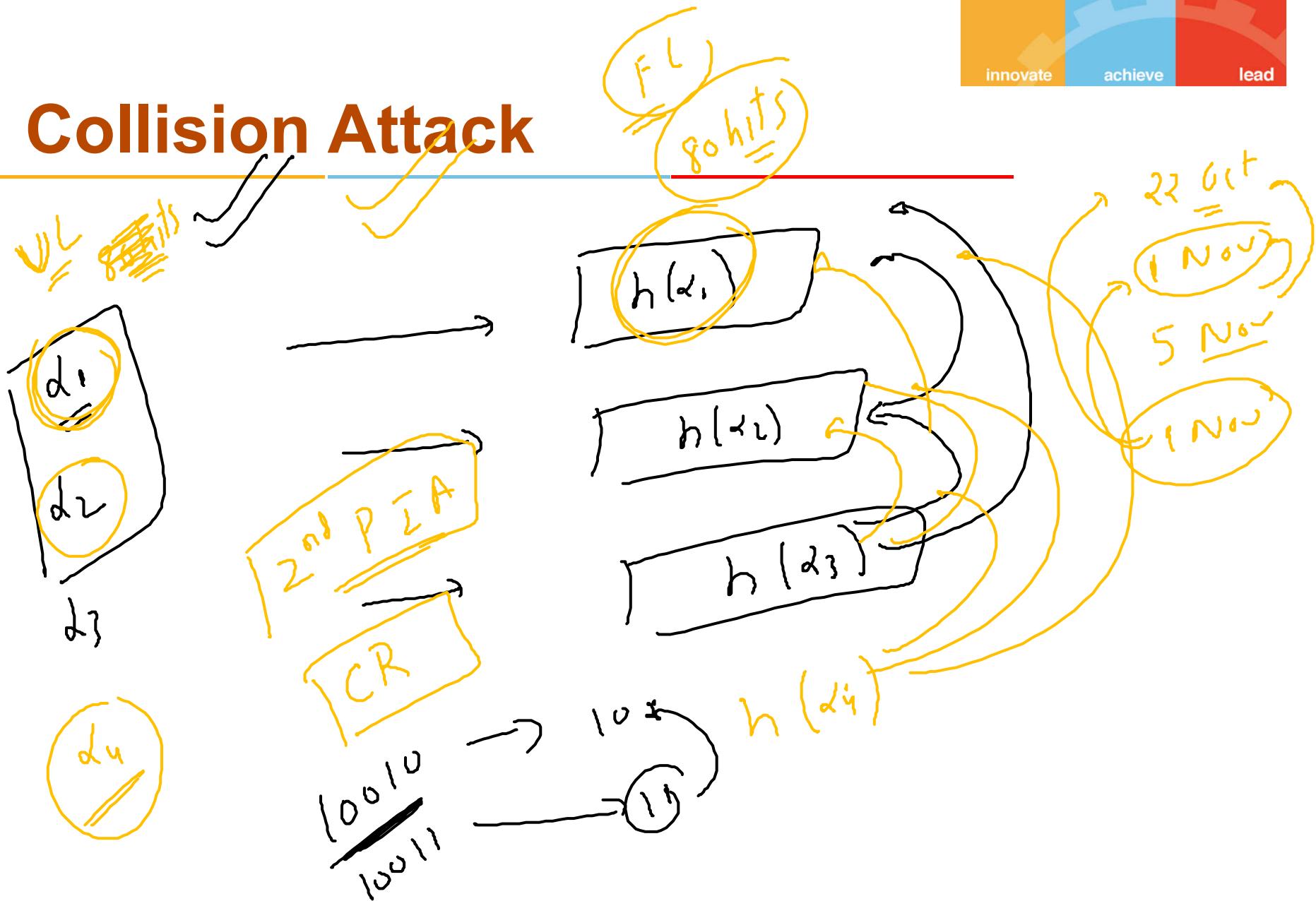
# Why Collision Attack



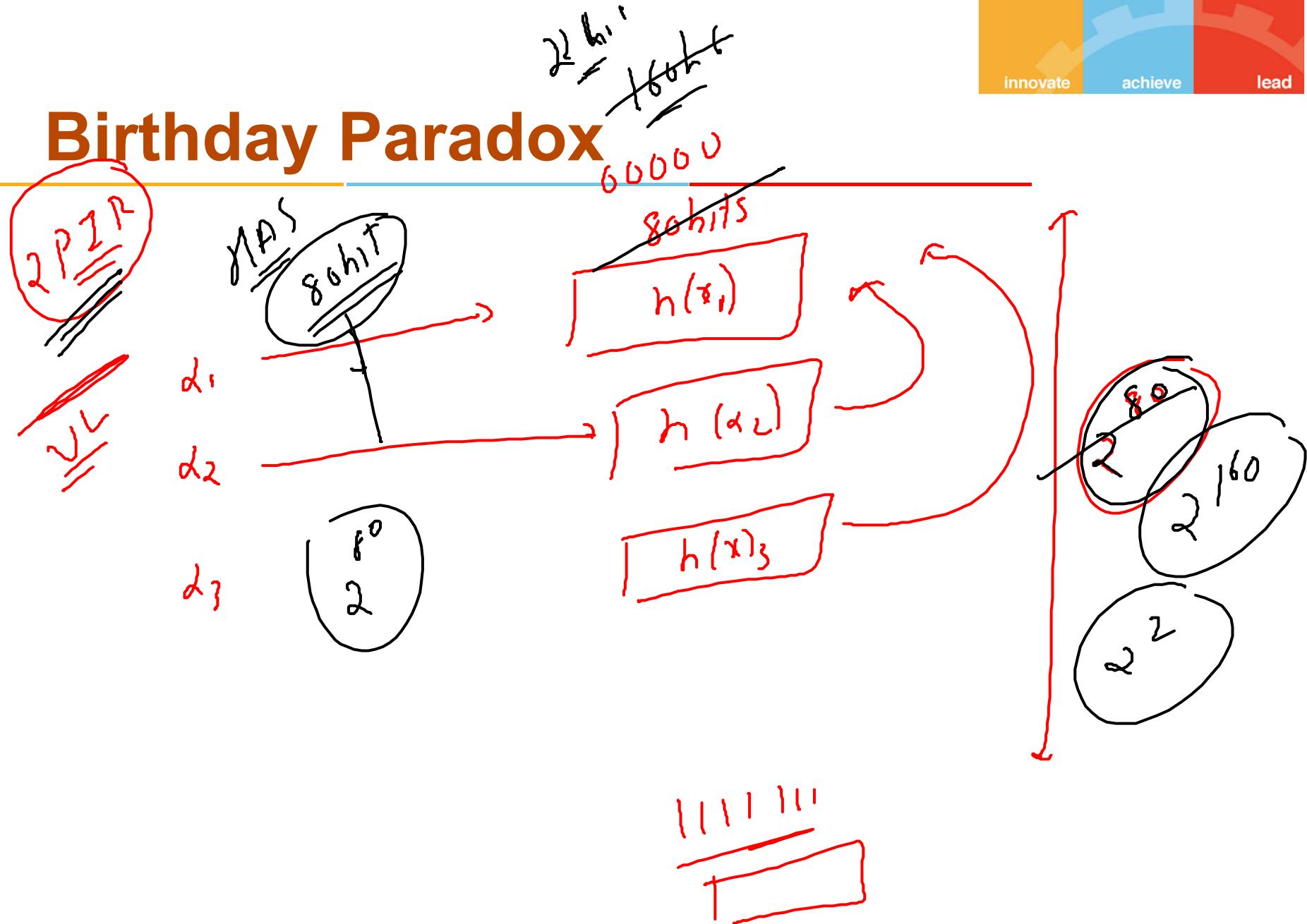
## 2<sup>nd</sup> Preimage Attack



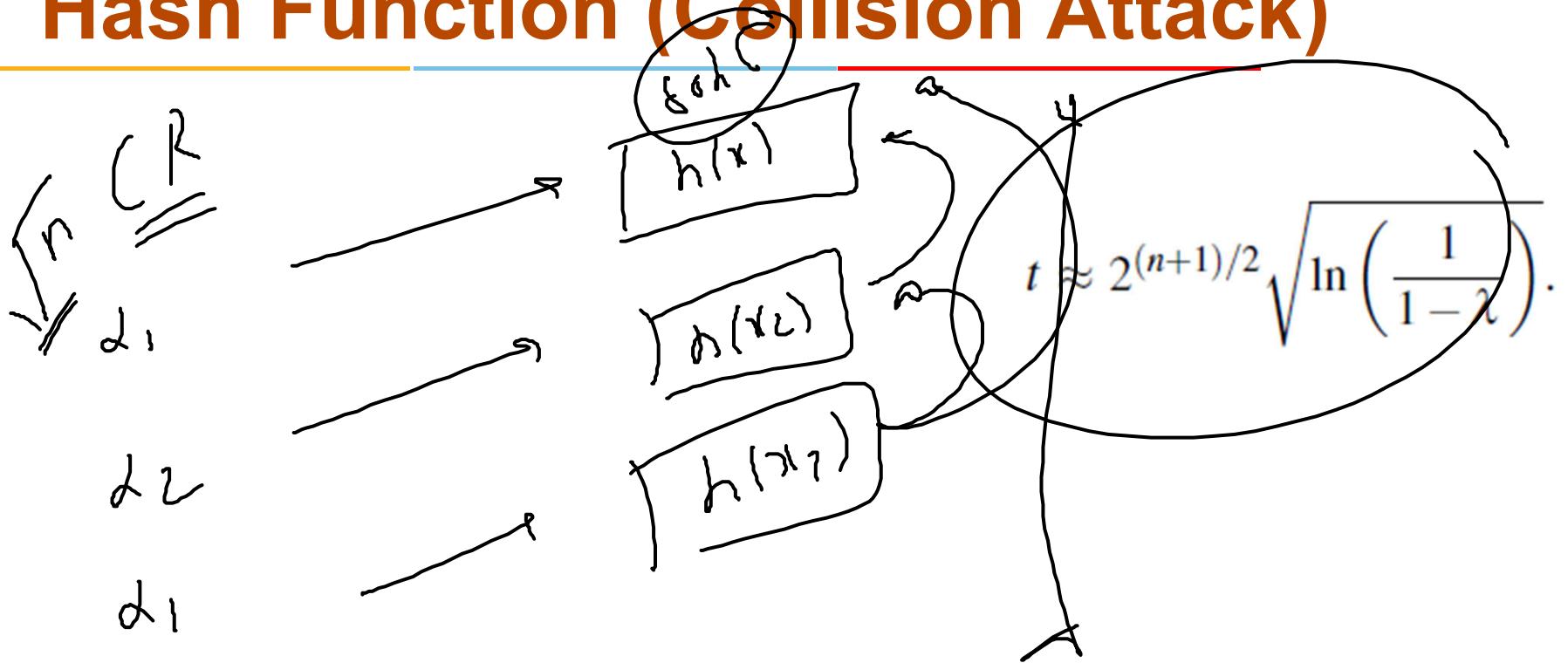
# Collision Attack



# Birthday Paradox



# Hash Function (Collision Attack)



$\lambda$	Hash output length				
	128 bit	160 bit	256 bit	384 bit	512 bit
0.5	$2^{65}$	$2^{81}$	$2^{129}$	$2^{193}$	$2^{257}$
0.9	$2^{67}$	$2^{82}$	$2^{130}$	$2^{194}$	$2^{258}$



# Hash Function from Block Cipher

Davies Meyer:  $H_i = H_{i-1} \oplus E_{x_i}(H_{i-1})$

Matyas-Meyer-Oseas:  $H_i = E_{g(H_i - I)}(x_i) \oplus x_i$



# Hash Function from Block Cipher

Miyaguchi-Preneel:  $H_i = E_{g(H_{i-1})}(x_i) \oplus H_{i-1} \oplus x_i$



# Hash Function from Block Cipher

Hirose Construction: The final output is  $H_i \parallel G_i$

$$H_i = E_{H_{i-1}} \cdot x_i (G_{i-1} \oplus c) \oplus (G_{i-1} \oplus c)$$

$$G_i = E_{H_{i-1}} \cdot x_i (G_{i-1}) \oplus G_{i-1}$$



# Hash Construction

---



---

Thanks!!!  
Queries?



**BITS** Pilani  
K K Birla Goa Campus

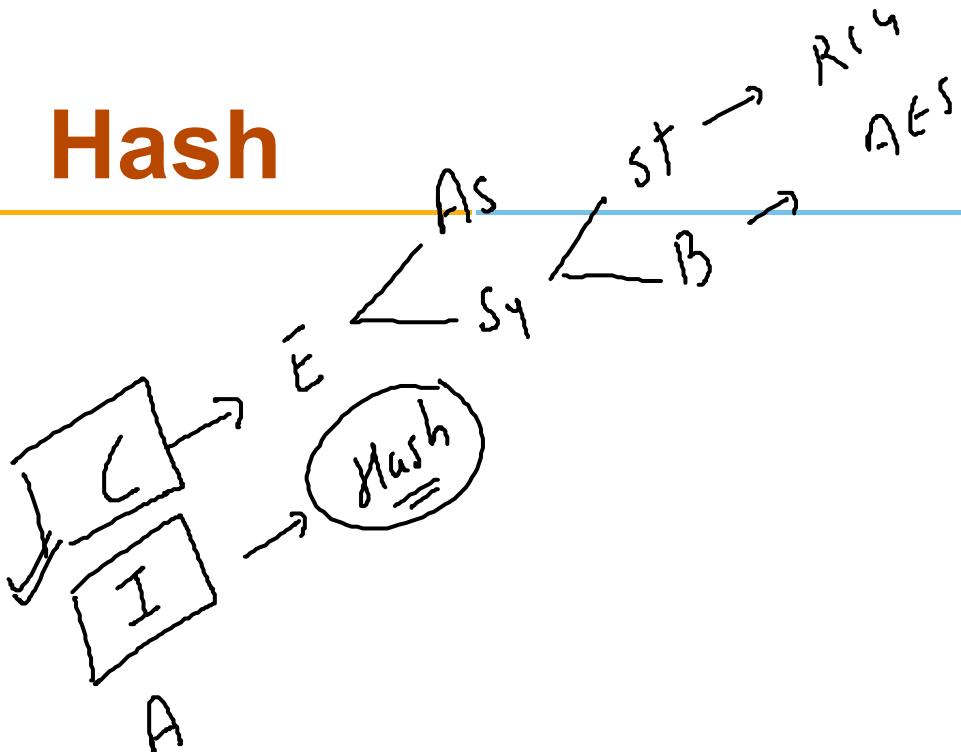
# Network Security

## SS ZG513

Hemant Rathore  
Department of Computer Science and Information Systems



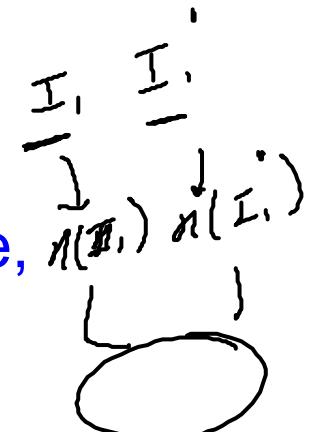
# Hash



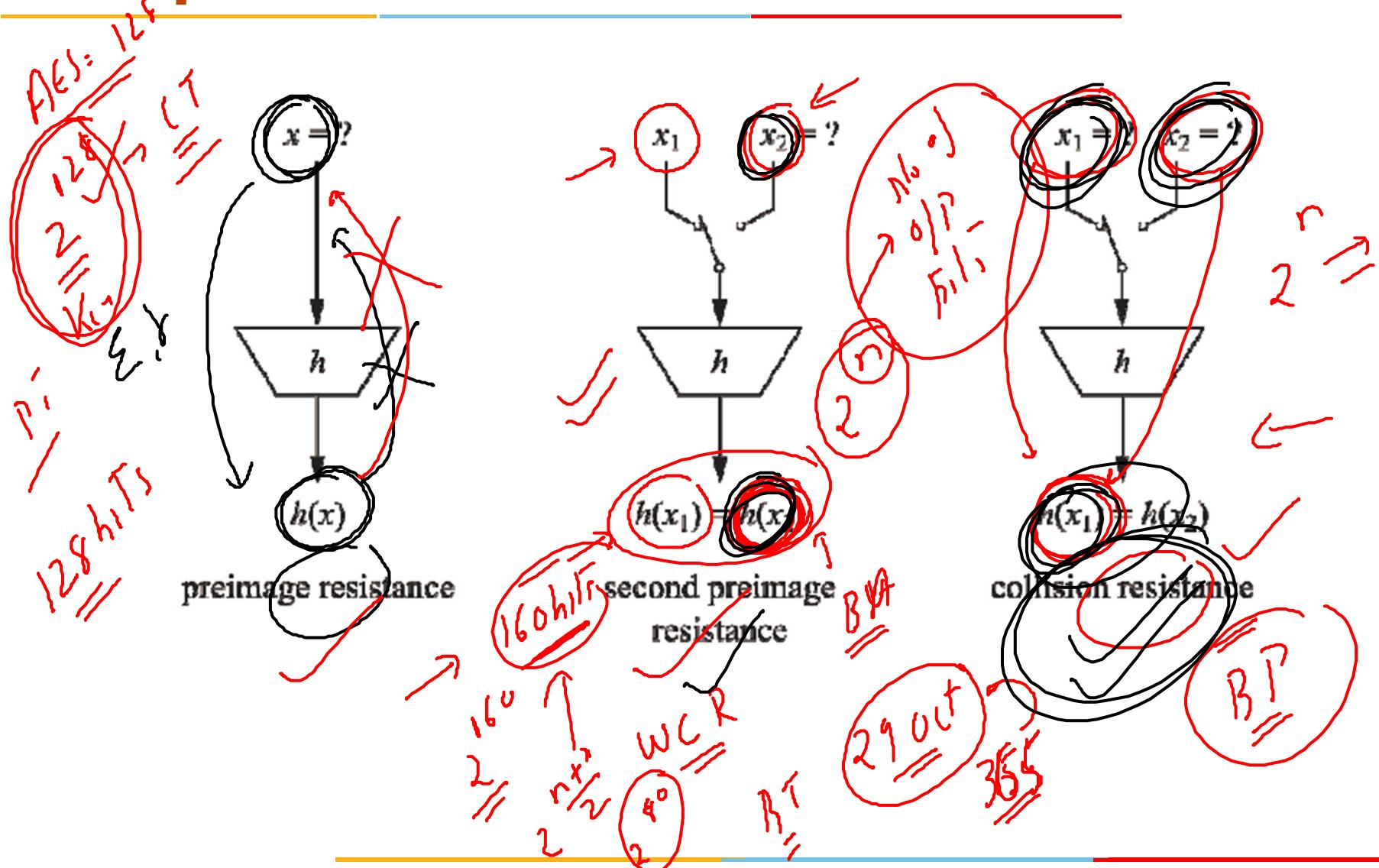
# Hash Function

$$\text{Input} = \text{arbitrary length string}$$

- Principle: Takes a string of arbitrary length and maps it to a fixed-length output, referred to as hash value.
- Practical Requirements: Arbitrary message size, Fixed output length, Finger print should be highly sensitive, Efficiency.
- Security Requirements: Pre-image resistance, Second pre-image resistance, collision resistance, Non-correlation, Near collision resistance, partial-image resistance.



# Requirements of Hash Function





# Requirements of Hash Function

- Preimage resistance / One way function:

Given  $x$  it should be easy to calculate  $h(x)$  -> one way  
but given  $h(x)$  it should be computationally infeasible to  
compute  $x$

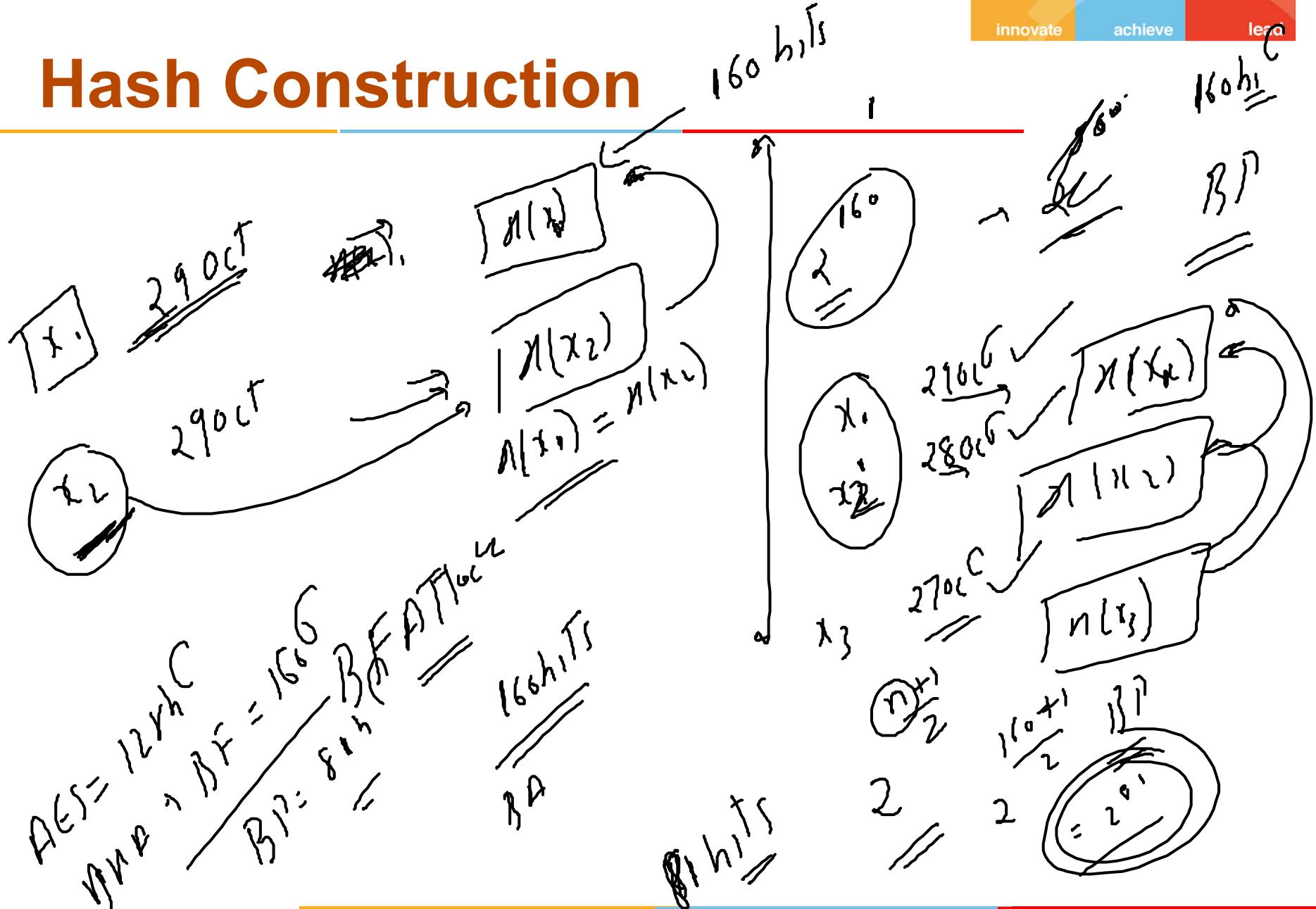
- Second preimage resistance / Weak Collision Resistance

Given  $x_1$ , and thus  $h(x_1)$ , it is computationally infeasible  
to find any  $x_2$  such that  $h(x_1) = h(x_2)$ .

- Collision resistance:

It is computationally infeasible to find any pairs  $x_1 \neq x_2$   
such that  $h(x_1) = h(x_2)$ .

# Hash Construction

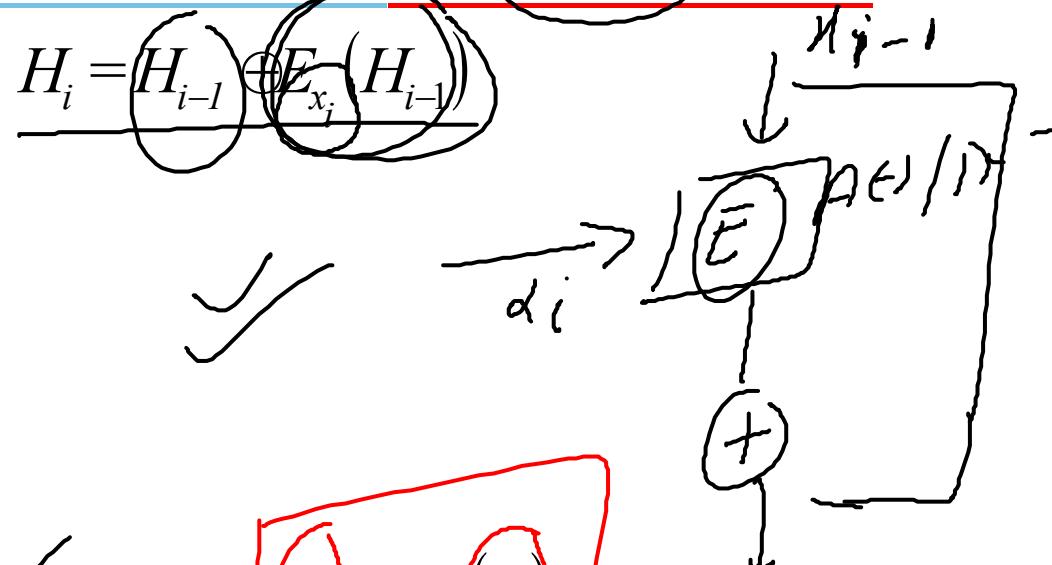


# Types of Hash Function

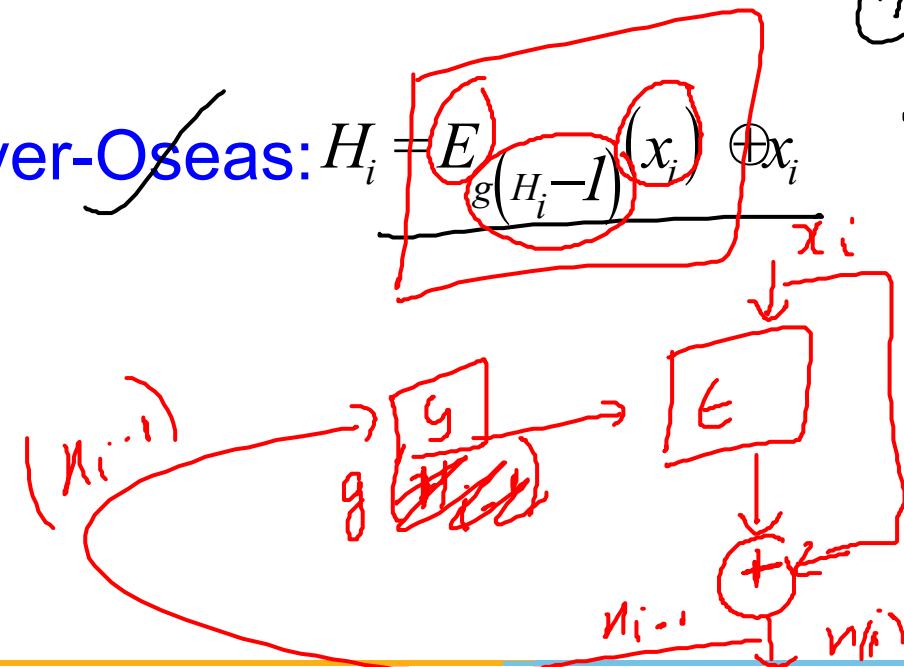
- Block cipher based hash functions:
  - Use block cipher such as AES to construct hash function.
- Dedicated Hash functions:
  - Specifically designed to serve as a hash function.
  - Fact: Any compression function 'f' which is collision resistant can be extended to a collision resistant hash function.
  - Merkle-Damgard Construction: Blocks are processed sequentially by the hash function, which has a compression function at its heart.

# Hash Function from Block Cipher

Davies Meyer:



Matyas-Meyer-Oseas:





# Hash Function from Block Cipher

Miyaguchi-Preneel:

$$H_i = E_{g(H_i - I)}(x_i) \oplus H_{i-1} \oplus x_i$$



# Hash Function from Block Cipher

Hirose Construction: The final output is  $H_i \parallel G_i$

$$H_i = E_{H_{i-1} \cdot x_i} (G_{i-1} \oplus c) \oplus (G_{i-1} \oplus c)$$

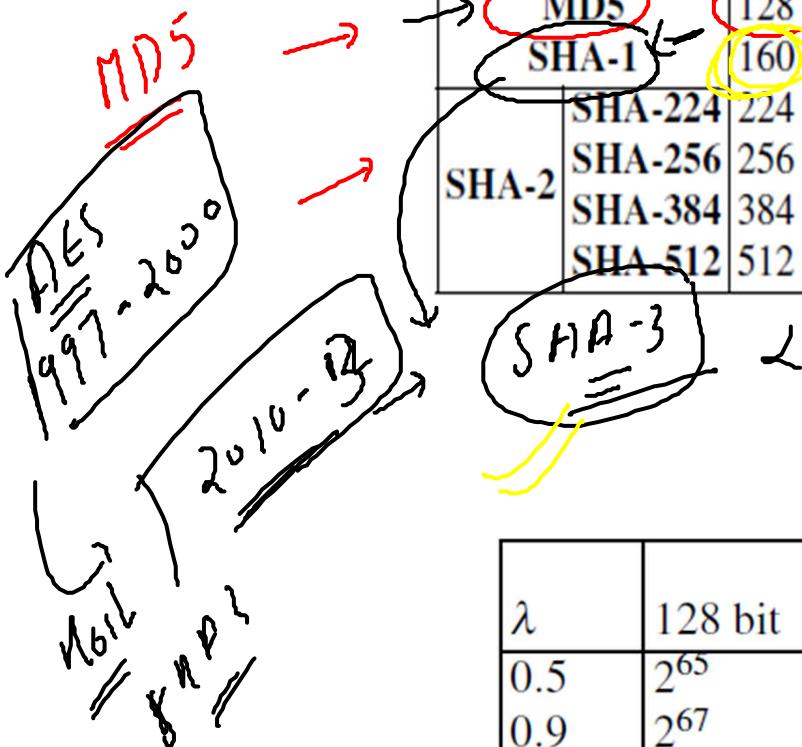
$$G_i = E_{H_{i-1} \cdot x_i} (G_{i-1}) \oplus G_{i-1}$$

# Dedicated Hash Function

- MD4 family of hash functions: ~~FO ALI~~

$$AES = \frac{128}{10 \text{ HR}}$$

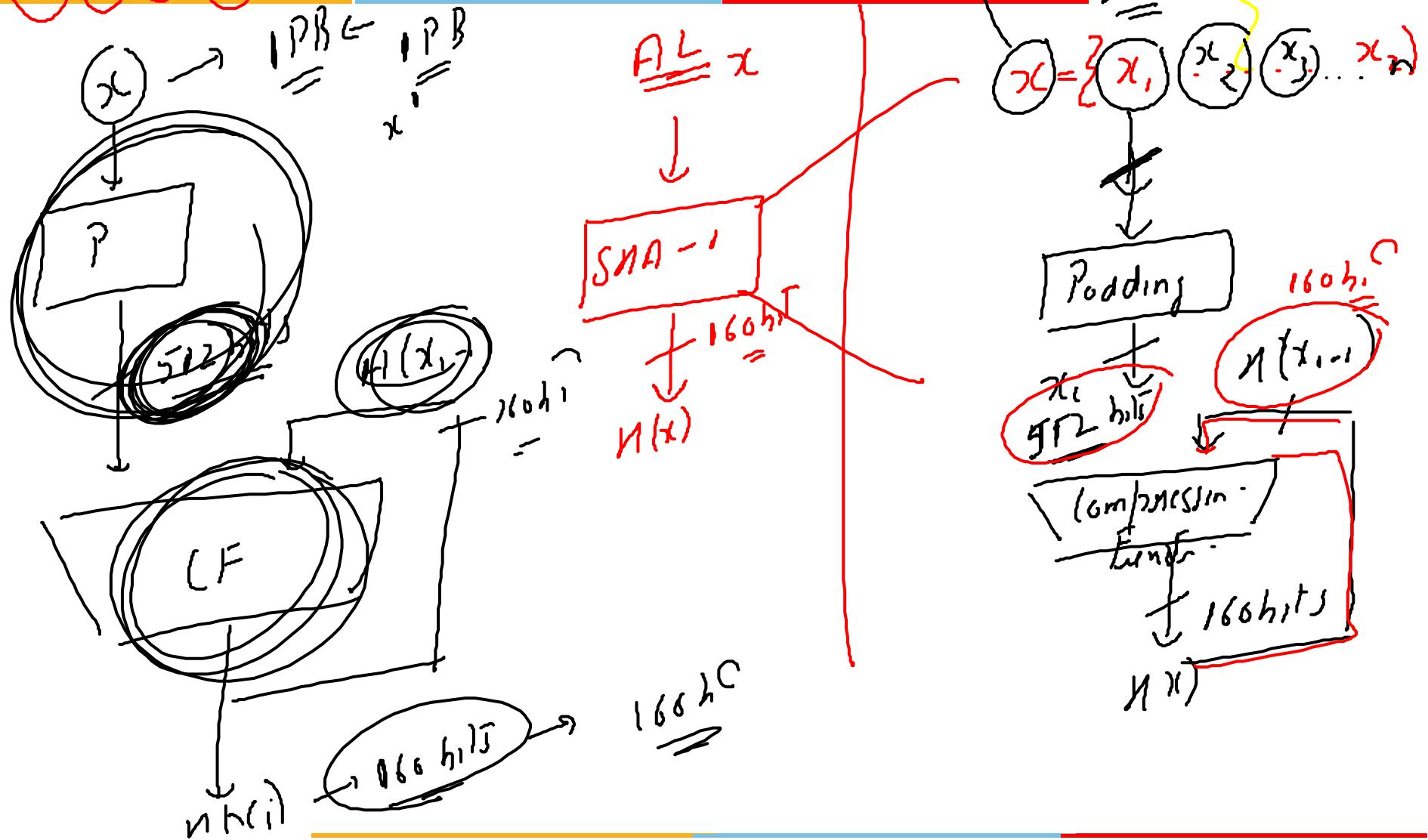
Algorithm	Output [bit]	Input [bit]	No. of rounds	Collisions found
MD5	128	512	64	yes
SHA-1	160	512	80	not yet
SHA-224	224	512	64	no
SHA-256	256	512	64	no
SHA-384	384	1024	80	no
SHA-512	512	1024	80	no



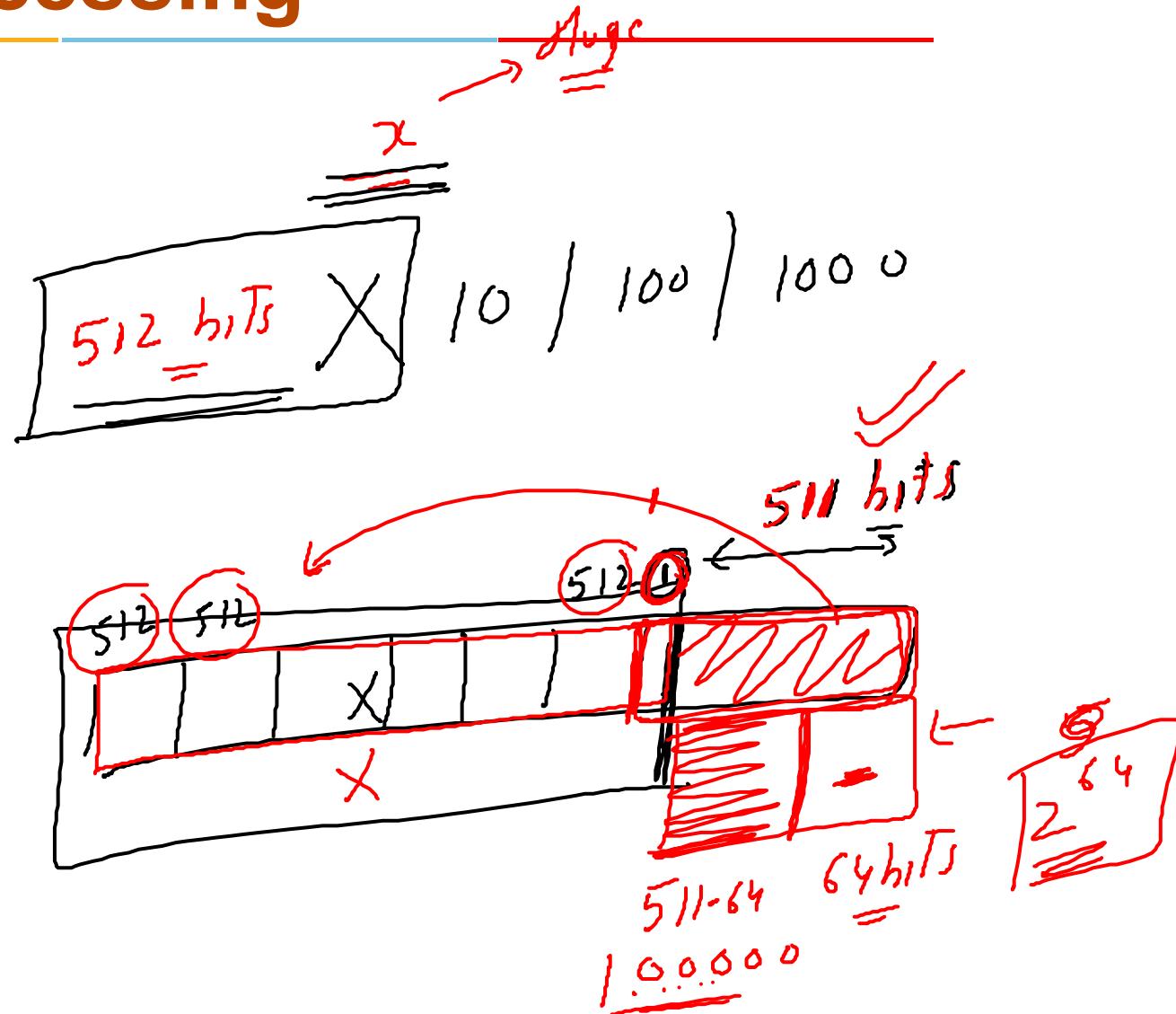
$$\begin{aligned} & 2^{8+1} \\ & \frac{2^8}{2} \\ & 2 \quad 65 \\ & \frac{160+1}{2} \\ & 2 \end{aligned}$$

$\lambda$	Hash output length				
	128 bit	160 bit	256 bit	384 bit	512 bit
0.5	$2^{65}$	$2^{81}$	$2^{129}$	$2^{193}$	$2^{257}$
0.9	$2^{67}$	$2^{82}$	$2^{130}$	$2^{194}$	$2^{258}$

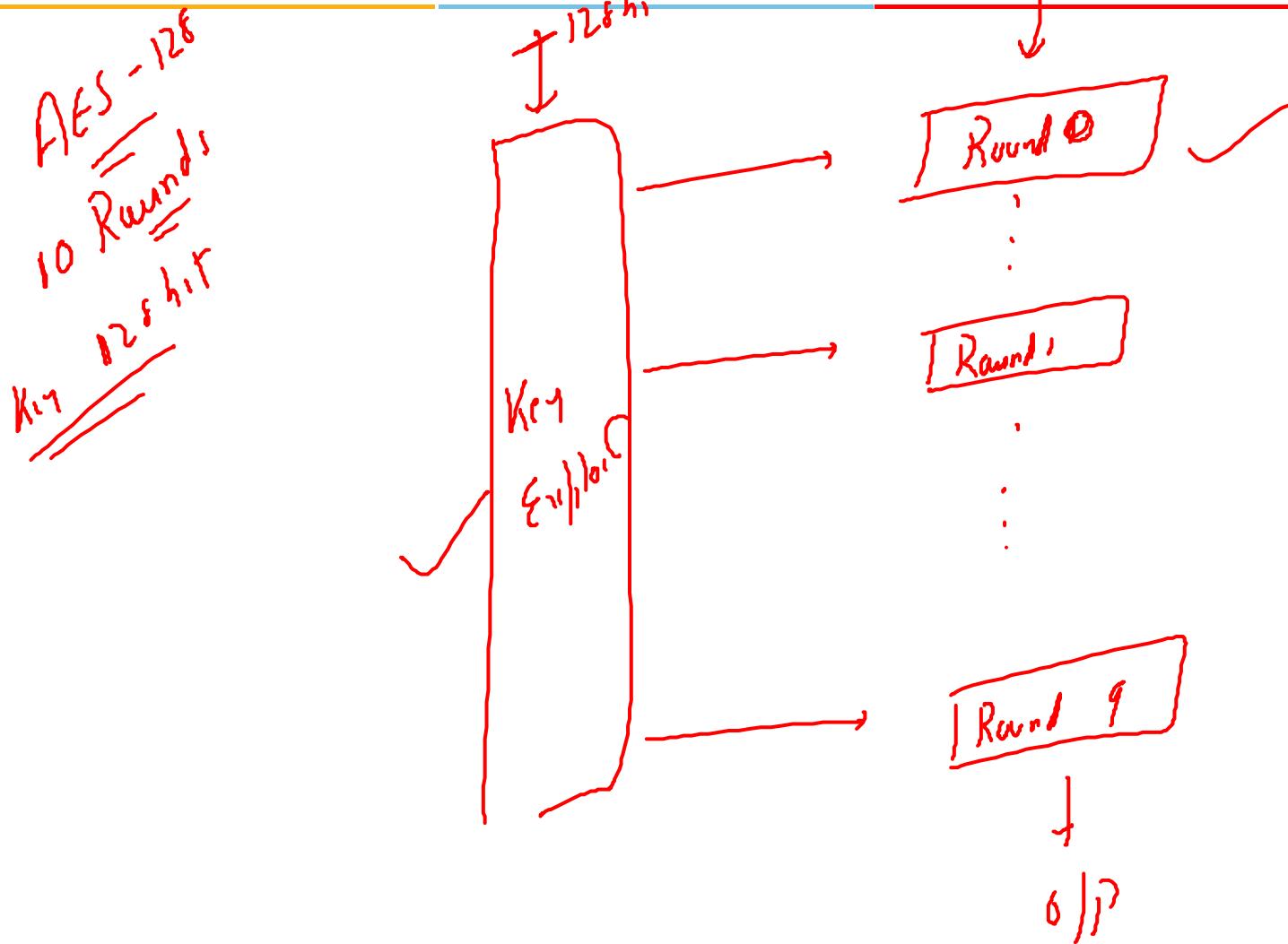
# SHA-1 High Level View



# Pre-processing

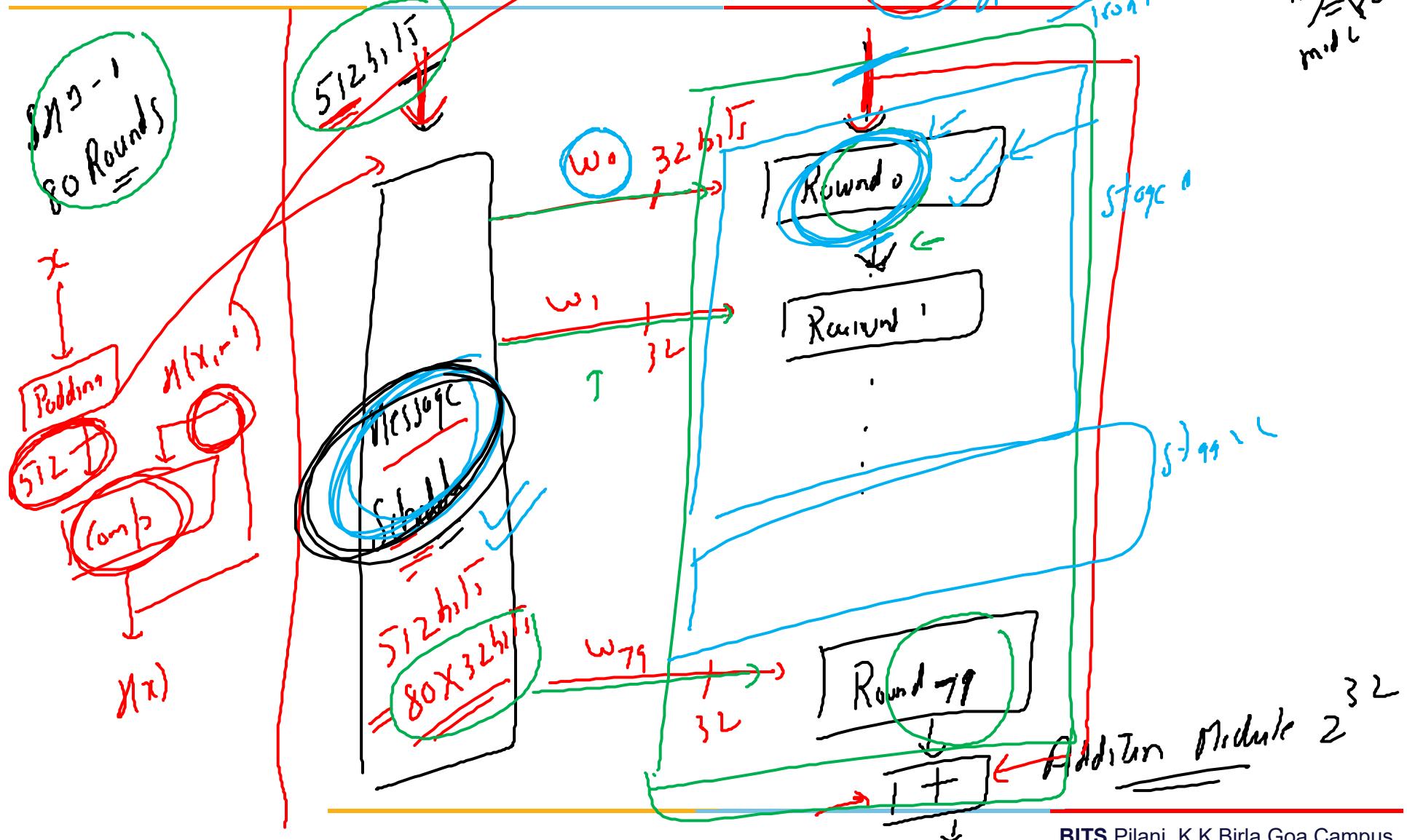
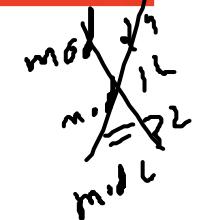


# Recall Block Cipher

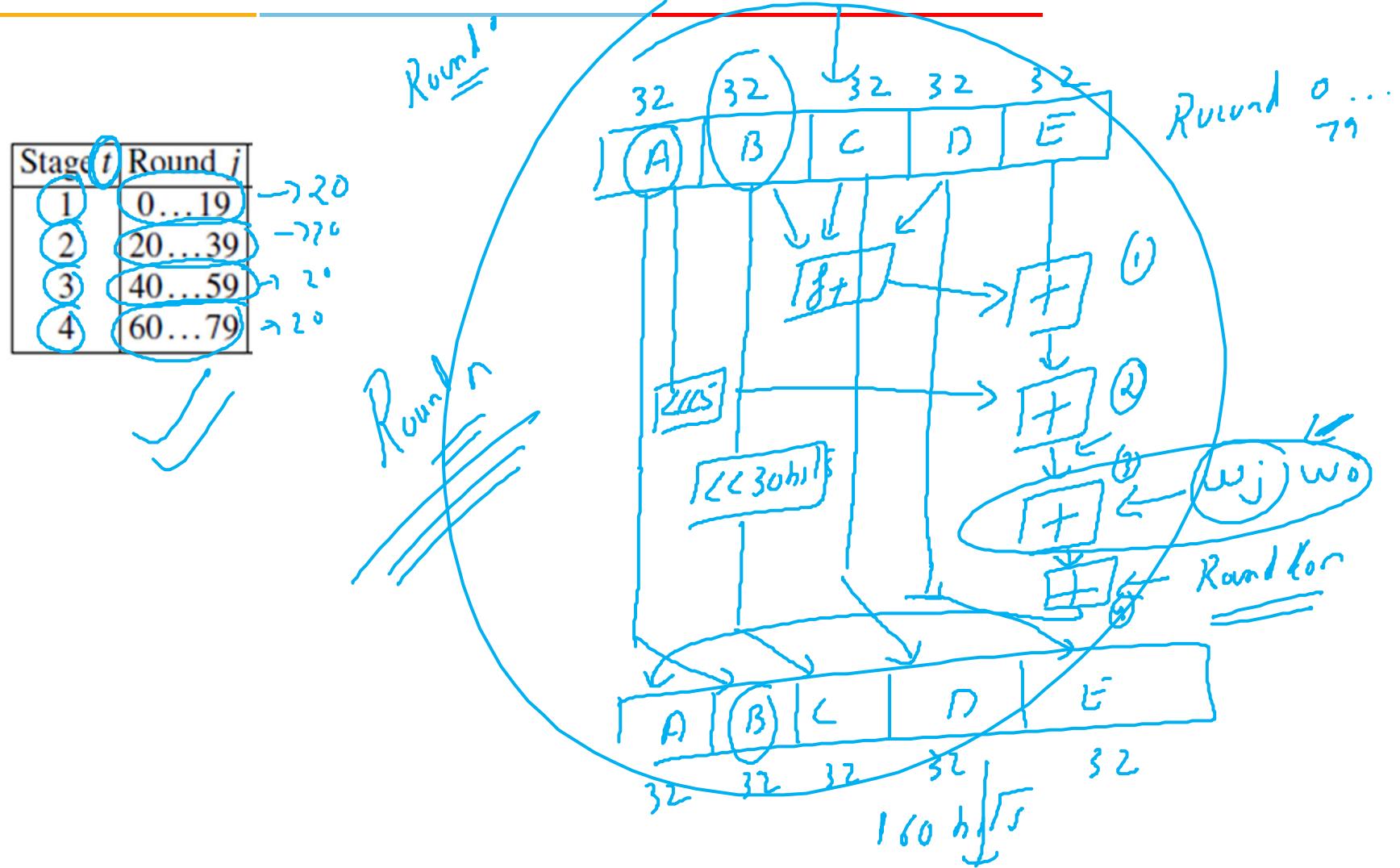


# SHA-1 (Block Diagram)

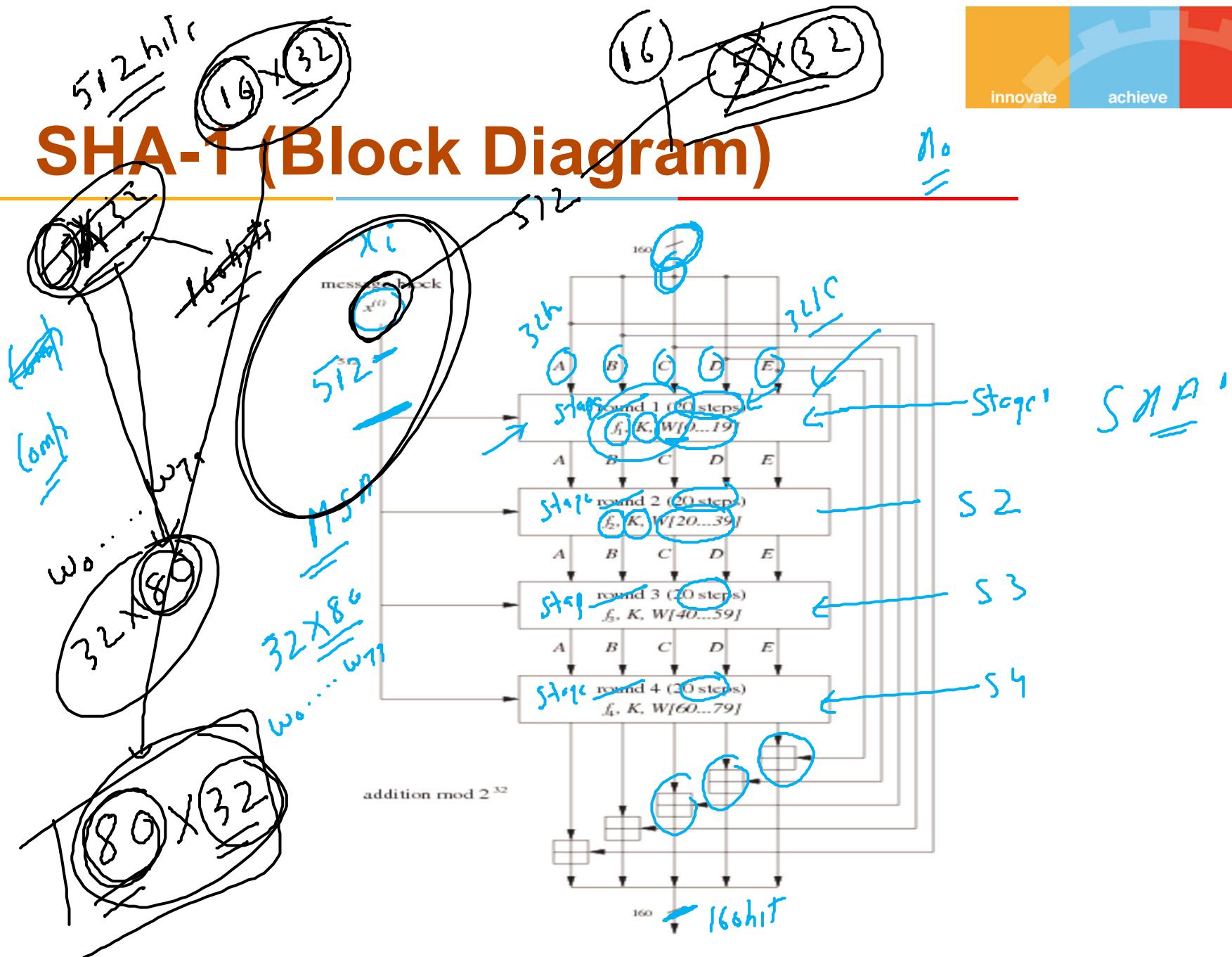
$\text{Init} \Rightarrow K_0$



# SHA-1



# SHA-1 (Block Diagram)





# SHA-I

- Preprocessing (Padding) of the message.
- Fiestel block cipher, based on Merkel-Damgard construction.
- Message block acts as a key and input is the previous hash value ( $H_{i-1}$ ).
- Message is divided into 512 bit blocks.
- Each 512-bit blocks are further subdivided into 16 words of size of 32 bits.

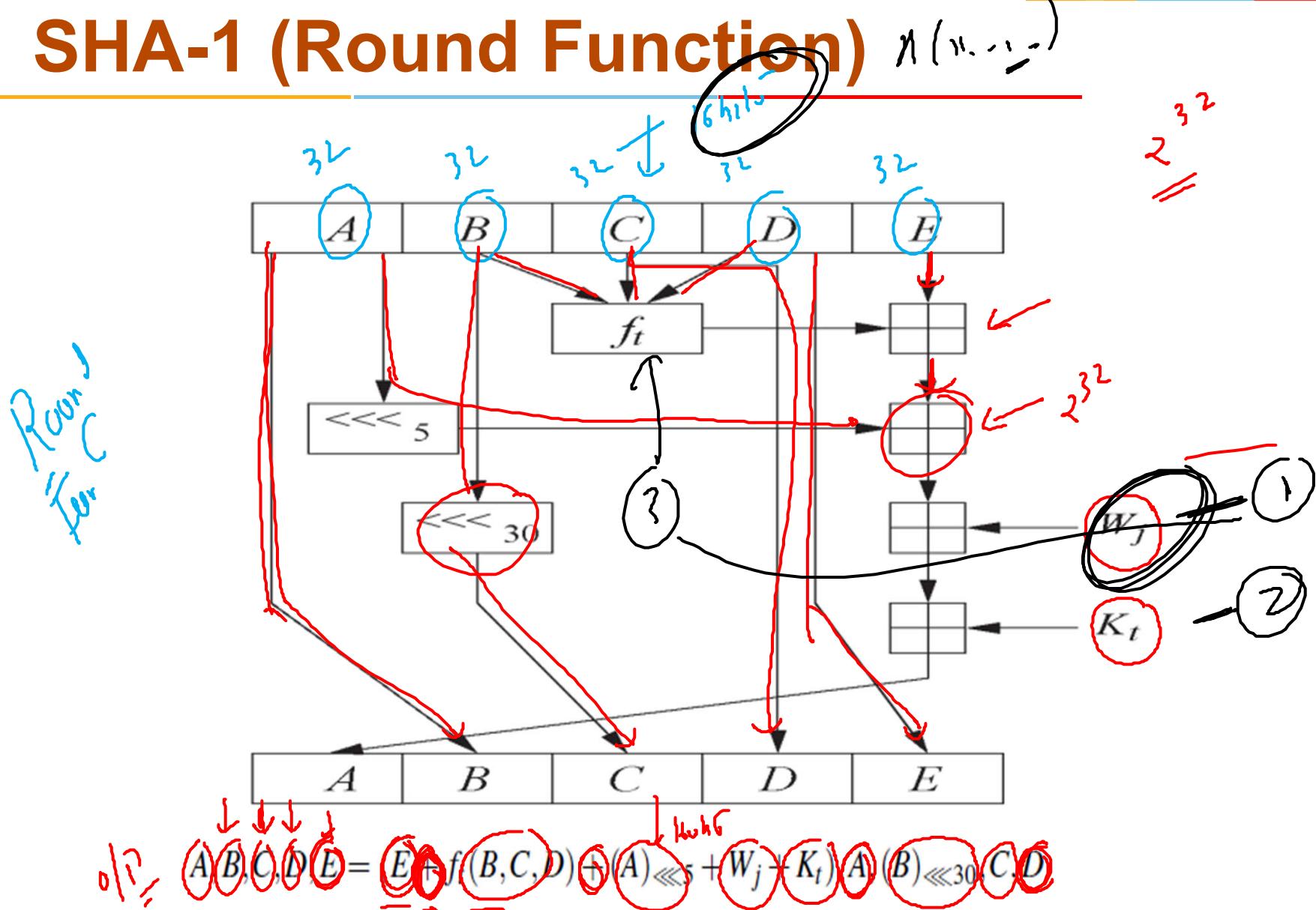


# SHA-1 (Round Function)

---

$$A, B, C, D, E = (E + f_t(B, C, D) + (A)_{\ll 5} + W_j + K_t), A, (B)_{\ll 30}, C, D$$

# SHA-1 (Round Function)



# SHA-1 (Round Function)

- Initial  $H_0$ :

$$A = H_0^{(0)} = 67452301$$

$$B = H_0^{(1)} = \text{EFCDAB89}$$

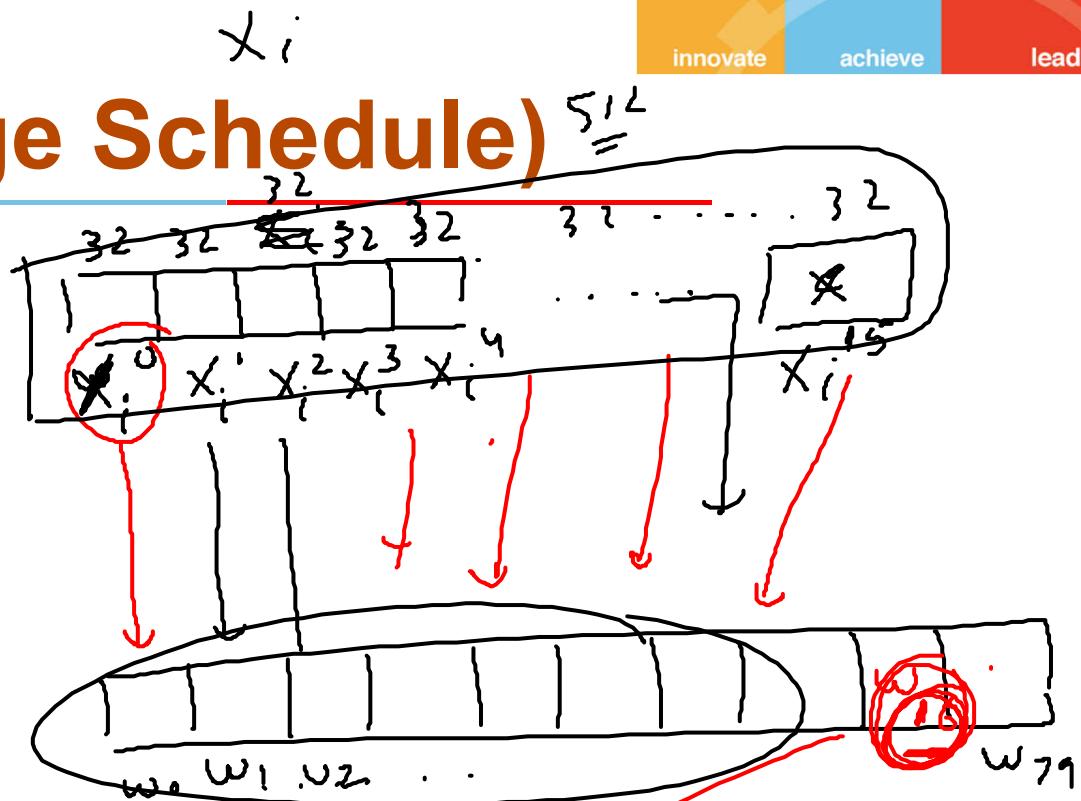
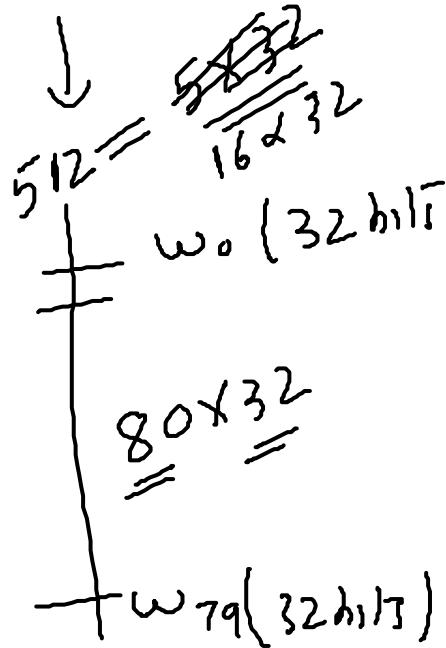
$$C = H_0^{(2)} = 98\text{BADCFE}$$

$$D = H_0^{(3)} = 10325476$$

$$E = H_0^{(4)} = \text{C3D2E1F0.}$$

Stage $t$	Round $j$	Constant $K_t$	Function $f_t$
1	0 ... 19	$K_1 = 5A827999$	$f_1(B, C, D) = (B \wedge C) \vee (\bar{B} \wedge D)$
2	20 ... 39	$K_2 = 6ED9EBA1$	$f_2(B, C, D) = B \oplus \bar{C} \oplus \bar{D}$
3	40 ... 59	$K_3 = 8F1BBCDC$	$f_3(B, C, D) = (B \wedge C) \vee (\bar{B} \wedge D) \vee (C \wedge D)$
4	60 ... 79	$K_4 = \text{CA62C1D6}$	$f_4(B, C, D) = B \oplus C \oplus D$

# SHA-1 (Message Schedule)



- 32-bit 80 words  $W_j$  are computed from the 512-bit message block for each round as follows:

$$W_j = \begin{cases} x_i^{(j)} & 0 \leq j \leq 15 \\ (W_{j-16} \oplus W_{j-14} \oplus W_{j-8} \oplus W_{j-3}) \ll 1 & 16 \leq j \leq 79, \end{cases}$$

Annotations in red highlight the addition of previous words ( $\oplus$ ) and the left shift operation ( $\ll 1$ ).

# SHA-1 (Message Schedule)

~~$w_{16}$~~      $\underline{\underline{w_{16}}}$

- 32-bit 80 words  $W_j$  are computed from the 512-bit message block for each round as follows:

$$W_j = \begin{cases} x_i^{(j)} \\ (W_{j-16} \oplus W_{j-14} \oplus W_{j-8} \oplus W_{j-3}) \lll 1 \end{cases} \quad \begin{array}{l} 0 \leq j \leq 15 \\ 16 \leq j \leq 79, \end{array}$$

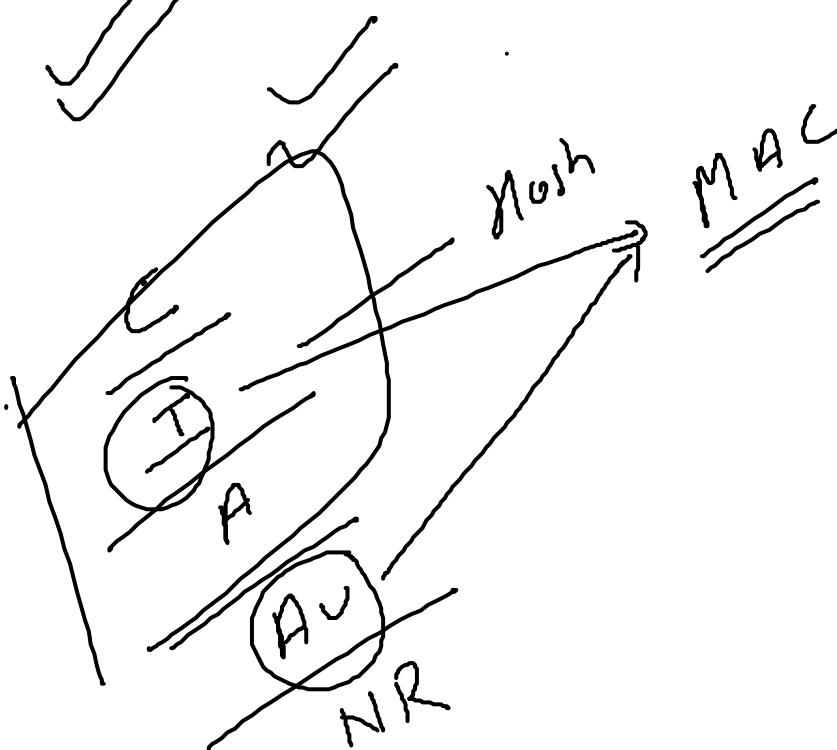


---

Thanks!!!  
Queries?

# Today's Agenda

- Message Authentication code (MAC)
- MAC's from hash function (HMAC)



D SK E

MAC  
S  
u





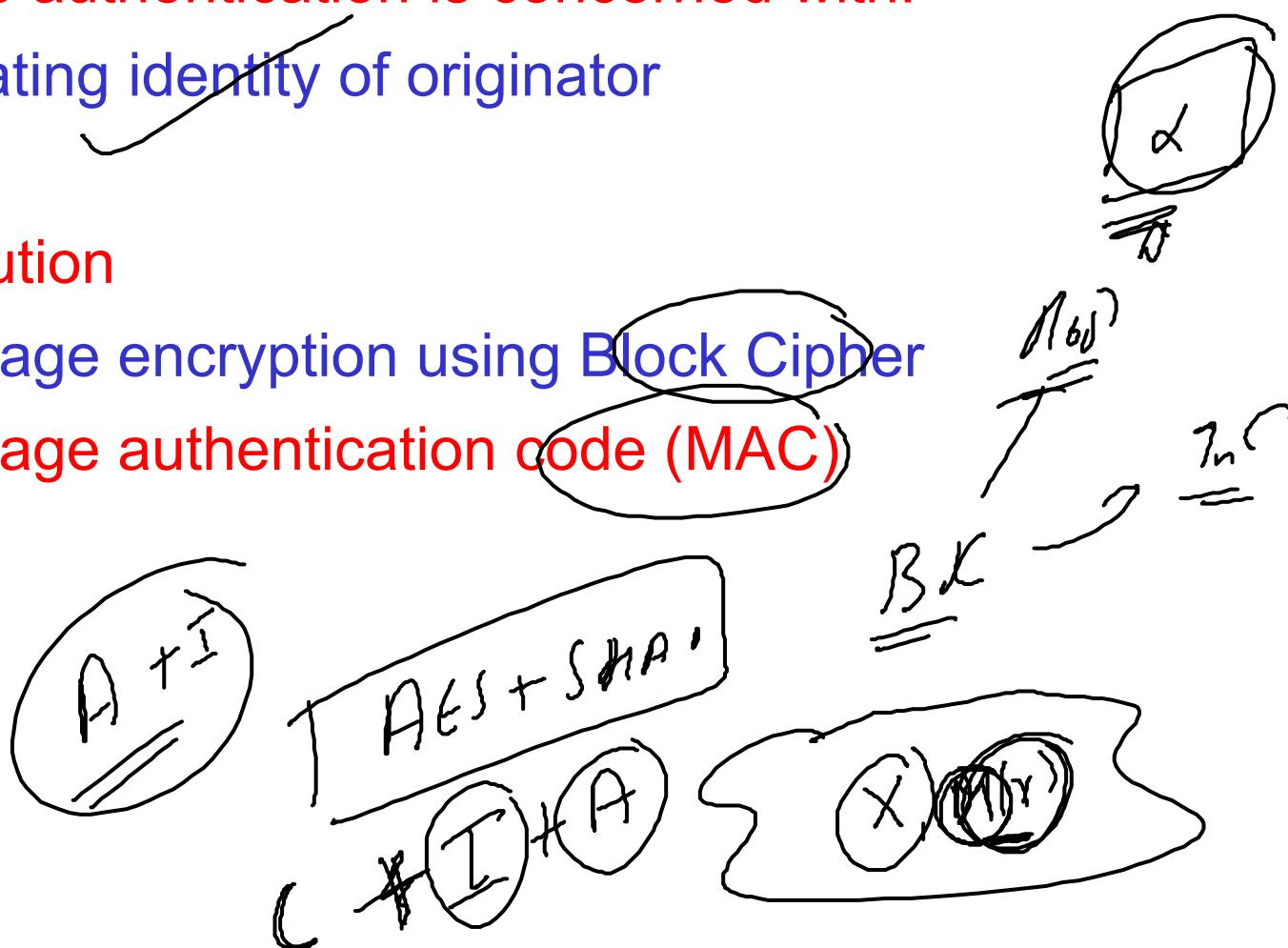
# NS Goals

---

- Information security is an important area of information technology and this course on Network Security help audience to understand the three important security goals in the networks –
  - Confidentiality
  - Integrity
  - Availability
  - Also Authenticity and Non-repudiation and cryptographic techniques to implement these security goals.

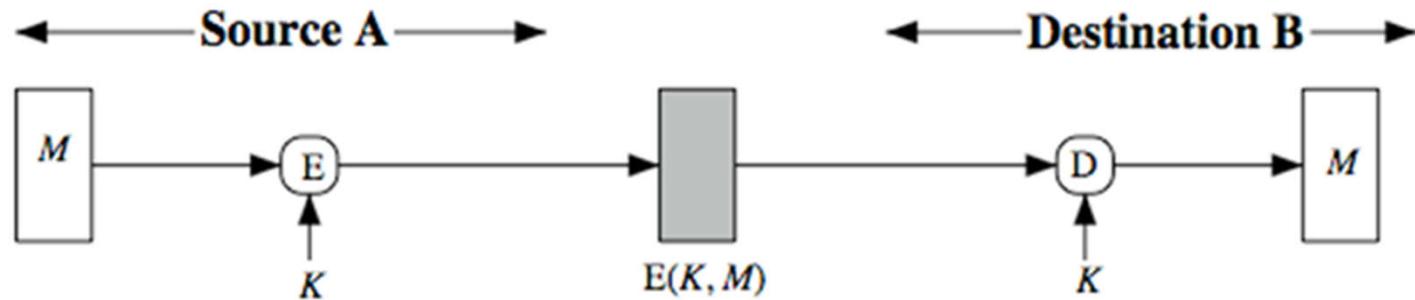
# Message Authentication

- Message authentication is concerned with:
  - validating identity of originator
- Two Solution
  - message encryption using Block Cipher
  - message authentication code (MAC)

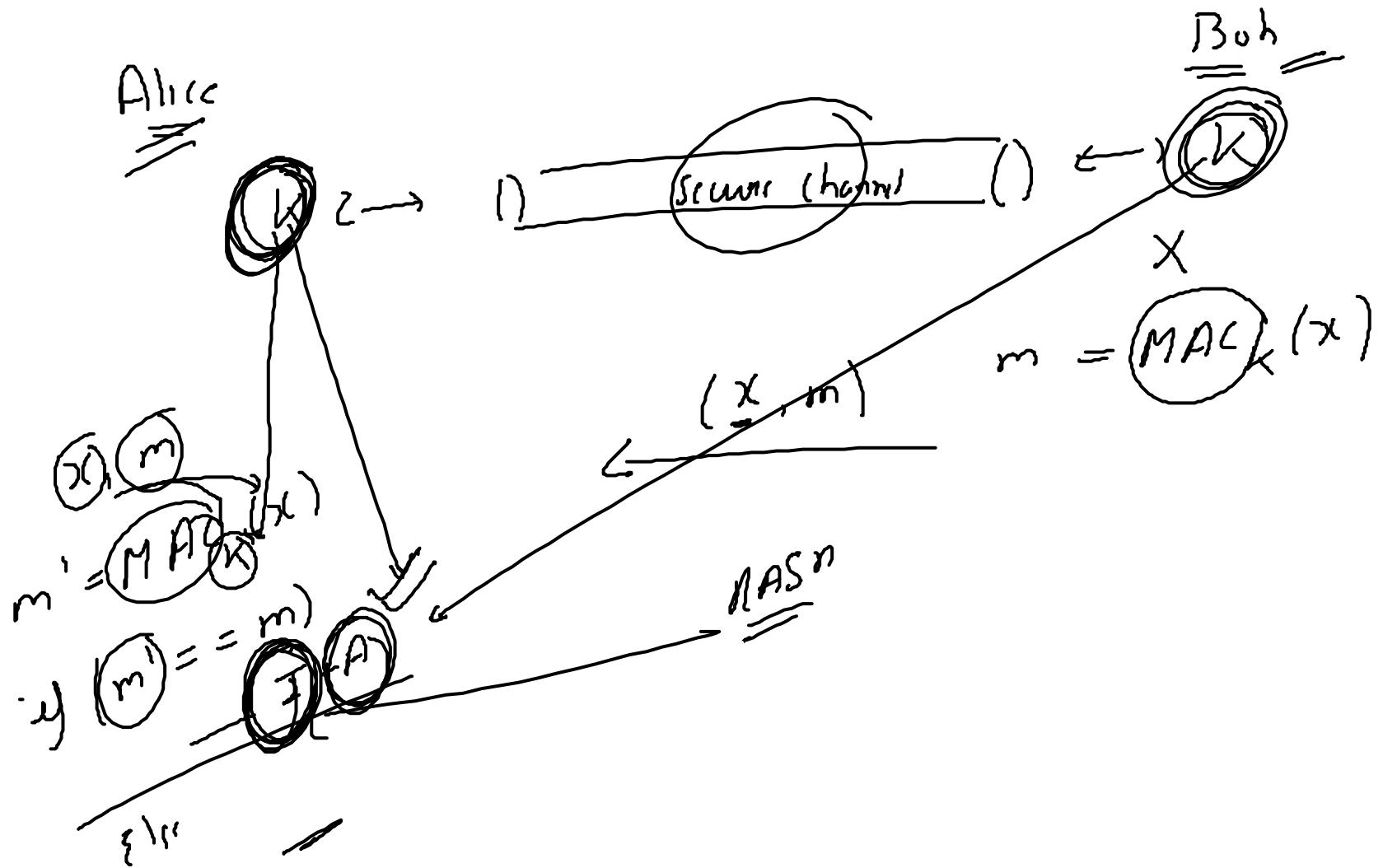


# Symmetric Message Encryption

- Encryption can also provides authentication



# Properties of MAC



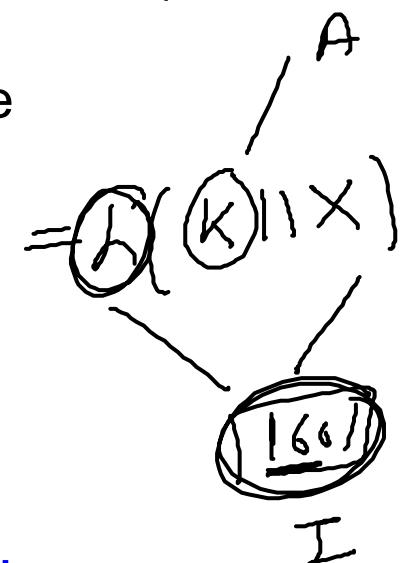
# Message Authentication Code

## Principle:

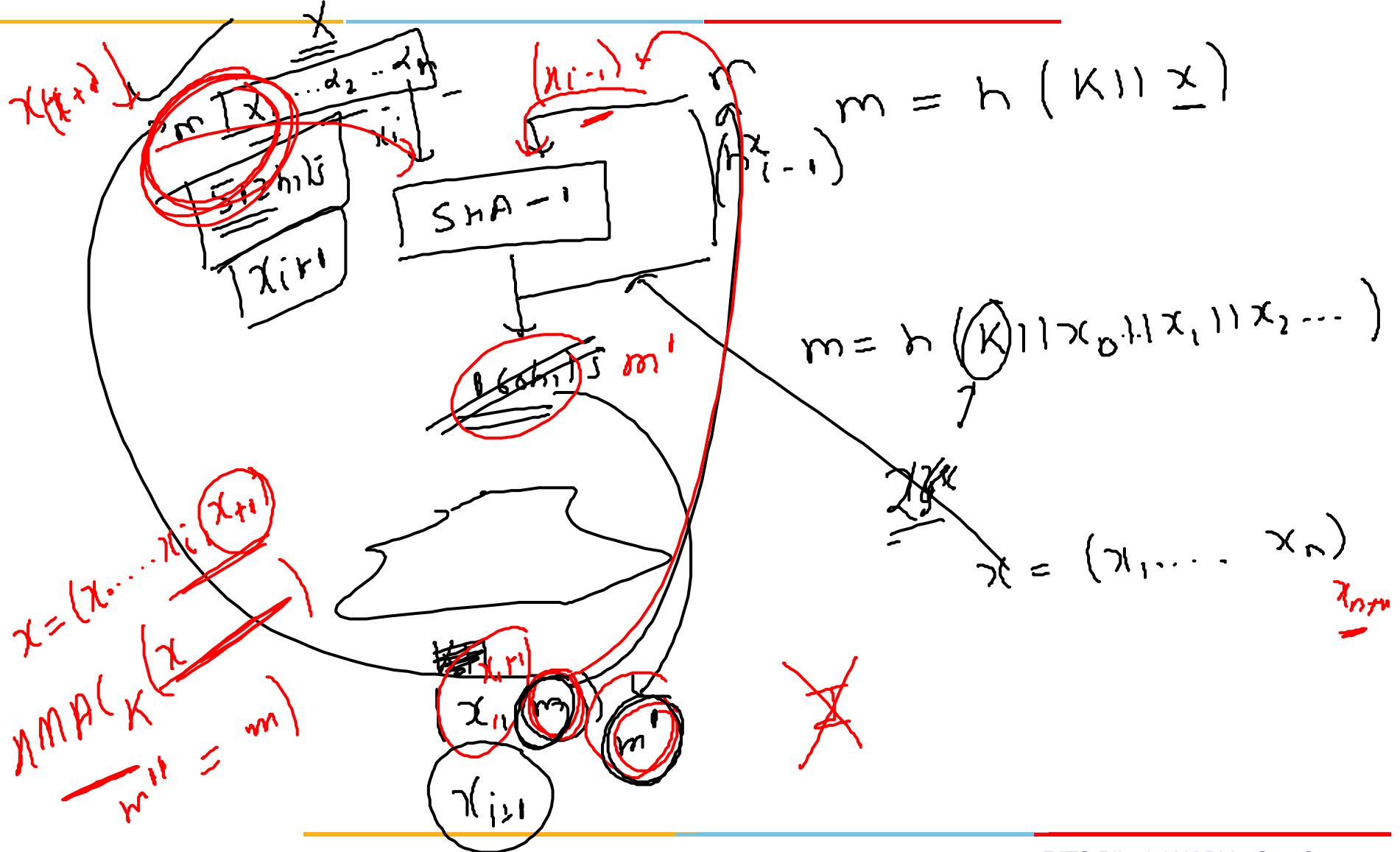
- MAC is realized with ~~cryptographic hash functions~~  
 (e.g., SHA-1)      SHA-1 HMAC      (O/P 160 bits)  
 (e.g., SHA-256)      SHA-256 HMAC      (O/P 256 bits)
- HMAC is such a MAC built from hash functions
- Basic idea: Key is hashed together with the message

- Secret prefix MAC:  $m = \text{MAC}_k(x) = h(k||x)$ .
- Secret suffix MAC:  $m = \text{MAC}_k(x) = h(x||k)$ .
- Have strong one-wayness, hence strong checksum.
- Weakness in secret prefix and suffix MACs.

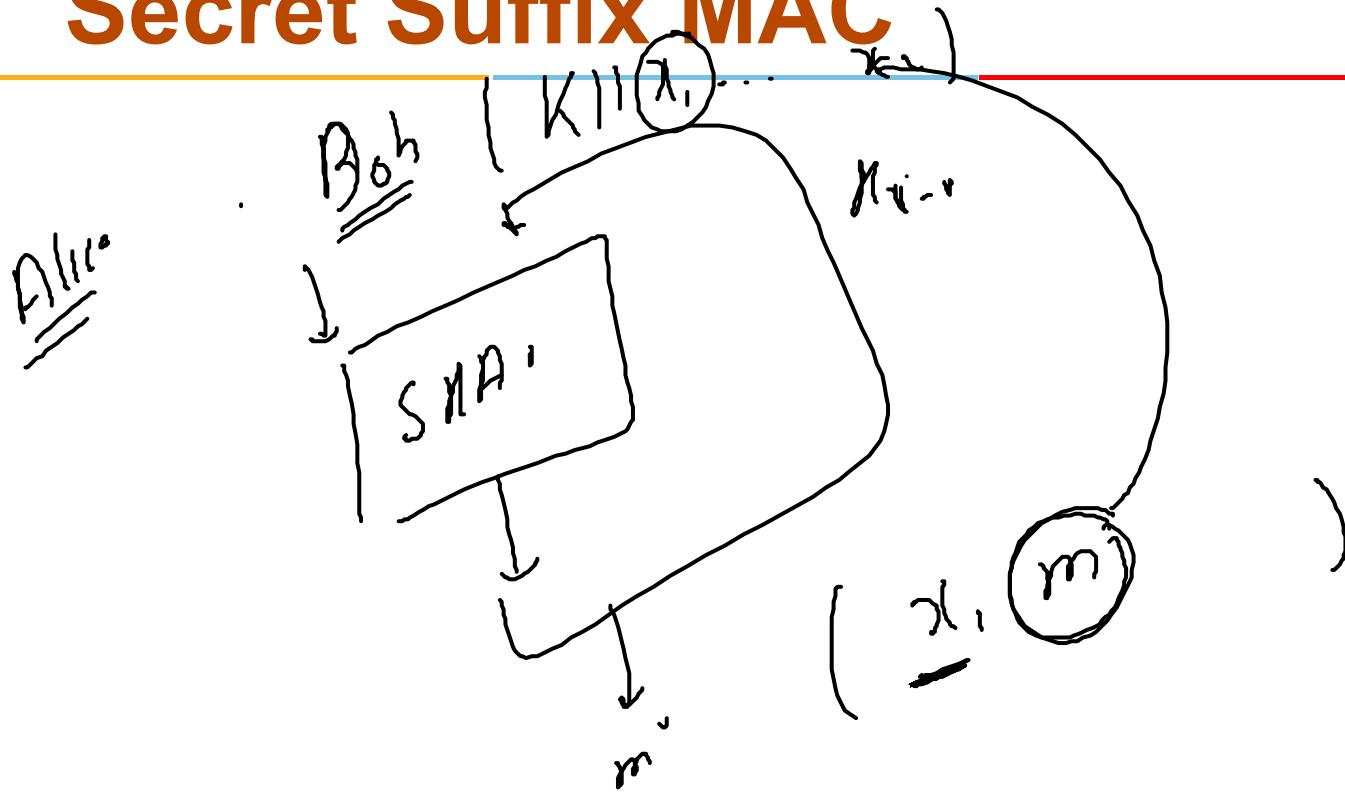
~~MAC~~  
~~No 1~~  
~~(J n o - . )~~



# Secret Prefix MAC



# Secret Suffix MAC

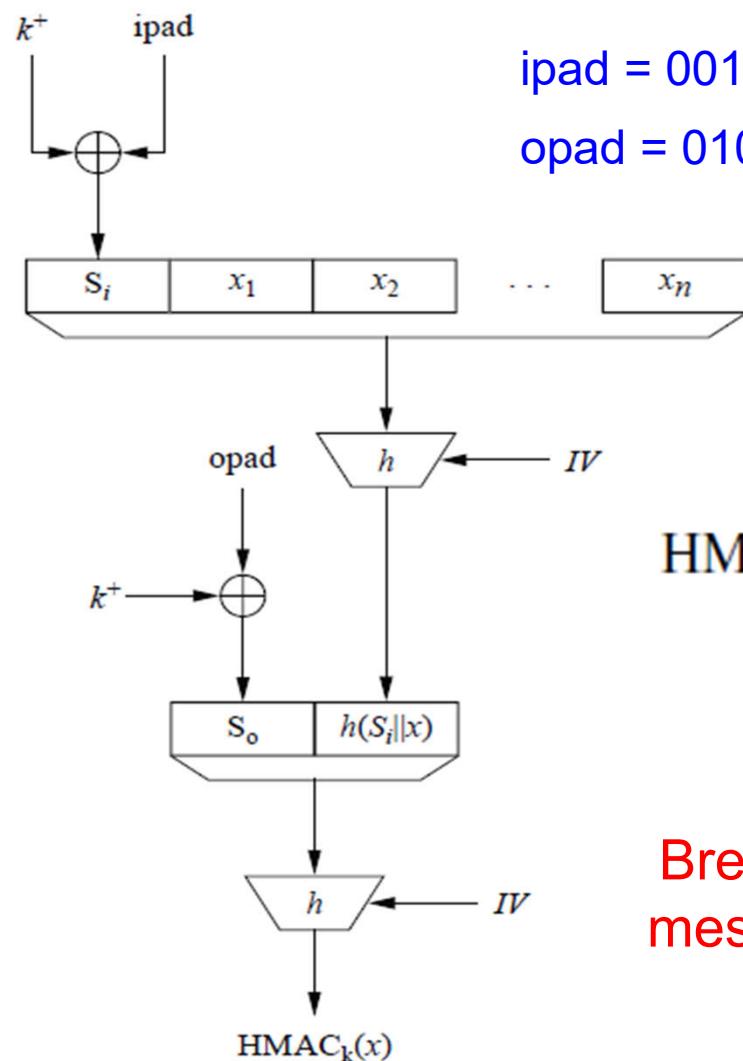




# HMAC

---

# HMAC



ipad = 0011 0110, 0011 0110, ...., 0011 0110, 0011 0110  
 opad = 0101 1100, 0101 1100, ...., 0101 1100, 0101 1100

$$HMAC_k(x) = h [(k^+ \oplus \text{opad}) || h [(k^+ \oplus \text{ipad}) || x]]$$

Breaking the HMAC means constructing valid message tag without knowledge of key.



---

Thanks!!!  
Queries?



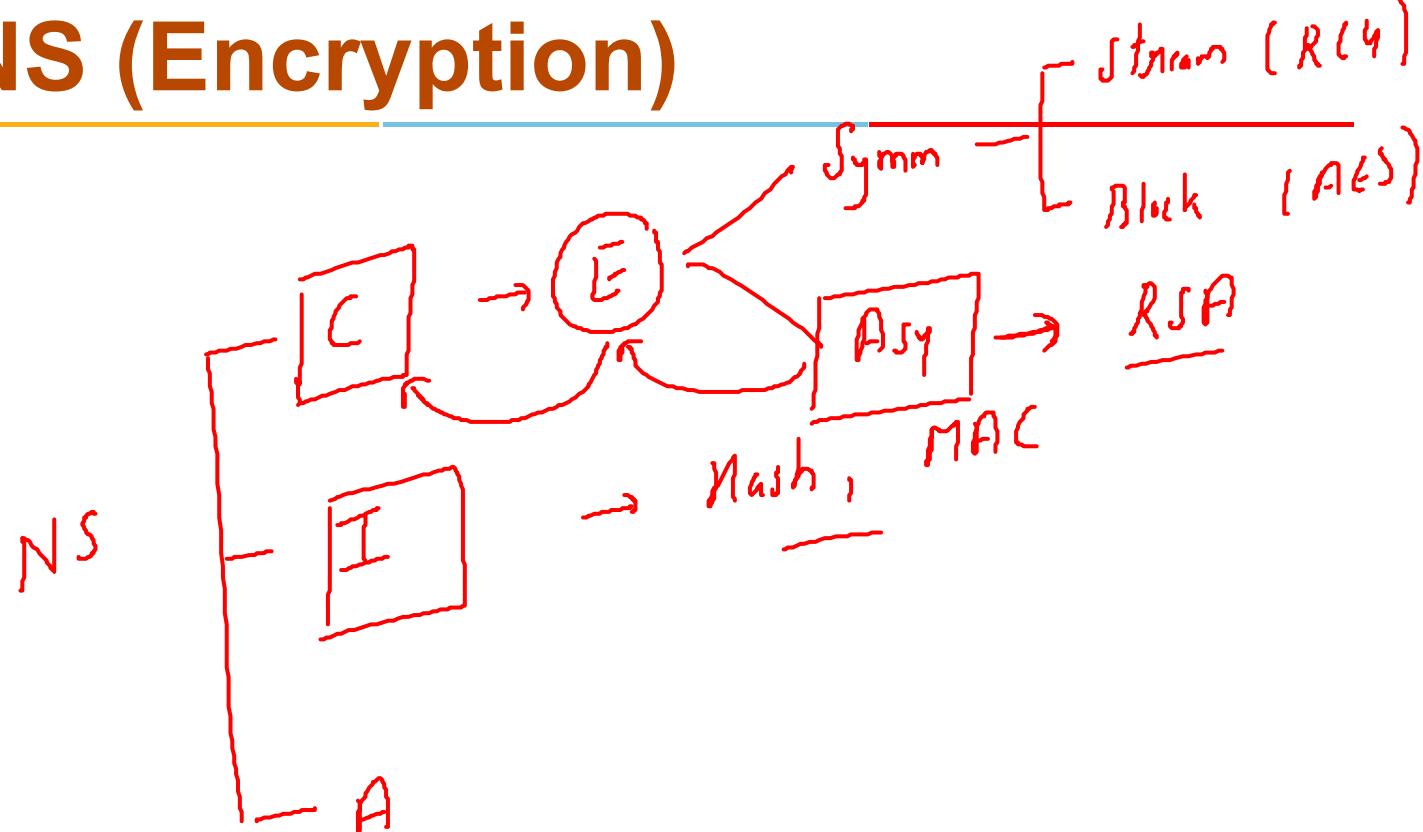
**BITS** Pilani  
K K Birla Goa Campus

# Network Security

## SS ZG513

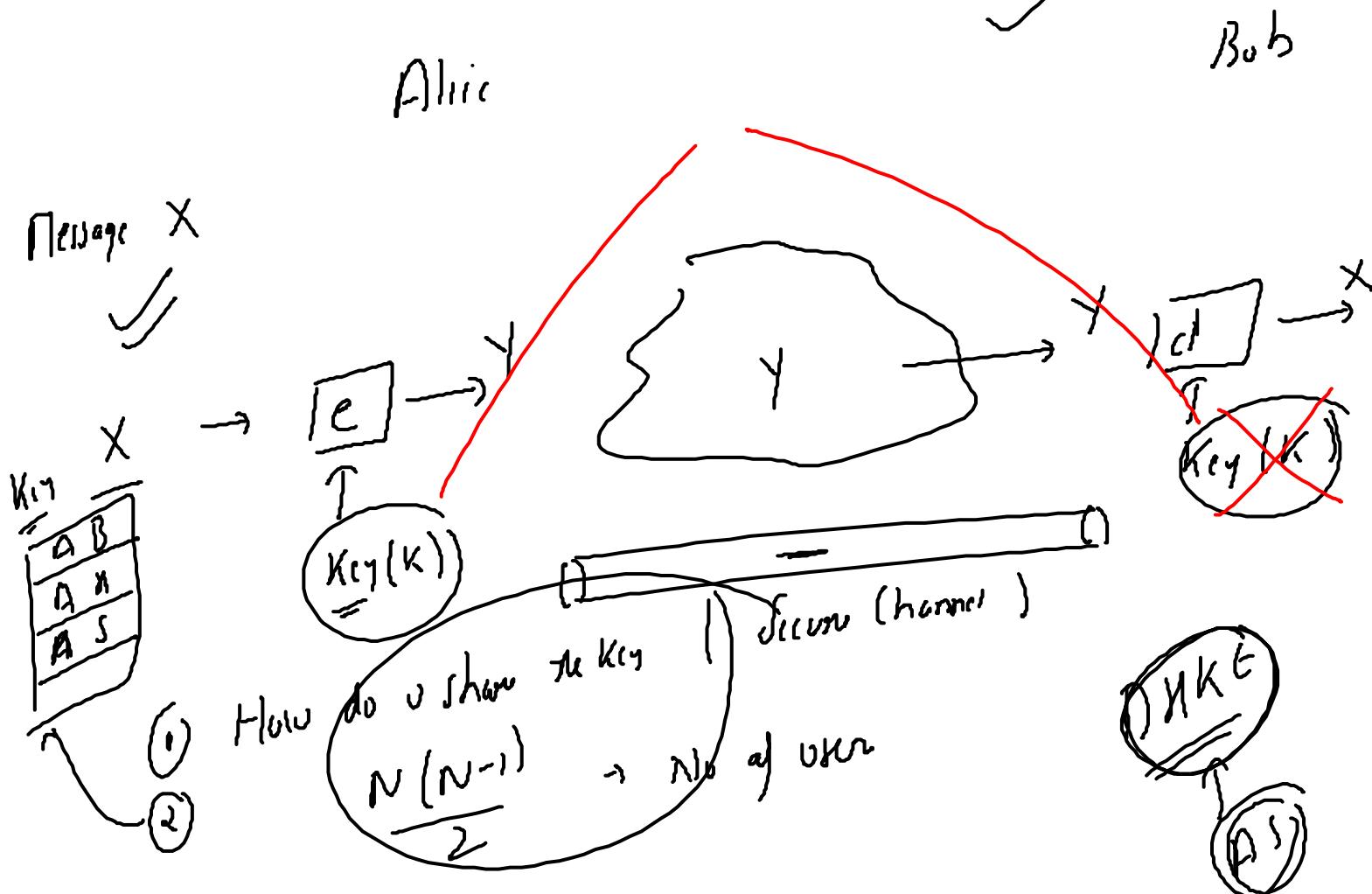
Hemant Rathore  
Department of Computer Science and Information Systems

# NS (Encryption)

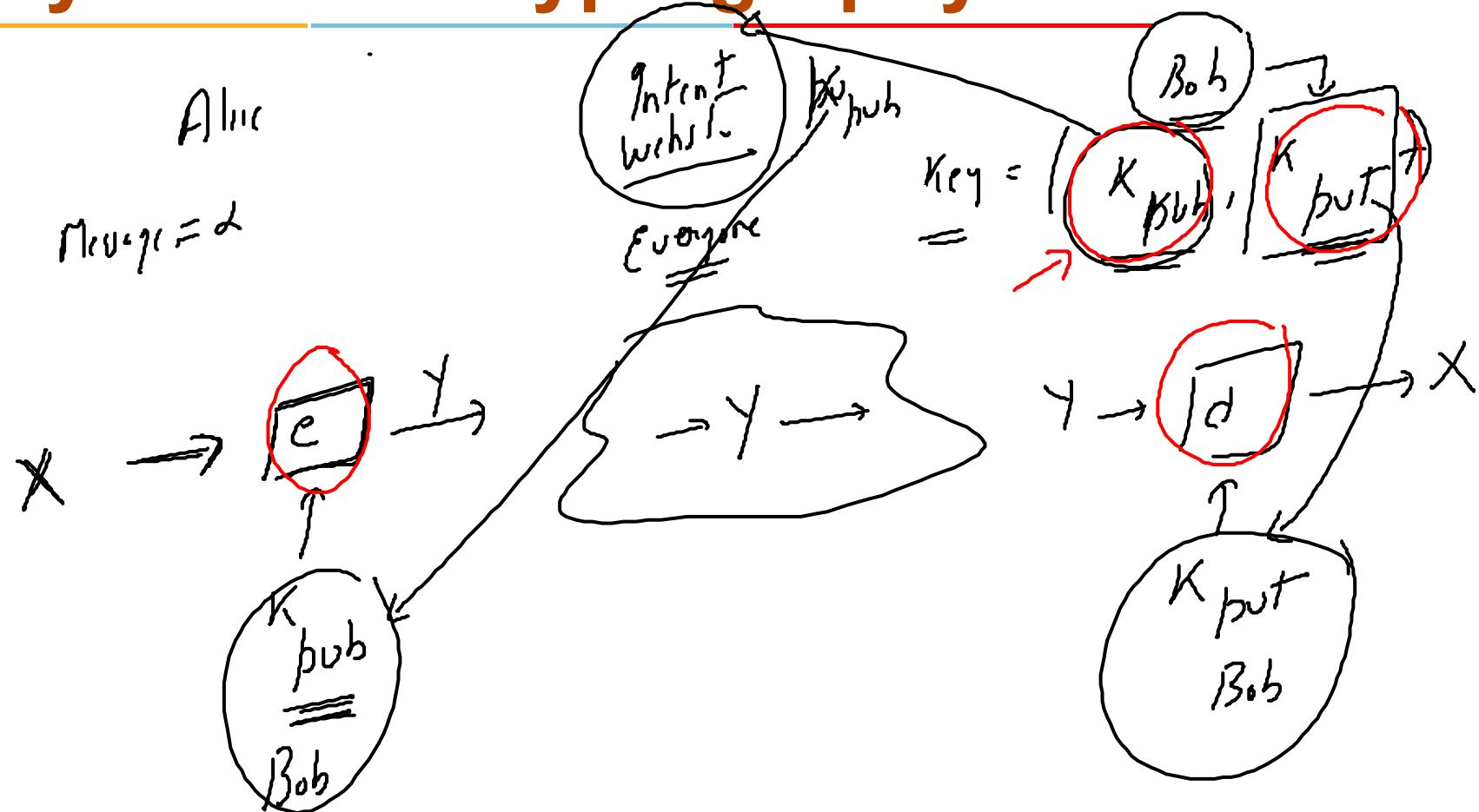


A, NIR

# Symmetric Cryptography



# Asymmetric Cryptography



# Need of Asymmetric Cryptography

- Symmetric Cryptography: Same key, encryption and decryption are similar, efficient and very secure.

$P_{i,j}$   $K_{i,j}$



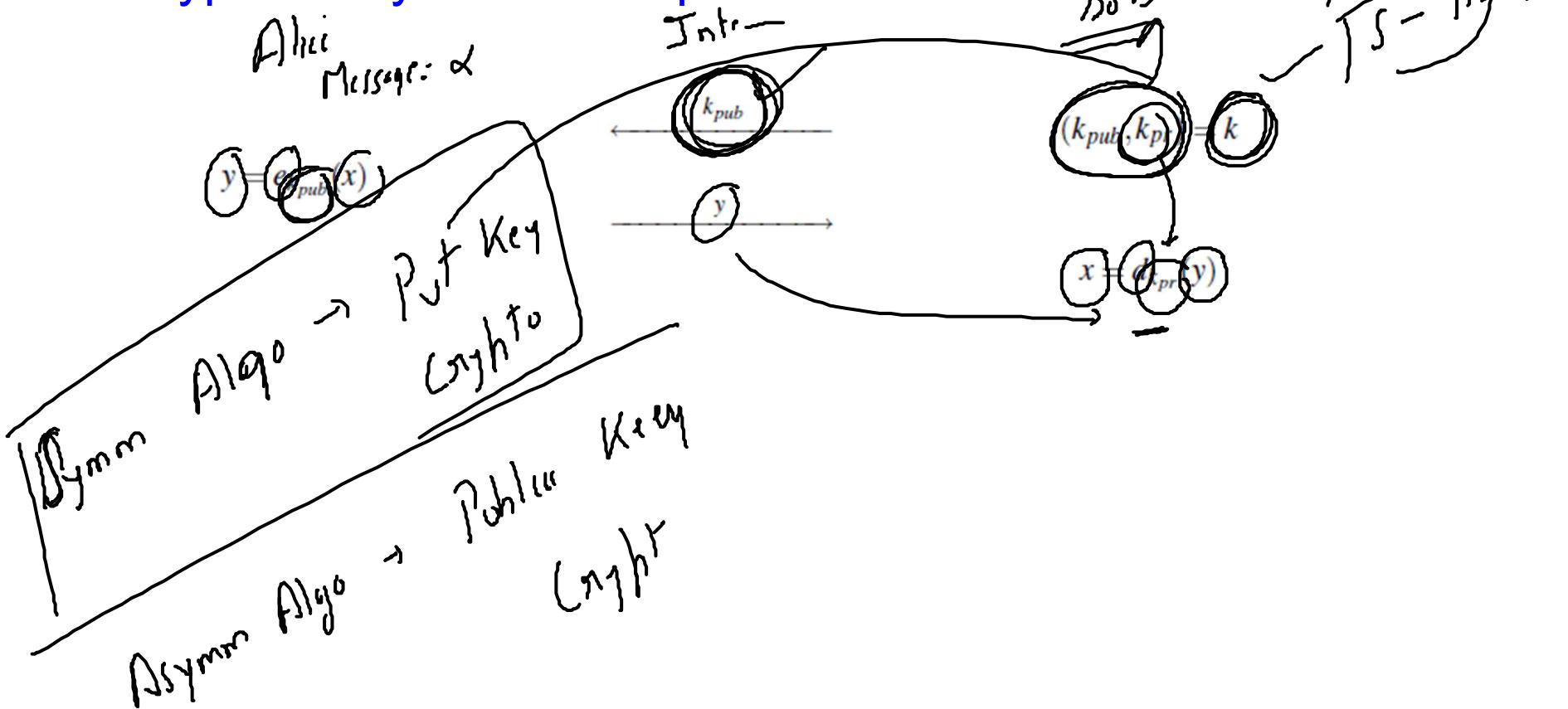
- Short comings of Symmetric Cryptography: Key distribution problem, Number of keys  $n(n-1)/2$ , single point failure, No protection against cheating (non-repudiation).

- Asymmetric cryptography based on number theory was introduced by Whitfield Diffie, Martin E. Hellman and Ralph Mebleam.

(1950)  
RSP

# Principle of PKC

- Encryption key is not required to keep secret, only decryption key has to keep secret.



# Principle of PKC

- Public key algorithms can be built from the concept of one way function, defined as

A function  $f()$  is a one-way function if:

- $y = f(x)$  is computationally easy, and
- $x = f^{-1}(y)$  is computationally infeasible.

- Popular one way function are integer factorization, DLP, ECDL methods.

Eg :  $n = p \times q$

- Given  $p, q$  and performing ( $n = p \times q$ ) is computationally easy
- But, given  $n$ , computing  $p$  and  $q$  is computationally infeasible

# Security Mechanism of PKC

- Encryption (RSA, DLP, ECC, etc.)
- Key establishment (DHKE, RSA, etc.)
- Message integrity and non-repudiation with digital signature.
- Identification using challenge-response and digital signature.
- Problem with PKC is efficiency, long key and authenticity of public key.

PKC

Algorithm Family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer factorization	RSA	1024 bit	2048 bit	7680 bit	15360 bit
Discrete logarithm	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	16360 bit
Elliptic curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric-key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

DS

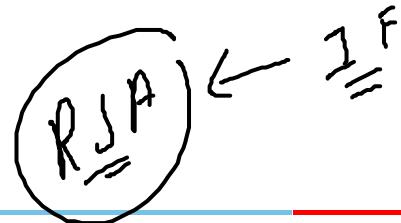
- Complexity grows roughly with cube of the bit size i.e. 1024 to 3072, will 27 times slower.



# RSA

- Not a replacement of Symmetric Cipher.
- Designed by Ronald Rivest, Adi Shamir Leonared Adleman in 1997.
- Based on integer factorization.
- Encryption of small size data, key transport, digital signature, exchange of symmetric key.

# Introduction



- **Encryption:**

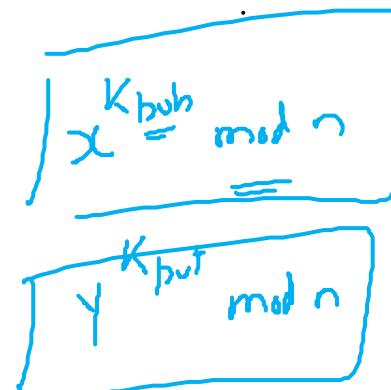
- Given:  $k_{pub} = e$ ,  $n$ , message= $x$  and encryption function

$$y = e_{k_{pub}}(x) \equiv x^e \pmod{n}$$

- **Decryption:**

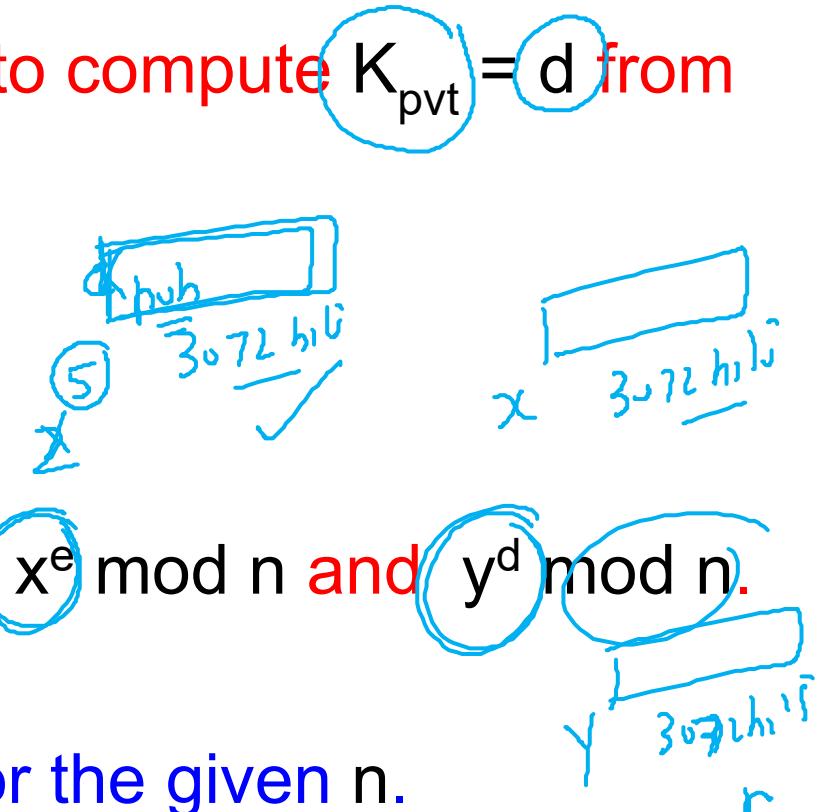
$$x = d_{k_{pri}}(y) \equiv y^d \pmod{n}$$

- $n, e, d, x, y \in \mathbb{Z}_n$



# RSA Requirements

- Computationally in-feasible to compute  $K_{\text{pvt}} = d$  from  $K_{\text{pub}} = n$  and  $e$
- $x \leq n$ .
- Efficient method to compute  $x^e \bmod n$  and  $y^d \bmod n$ .
- Multiple pairs should exist for the given  $n$ .



# Key Generation

How to generate  $n$ ,  $e$  and  $d$ ?

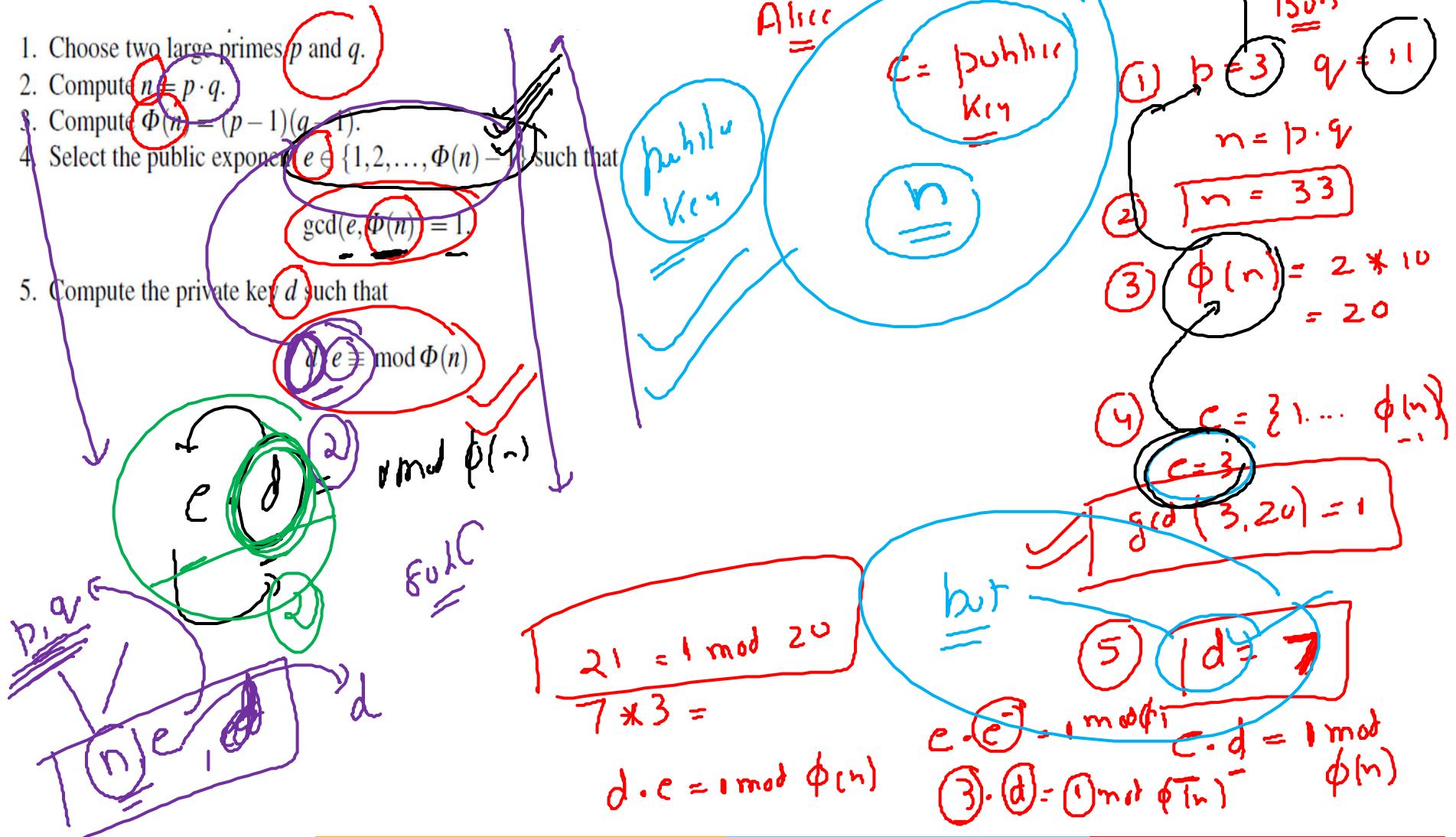
1. Choose two large primes  $p$  and  $q$ .
2. Compute  $n = p \cdot q$ .
3. Compute  $\Phi(n) = (p - 1)(q - 1)$ .
4. Select the public exponent  $e \in \{1, 2, \dots, \Phi(n) - 1\}$  such that  

$$\gcd(e, \Phi(n)) = 1$$
5. Compute the private key  $d$  such that  

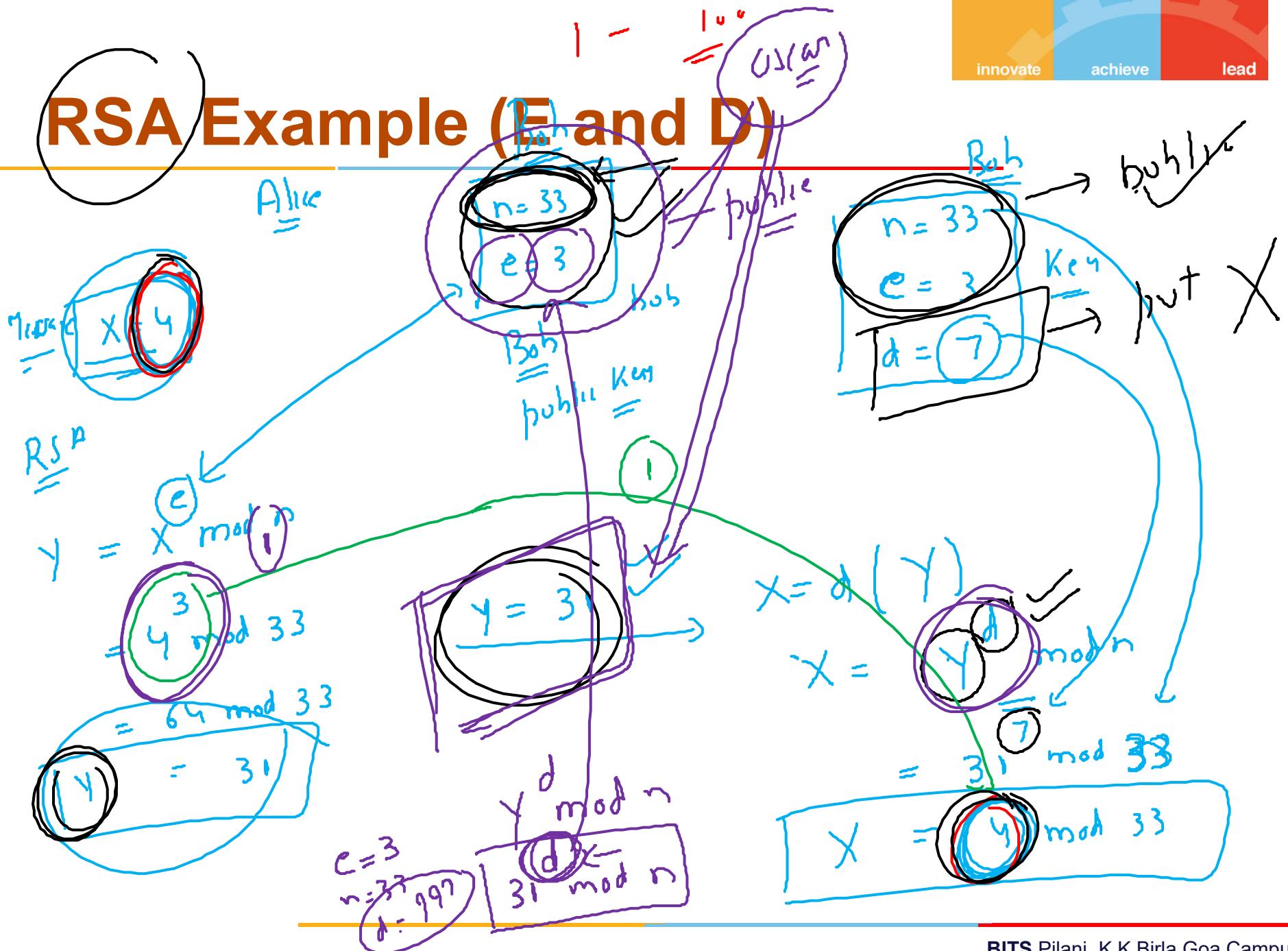
$$d \cdot e \equiv 1 \pmod{\Phi(n)}$$

# RSA Example (Key Generation)

1. Choose two large primes  $p$  and  $q$ .
2. Compute  $n = p \cdot q$ .
3. Compute  $\Phi(n) = (p-1)(q-1)$ .
4. Select the public exponent  $e \in \{1, 2, \dots, \Phi(n)-1\}$  such that  $\gcd(e, \Phi(n)) = 1$ .
5. Compute the private key  $d$  such that  $d \cdot e \equiv 1 \pmod{\Phi(n)}$ .



# RSA Example (E and D)



# Fast Exponentiation (1)

compute

$$y = x^e \bmod n$$

$$x = y^d \bmod n$$

$$x^4$$

Not  $x^2$

$$x \cdot x = x^2$$

$$x^2 \cdot x = x^3$$

$$x^3 \cdot x = \boxed{x^4}$$

better

$$x \cdot x = x^2$$

$$x^2 \cdot x^2 = x^4$$

$$2 \text{ MUL}$$

$$(E \circ y) \quad - (1)$$

$$(d \circ y) \quad - (2)$$

$$x^8$$

$$x \cdot x = x^2$$

$$x^2 \cdot x = x^3$$

$$x^3 \cdot x = x^4$$

$$\vdots$$

$$x^7 \cdot x = x^8$$

$$x^8 = \boxed{mu.}$$

$$x \cdot x = x^2$$

$$x^2 \cdot x^2 = x^4$$

$$x^4 \cdot x^4 = x^8$$

$$\vdots$$

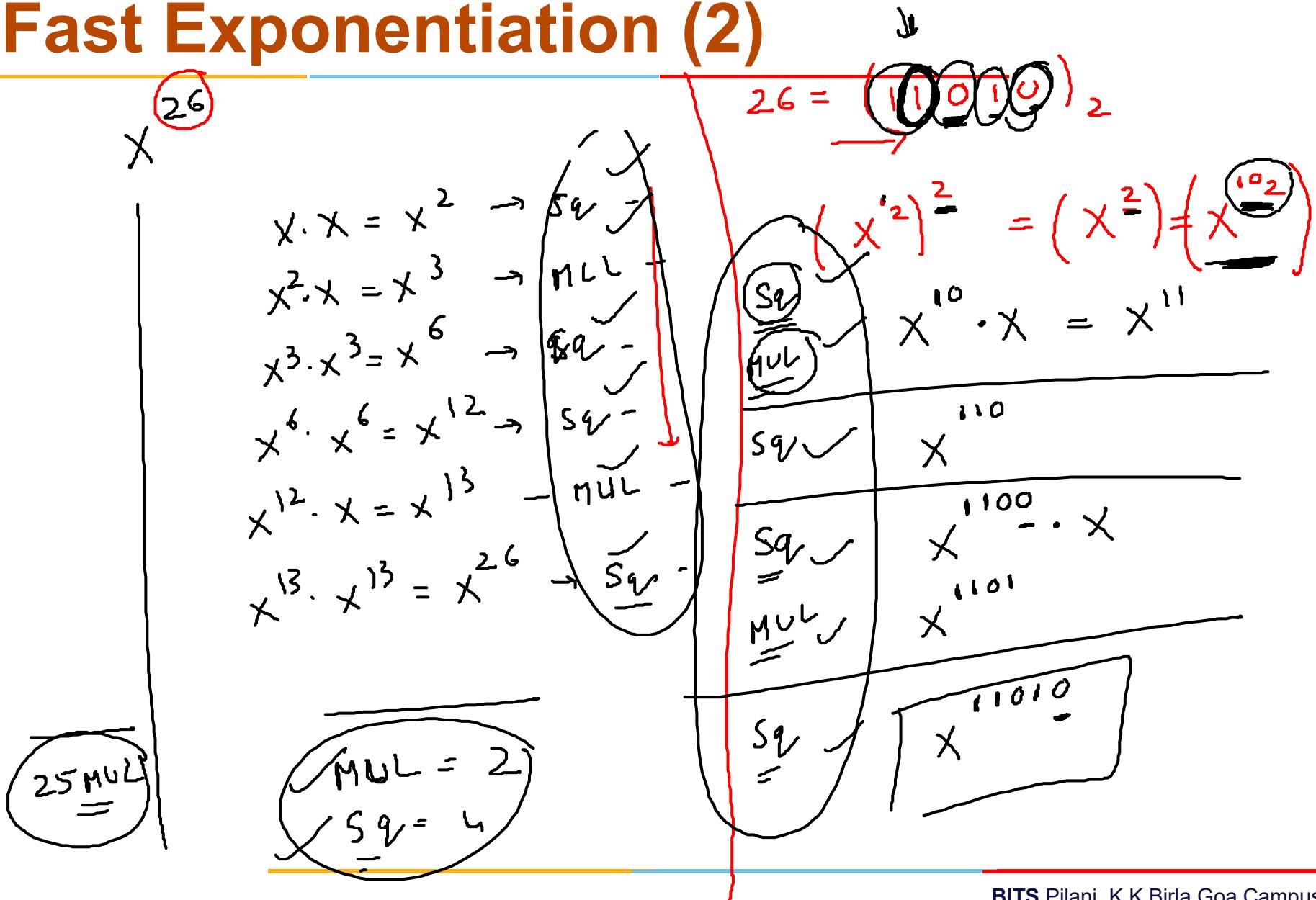
$$x^7 \cdot x = x^8$$

$$x^8 = \boxed{mu.}$$

$$3 \text{ MUL}$$

$$3 \text{ MUL}$$

# Fast Exponentiation (2)





# Fast Exponentiation (3)

---

$s_i = r^i$



# Square and Multiply Algo

Algorithm: Square-and-Multiply for  $x^H \bmod n$

**Input:** Exponent  $H$ , base element  $x$ , Modulus  $n$

**Output:**  $y = x^H \bmod n$

$$x^H \bmod n =$$

1. Determine binary representation  $H = (h_t, h_{t-1}, \dots, h_0)_2$

2. **FOR**  $i = t-1$  TO 0

3.  $y = y^2 \bmod n \rightarrow \text{Square}$

4. **IF**  $h_i = 1$  **THEN**

5.  $y = y * x \bmod n \rightarrow \text{MUL}$

6. **RETURN**  $y$



# gcd (Intro)



# gcd (27,21) using EA

---



# gcd (973, 301) using EA

---



# gcd (973, 301) using EEA

---



---

Thanks!!!  
Queries?



**BITS** Pilani  
K K Birla Goa Campus

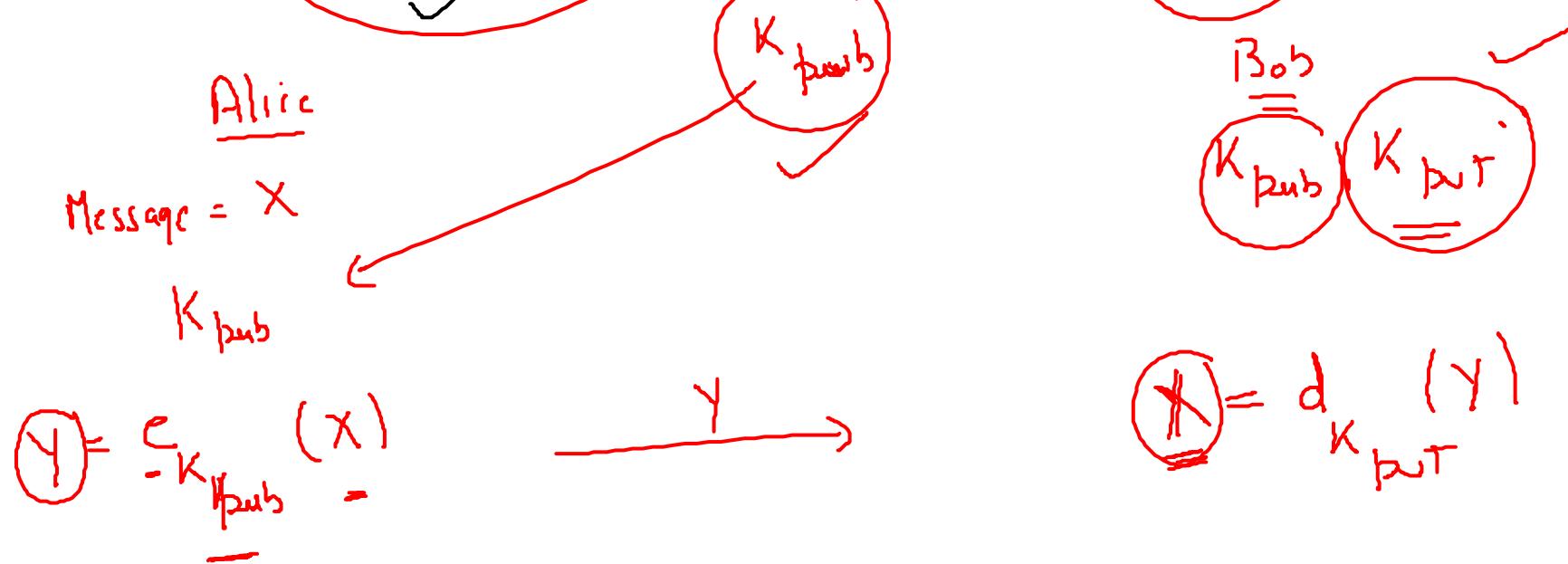
# Network Security

## SS ZG513

Hemant Rathore  
Department of Computer Science and Information Systems



# Asymmetric Cryptography / RSA



# RSA Key Generation

1. Choose two large primes  $p$  and  $q$ .

2. Compute  $n = p \cdot q$ .

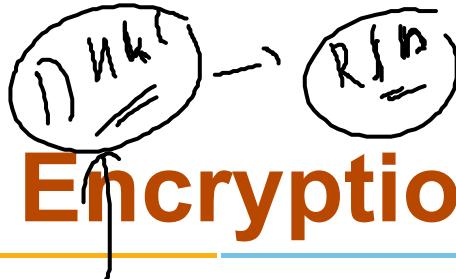
3. Compute  $\Phi(n) = (p-1)(q-1)$ .

4. Select the public exponent  $e \in \{1, 2, \dots, \Phi(n)-1\}$  such that

$$\begin{aligned} & e \\ & \text{gcd}(e, \Phi(n)) = 1. \end{aligned}$$

5. Compute the private key  $d$  such that

$$\begin{aligned} & d \cdot e \equiv 1 \pmod{\Phi(n)} \\ & d \cdot e \equiv 1 \pmod{\phi(n)} \end{aligned}$$



# RSA Encryption / Decryption

- Encryption:

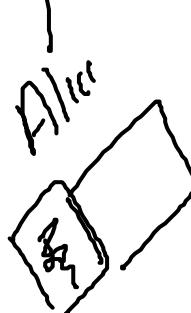
$$y = e_{k_{pub}}(x) \equiv x^e \pmod{n}$$

- Decryption:

$$x = d_{k_{pr}}(y) \equiv y^d \pmod{n}$$

- $n, e, d, x, y \in \mathbb{Z}_n$

Bob



15+

AES  
128

Ech

AES-128

128



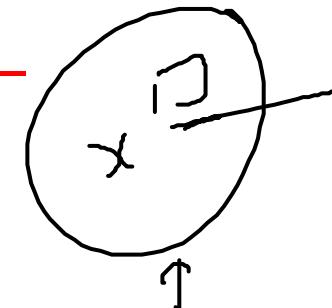
# Square and Multiply Algo

## Algorithm: Square-and-Multiply for $x^H \bmod n$

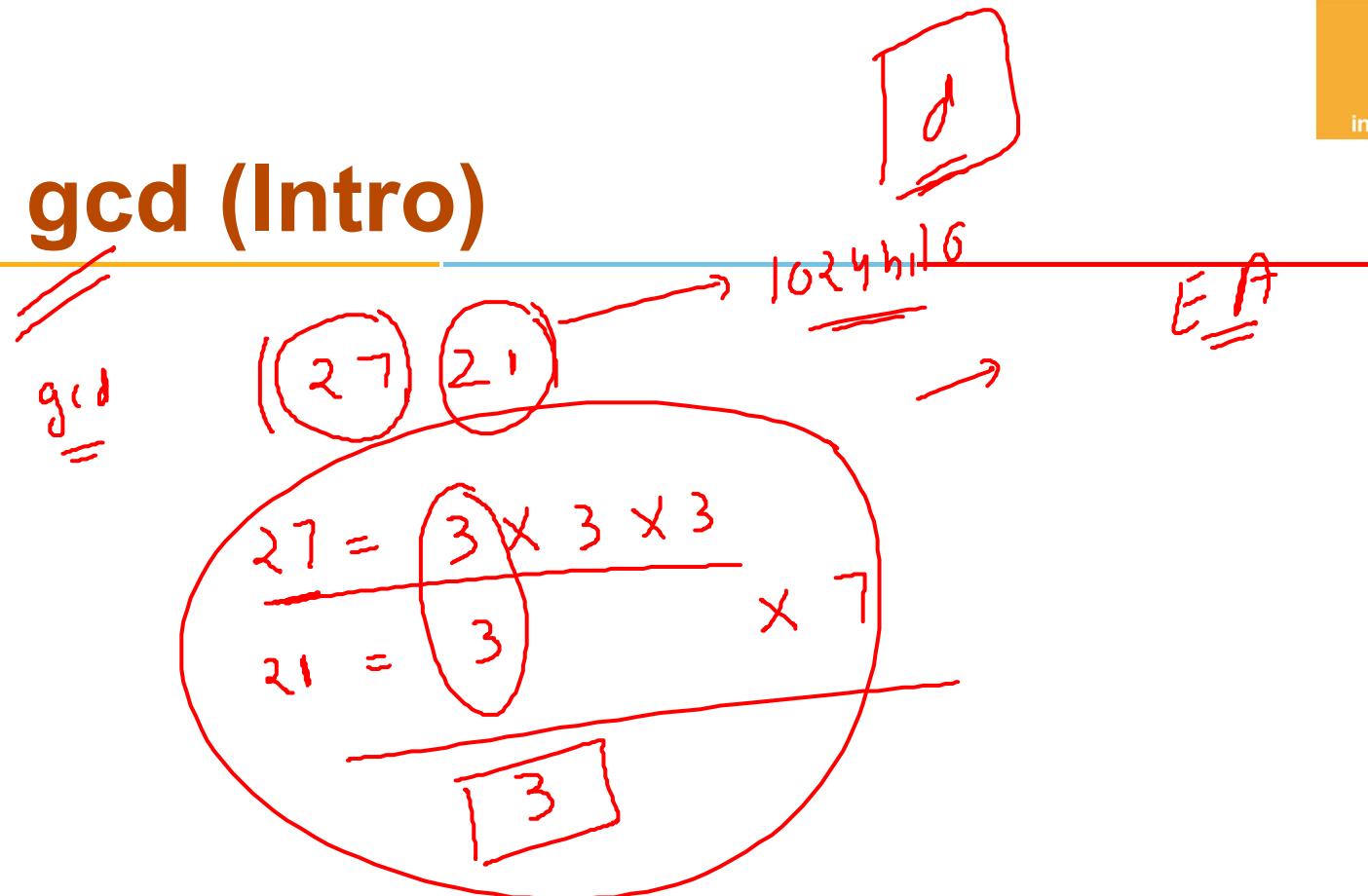
**Input:** Exponent  $H$ , base element  $x$ , Modulus  $n$

**Output:**  $y = x^H \bmod n$

1. Determine binary representation  $H = (h_t, h_{t-1}, \dots, h_0)_2$
2. **FOR**  $i = t-1$  **TO** 0
3.        $y = y^2 \bmod n$
4.       **IF**  $h_i = 1$  **THEN**
5.            $y = y * x \bmod n$
6. **RETURN**  $y$



# gcd (Intro)



# gcd (27, 21) using EA

$$\begin{aligned}
 & \text{gcd} (27, 21) \\
 &= \text{gcd} (\underline{\text{gcd} (\underline{m_0}, \underline{m_1})}, \underline{\underline{m_1}}) \\
 &= \text{gcd} (\underline{\text{gcd} (\underline{m_0 \text{ mod } m_1}, \underline{\underline{m_1}})}, \underline{\underline{\underline{m_1}}}) \\
 &= \text{gcd} (\underline{\text{gcd} (\underline{m_2}, \underline{m_1})}, \underline{\underline{\underline{m_1}}}) \\
 &= \text{gcd} (\underline{\text{gcd} (\underline{m_2 \text{ mod } m_1}, \underline{\underline{\underline{m_1}}})}, \underline{\underline{\underline{\underline{m_1}}}})
 \end{aligned}$$

$$\begin{aligned}
 3 &= \text{gcd} (\underline{\underline{\underline{\underline{m_0}}}}, \underline{\underline{\underline{\underline{m_1}}}}) \\
 3 &= \text{gcd} (\underline{\underline{\underline{\underline{27 \text{ mod } 21}}}}, \underline{\underline{\underline{21}}}) \\
 3 &= \text{gcd} (\underline{\underline{\underline{6}}}, \underline{\underline{\underline{21}}}) \\
 &= \text{gcd} (\underline{\underline{\underline{6}}}, \underline{\underline{\underline{21 \text{ mod } 6}}}) \\
 &= \text{gcd} (\underline{\underline{\underline{6}}}, \underline{\underline{\underline{3}}}) \\
 &= \text{gcd} (\underline{\underline{\underline{6 \text{ mod } 3}}}, \underline{\underline{\underline{3}}})
 \end{aligned}$$

$$7 = 13 * 973 + 42 * 301$$

innovate

achieve

lead

## gcd (973, 301) using EA

$$7 = \underline{973}$$

$$301 = 1 \\ 973 = 3 * \underline{301} + 70$$

$$7 = \underline{70} - 3 * 21$$

$$2 \\ 301 = \cancel{3} * 70 + 21$$

$$7 = \underline{3 * 21} + 7 - \underline{3 * 21}$$

$$7 = 3 * (301 - 4 * 70) + 7$$

$$- 3(301 - 4 * 70)$$

$$= 3 * \underline{301} - 12 * 70$$

$$3 \\ 70 = 3 * 21 + 7$$

$$+ 7 - 3 * 301 + 12 * 70$$

$$7 = 973 \\ 301$$

$$+ 7 - 3 * 301 + 12 * (973 - 3 * 301)$$

$$7 = 70 - 3 * 21$$

# gcd (973, 301) using EEA

$$\text{gcd} \left( \frac{973}{m_0}, \frac{301}{m_1} \right) = S \cdot \frac{973}{m_0} + T \cdot \frac{301}{m_1} = 7$$

(1)

$$7 = 5 \cdot 973 + (-1) \cdot 301$$

(2)



# EEA for finding inverse

---



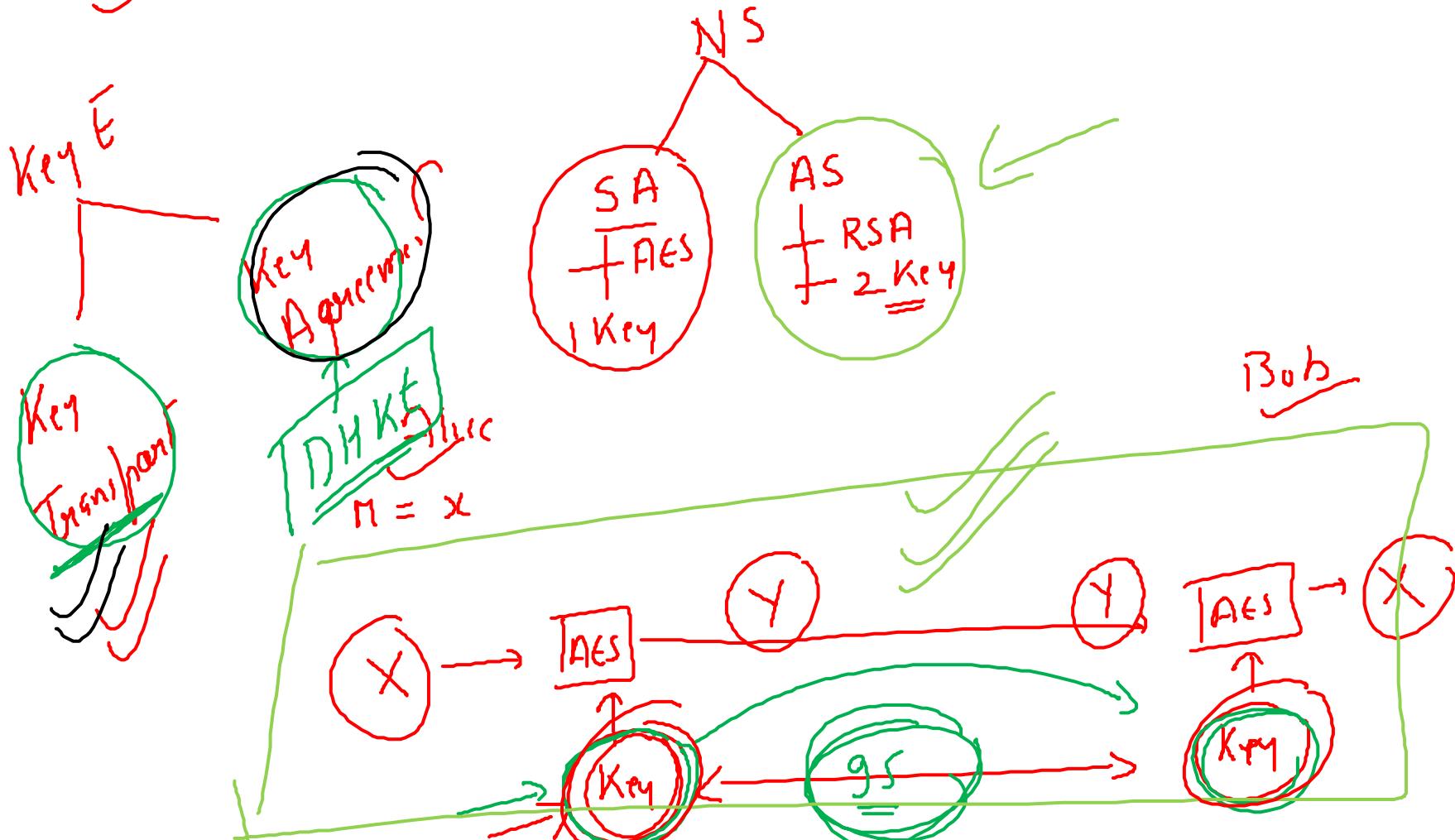
---

# Thank You

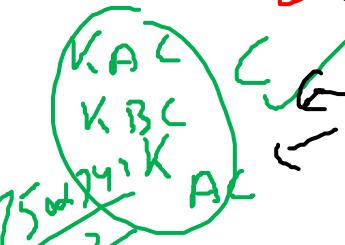
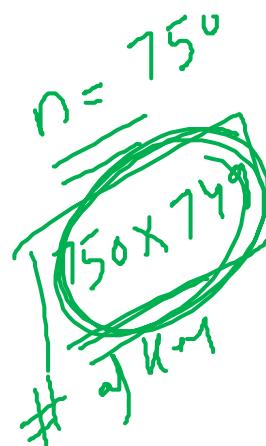


# Key Establishment (Motivation)

# Key Establishment (Intro)

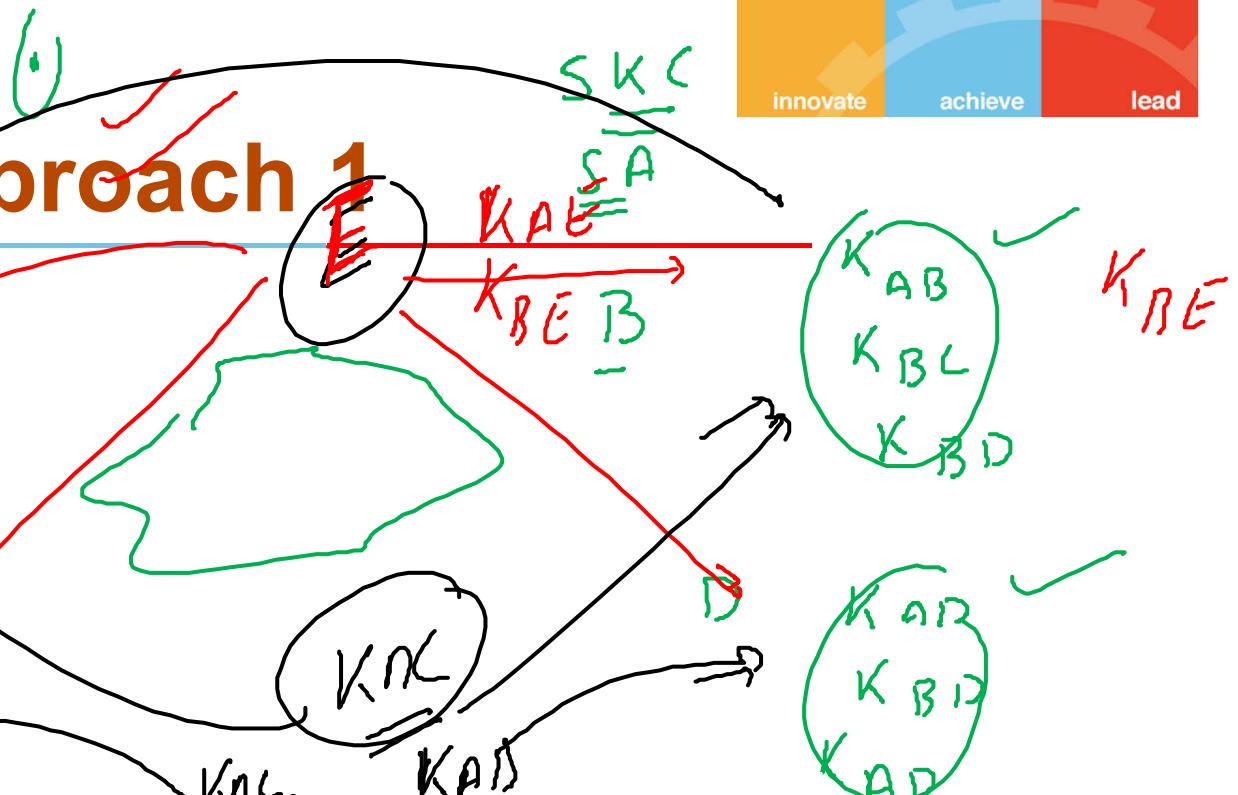


# KE Naïve Approach 1



$$\begin{aligned} \text{No of Keys} &= n \times (n-1) \\ \text{where } n &= \text{no of users} \\ &= 4 \times 3 \\ &= 12 \end{aligned}$$

$$\begin{aligned} \# \text{ of } K_{\text{set}} \text{ pairs} &= \frac{n \times (n-1)}{2} \\ &= \frac{12}{2} \\ &= 6 \end{aligned}$$





# KE Naïve Approach 2

Drawback

1)

No of

K+1

queries

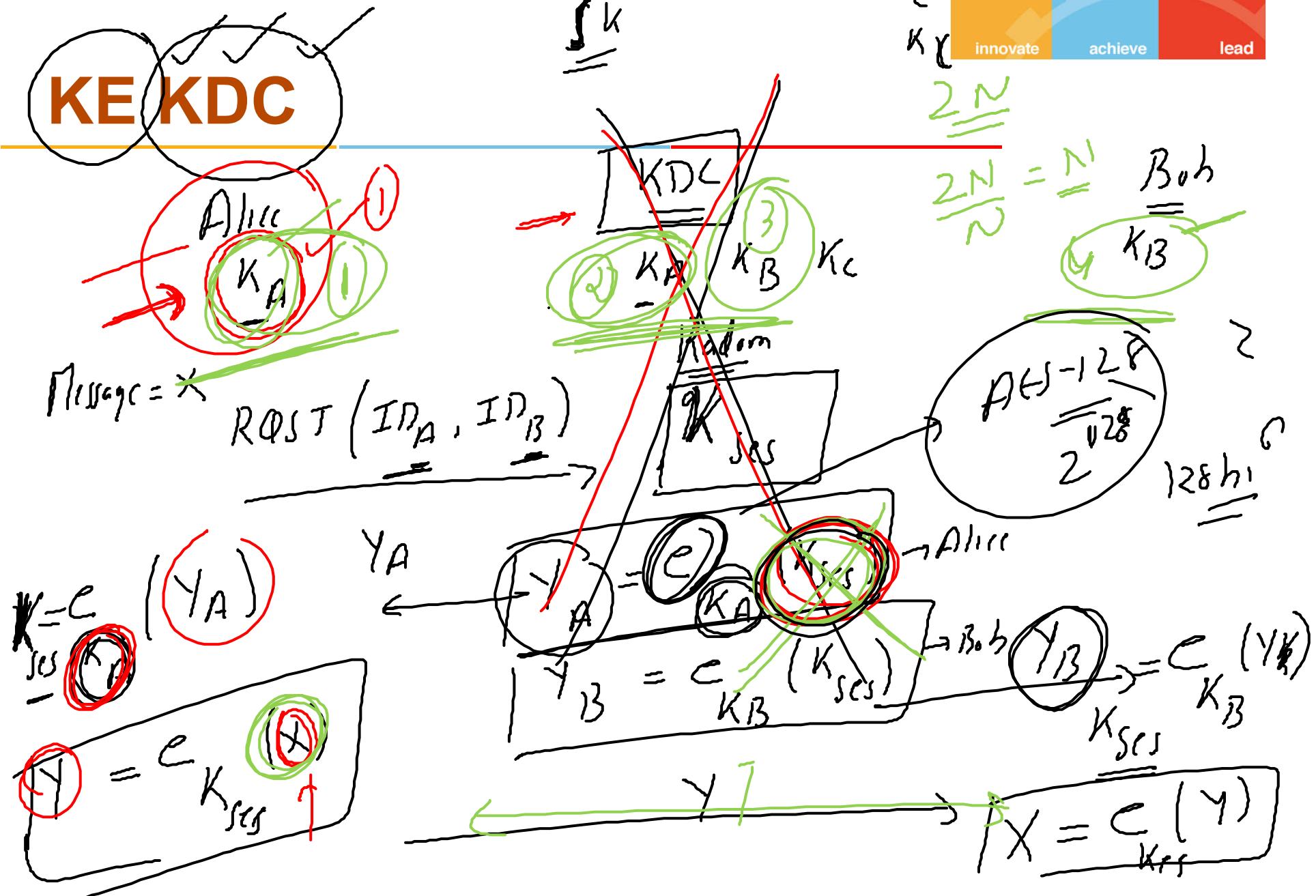
2)

quadratic



# KE Naïve Approach Drawbacks

---



# KE KDC (No of keys, +ve, -ve) 1

$$\# \text{ of keys} = \underline{\underline{2n}} \rightarrow \text{Linear} = \\ n = \# \text{ of users}$$

$$\# \text{ of key pairs} = \underline{\underline{N}}$$

+ve  
-ve

Single point of failure



# KE KDC (No of keys, +ve, -ve) 2

---



# KE Asymmetric Key Establishment

---

# Diffe-Hellman Key Exchange

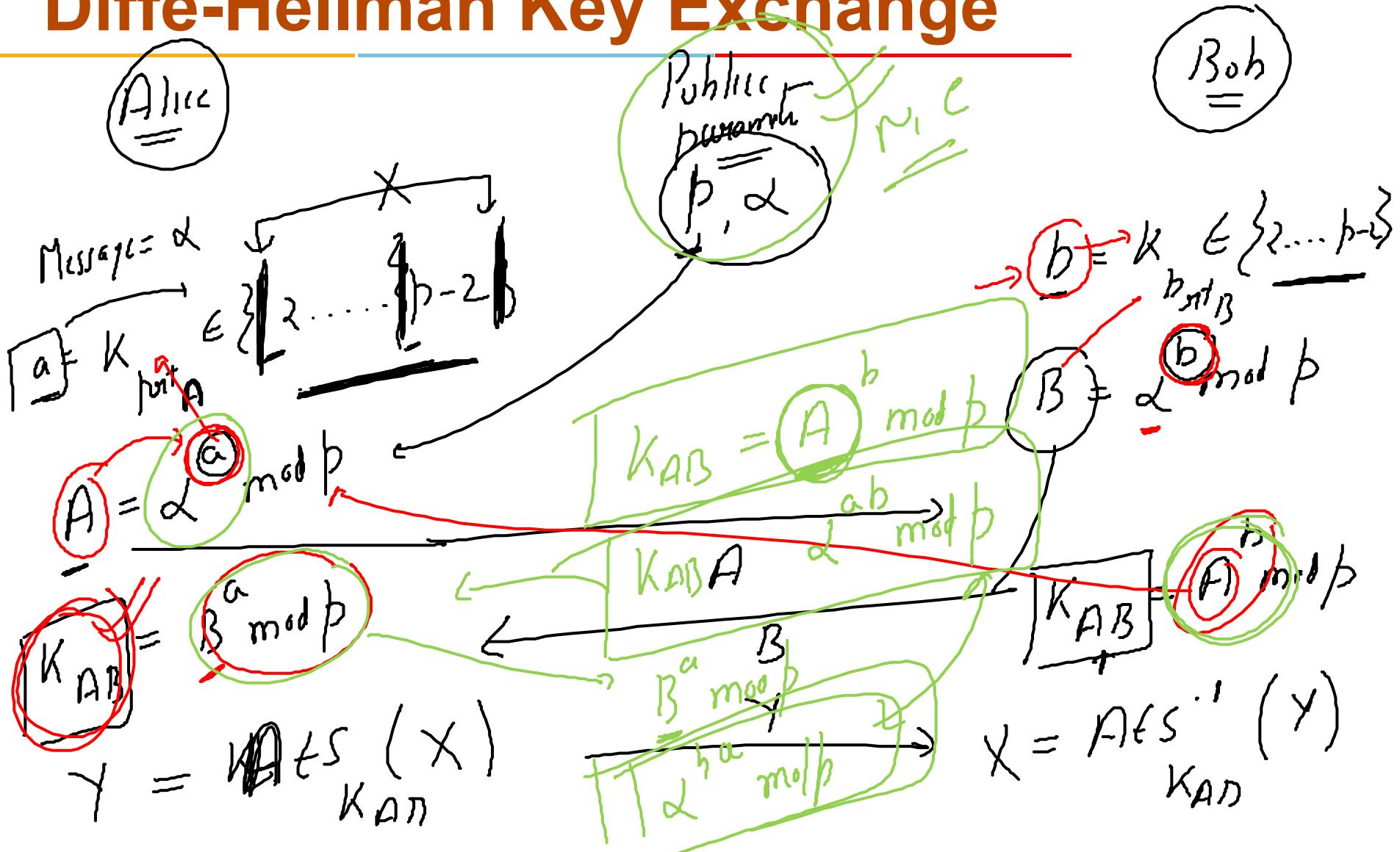
- Based on the property that exponentiation is commutative.

$$k = (\alpha^x)^y = (\alpha^y)^x \bmod p$$

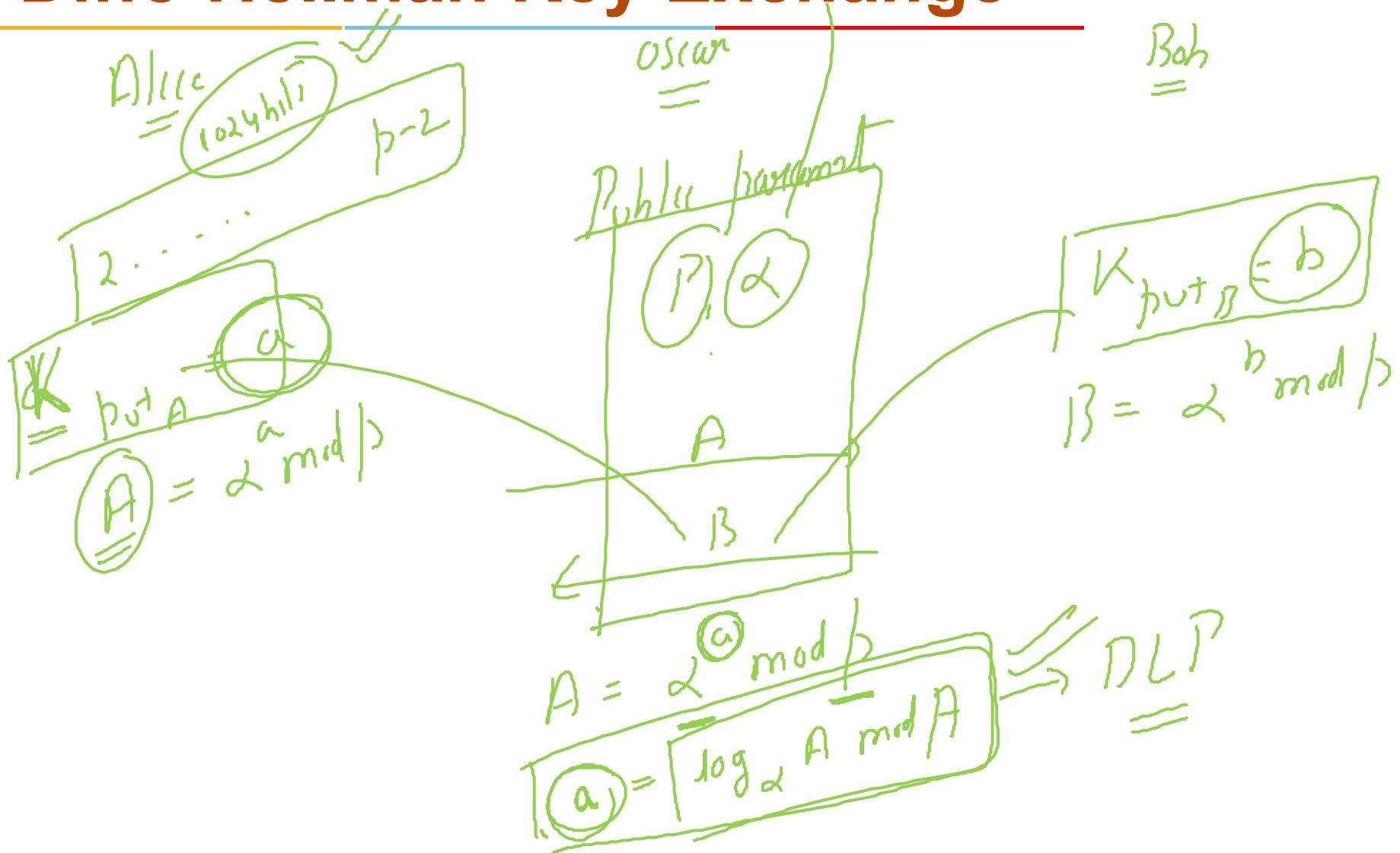
- Set-up protocol:

- Choose a large prime  $p$ .
- Choose integer  $a$  element of  $\{2, 3, \dots, p-2\}$ .
- Publish  $p$  and  $a$ .

# Diffe-Hellman Key Exchange



# Diffe-Hellman Key Exchange

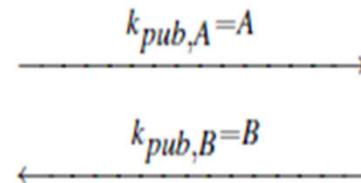




# Diffe-Hellman Key Exchange

A

choose  $a = k_{pr,A} \in \{2, \dots, p-2\}$   
compute  $A = k_{pub,A} \equiv \alpha^a \pmod{p}$



$$k_{AB} = k_{pub,B}^{k_{pr,A}} \equiv B^a \pmod{p}$$

choose  $b = k_{pr,B} \in \{2, \dots, p-2\}$   
compute  $B = k_{pub,B} \equiv \alpha^b \pmod{p}$

$$k_{AB} = k_{pub,A}^{k_{pr,B}} \equiv A^b \pmod{p}$$



---

Thanks!!!  
Queries?

# Malware Analysis and Detection

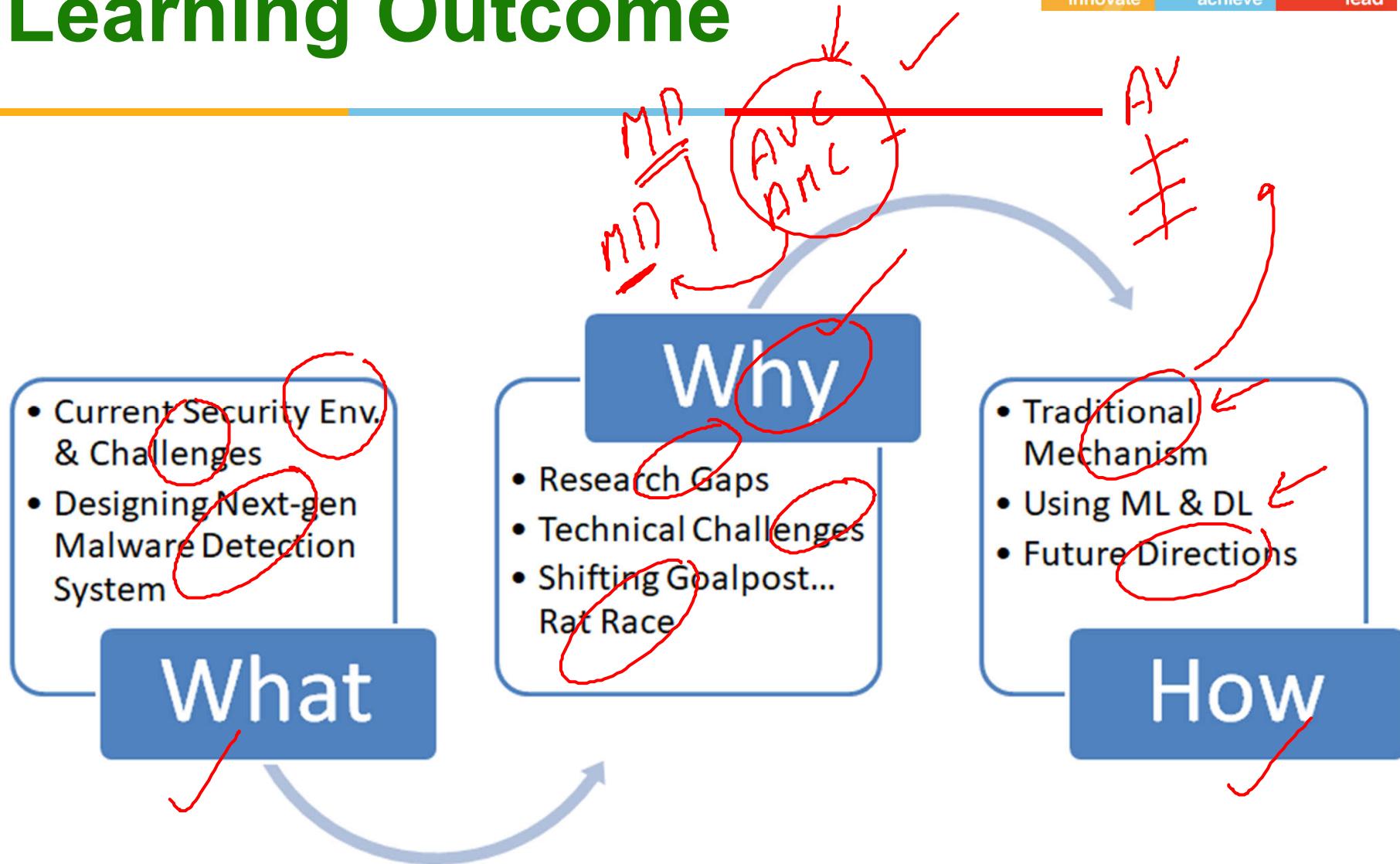


**BITS Pilani**  
K K Birla Goa Campus

Hemant Rathore  
Dept. of CS & IS  
Birla Institute of Technology & Science,  
Pilani

K . K. Birla Goa Campus, India  
[hemantr@goa.bits-pilani.ac.in](mailto:hemantr@goa.bits-pilani.ac.in)

# Learning Outcome





# Recent Malware Attacks

## US Airport Websites Hit By Suspected Pro-Russian Cyberattacks

By AFP - Agence France Presse | October 10, 2022

## Cyber attackers disrupt services at French hospital, demand \$10 million ransom

Issued on: 23/08/2022 - 15:58

## President Rodrigo Chaves says Costa Rica is at war with Conti hackers

18 May 2022

The president of Costa Rica says his country is "at war", as cyber-criminals cause major disruption to IT systems of numerous government ministries.

1,440,336 views | Aug 10, 2019, 06:39am

## Google Warning: Tens Of Millions Of Android Phones Come Preloaded With Dangerous Malware

Google confirms some Android devices were infected with malware even before they shipped

## SpiceJet: Passengers stranded as India airline hit by ransomware attack

25 May 2022

BBC

Ferrari falls victim to ransomware attack; 7GB of its internal documents made public

Oct 04, 2022, 12:42 PM(IST)

WION

## Nvidia says hackers stole employee credentials and company data

By Brian Fung, CNN  
Updated 12:42 PM EST, Tue March 1, 2022

October 4, 2019 1:28 PM IST

172 malicious apps with 335M+ installs found on Google Play

Harmful app type	Number of apps	Number of installs
Adware	48	300,600,000+
Subscription Scam	15	20,000,000+
Hidden Ads	57	14,550,000+
SMS Premium Subscription	24	472,000+
Hidden App	7	310,000+
Banking Trojan	1	10,000+
Stalkware	1	10,000+
Fake Antivirus	1	10,000+
Credit Card Phishing	2	200+
Fake Cryptocurrency Exchanges	1	100+
Fake App	15	100+
sum	172	335,962,400+



# Overview

- 
1. Introduction
  2. Preliminaries
  3. Literature Review
  4. Permission based Malware Detection Models using Machine and Deep Learning
  5. Robust Malware Detection Models: Adversarial Attacks Using Q-Learning



# Overview

- 
1. Introduction
  2. Preliminaries
  3. Literature Review
  4. Permission based Malware Detection Models using Machine and Deep Learning
  5. Robust Malware Detection Models: Adversarial Attacks Using Q-Learning

# Malware

- The term **malware** comes from “**Malicious Software**”.
- **Malware** is a software program that meets the **harmful intent** of malware designer/developer. [Bayer et al. 2006]
- **Malware** enters the computer system without users’ authorization and do undesirable actions: [NIST 2011]
  - steal confidential information,
  - deleting/modify/lock files,
  - send spam emails,
  - compromises computers/smart devices,
  - bring down servers,
  - cripple critical infrastructures, etc.
- Malware includes Virus, Worm, Trojan, Spyware, Adware, Spamware, Bots, Rootkit, Ransomware, etc.

# Malware

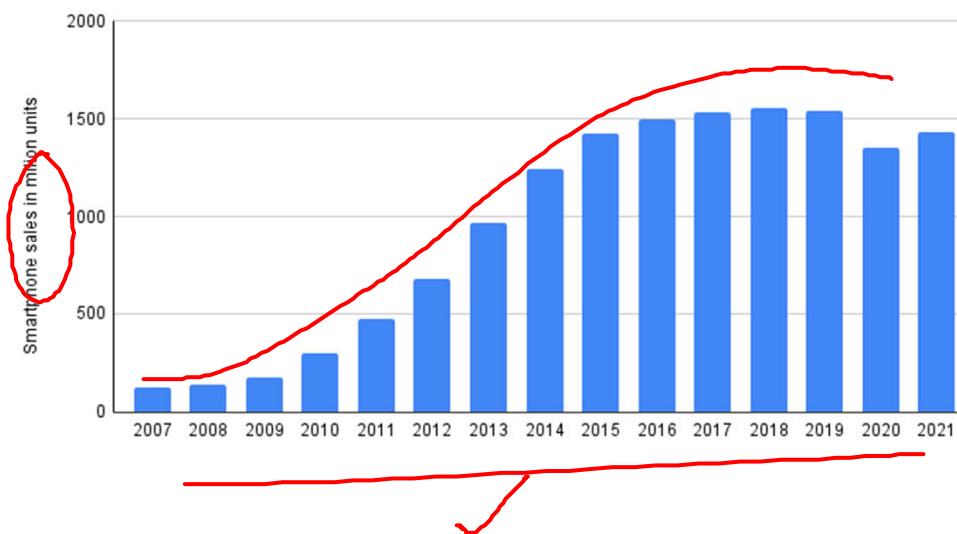
- Initially, malware were developed to show one's tech knowledge and skill or for fun.
  - Creeper just displayed taunting messages: "I'm the creeper: catch me if you can"
- Now, it's a profit-driven industry:
  - "Exploit kits are sold (including source code) at exorbitant prices (\$20-30K). For this reason, users rent them for the limited periods (\$500/month)". [NORSE 2016]
  - Cybercrime will cost \$10.5 trillion annually by 2025. [Forbes 2020]



Source: <https://cybersecurityventures.com/hackerpocalypse-cybercrime-report/>

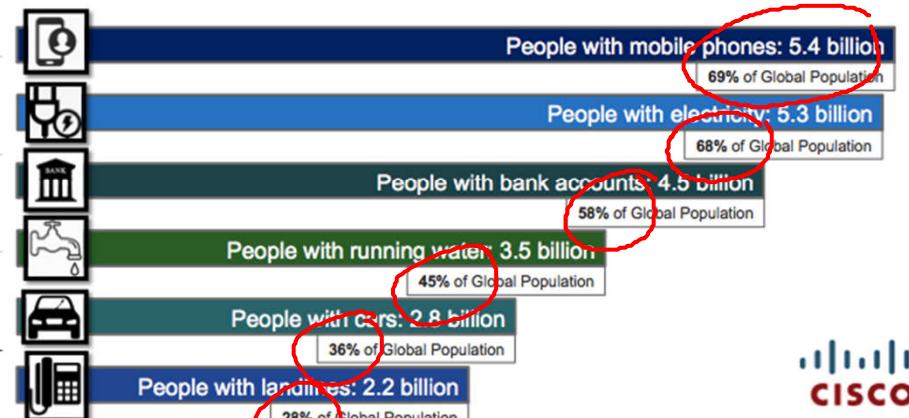
# Smartphone !!! Growth !!!

- Number of smartphones is growing exponentially.
- 1.4 billion smartphones were sold worldwide in 2021 [Statista 2021]
- 5.4 billion unique smartphone users in 2020. [Cisco 2021]



## Mobile Growth Continues Through 2020

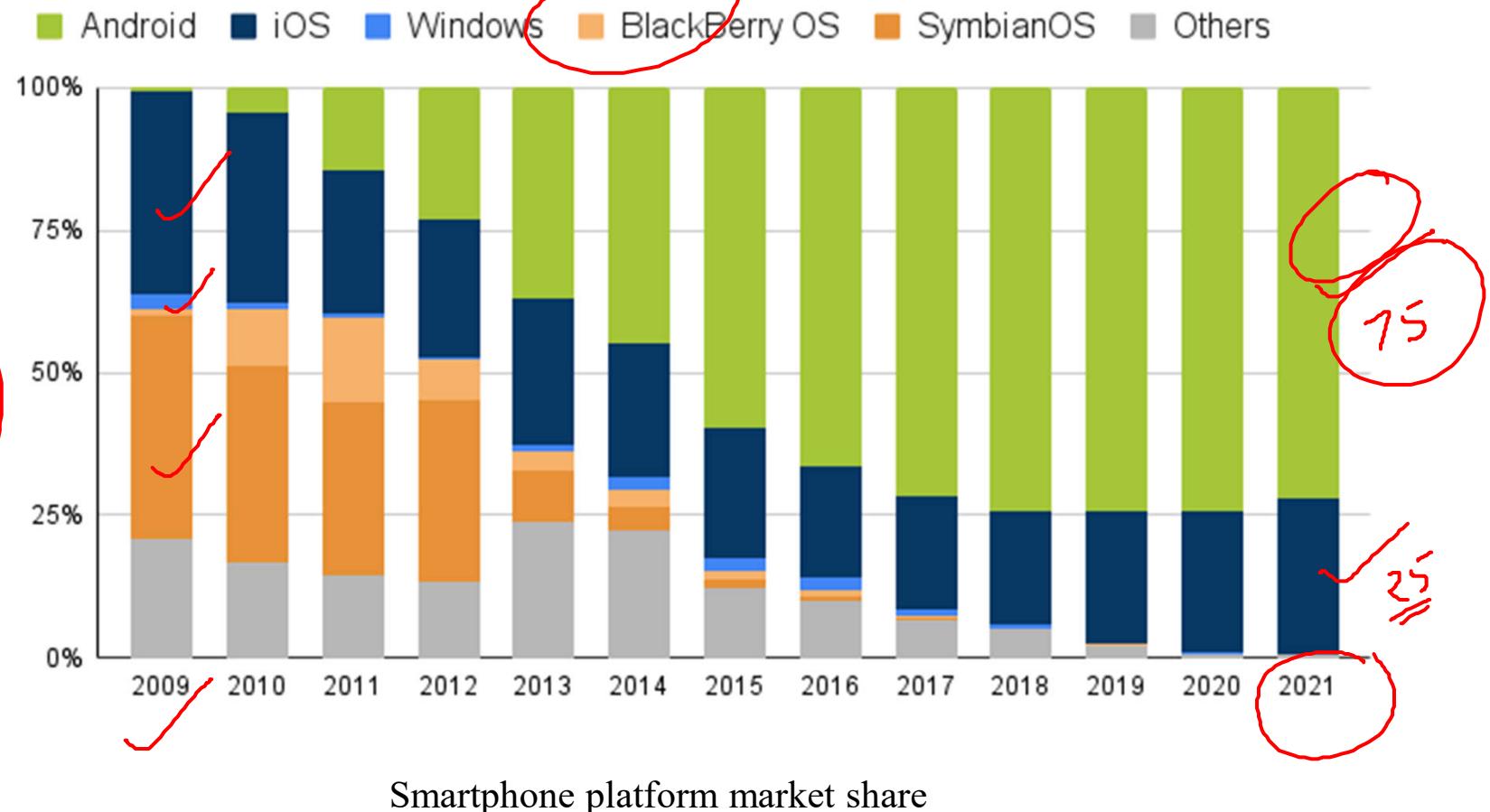
By 2020, more people will have mobile phones than electricity at home



Source L: <https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/>

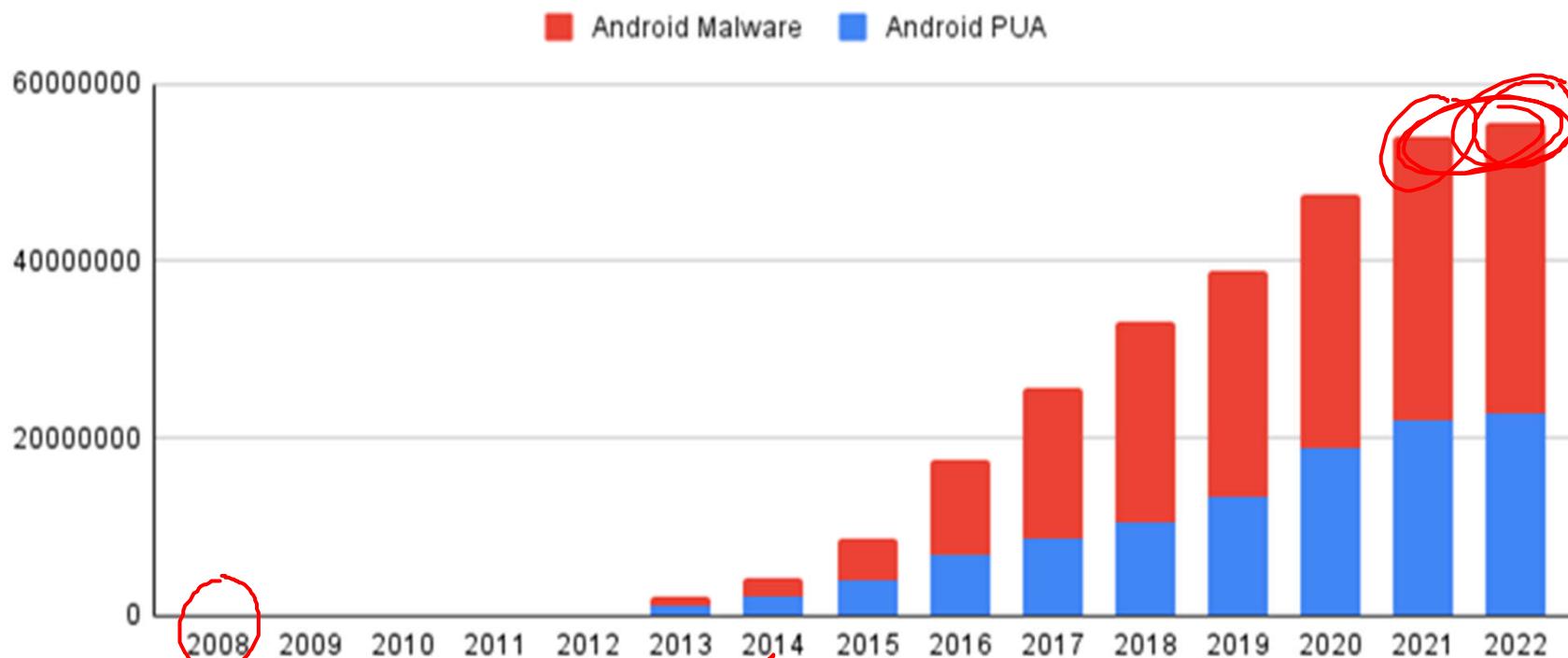
Source R: <https://newsroom.cisco.com/press-release-content?articleId=1741352>

# Smartphone: !!! OS Duopoly !!!



Source: <https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/>

# Malware Growth in Android



Total number of Malware and PUA on Android (More than 50 million in 2021)

\*PUA: Potentially Unwanted Application (e.g., adware, spyware, etc.)

- Google blocked 1.2 million malicious/unwanted apps from Play Store in 2021. [Google 2022]
  - Google banned 190k bad accounts & closed 500k abandoned developer accounts.

Source: <https://portal.av-atlas.org/malware/statistics>

# Research Motivation

- It is impossible to develop a generic algorithm to detect all possible malware. [Cohen, 1987]
  - Currently malware are thriving with 3Vs (volume, velocity and variety). [Lindorfer, 2014]
  - Existing literature suggests that traditional malware detection approaches are unable to cope up with current malware challenges. [Ye et al., 2017]
- Rat Race
- Malware designers:
    - more .. more malware
    - complex/sophisticated malware
  - AV designers: want to detect both old and new malware (especially zero day malware).



# Objectives

Our primary focus is on developing state-of-the-art next-gen effective, efficient and adversarially robust android malware detection models.

The objectives are:

1. To develop effective and efficient android malware detection models based on machine learning and deep learning algorithms.
2. To investigate the adversarial robustness of malware detection models against evasion attacks. Design countermeasures to achieve overall superiority in the rat-race between malware designer and anti-malware community.
3. To explore image-based android malware detection models and investigate their adversarial robustness.



# Overview

- 
- 1. Introduction
  - 2. Preliminaries
  - 3. Literature Review
  - 4. Permission based Malware Detection Models using Machine and Deep Learning
  - 5. Robust Malware Detection Models: Adversarial Attacks Using Q-Learning



# Artificial Intelligence

- Artificial Intelligence is a form of applied statistics that learns from data and perform tasks using the information it has learned.
- Supervised Learning
  - given: data points and corresponding labels.
  - design: function which fits the data with minimum error.
  - perform: prediction for unlabelled data.
  - example: classification, regression, etc.
- Unsupervised Learning
  - given: unlabelled data.
  - design: function to find pattern and structure in the unlabeled data.
  - example: clustering, association rule mining, etc.
- Performance Metric
  - used to measure performance of a model.

# Classification Algorithms

- Classical Machine Learning
  - Decision Tree
  - Support Vector Machine
  - k Nearest Neighbour ....
- Bagging based Learning
  - Bagged Decision Tree
  - Random Forest
  - Extra Tree ....
- Boosting based Learning
  - Adaptive Boosting
  - Gradient Boosting
  - eXtreme Gradient Boosting ....
- Deep Learning
  - Deep Neural Network
  - Convolutional Neural Network ....

# Performance Metrics

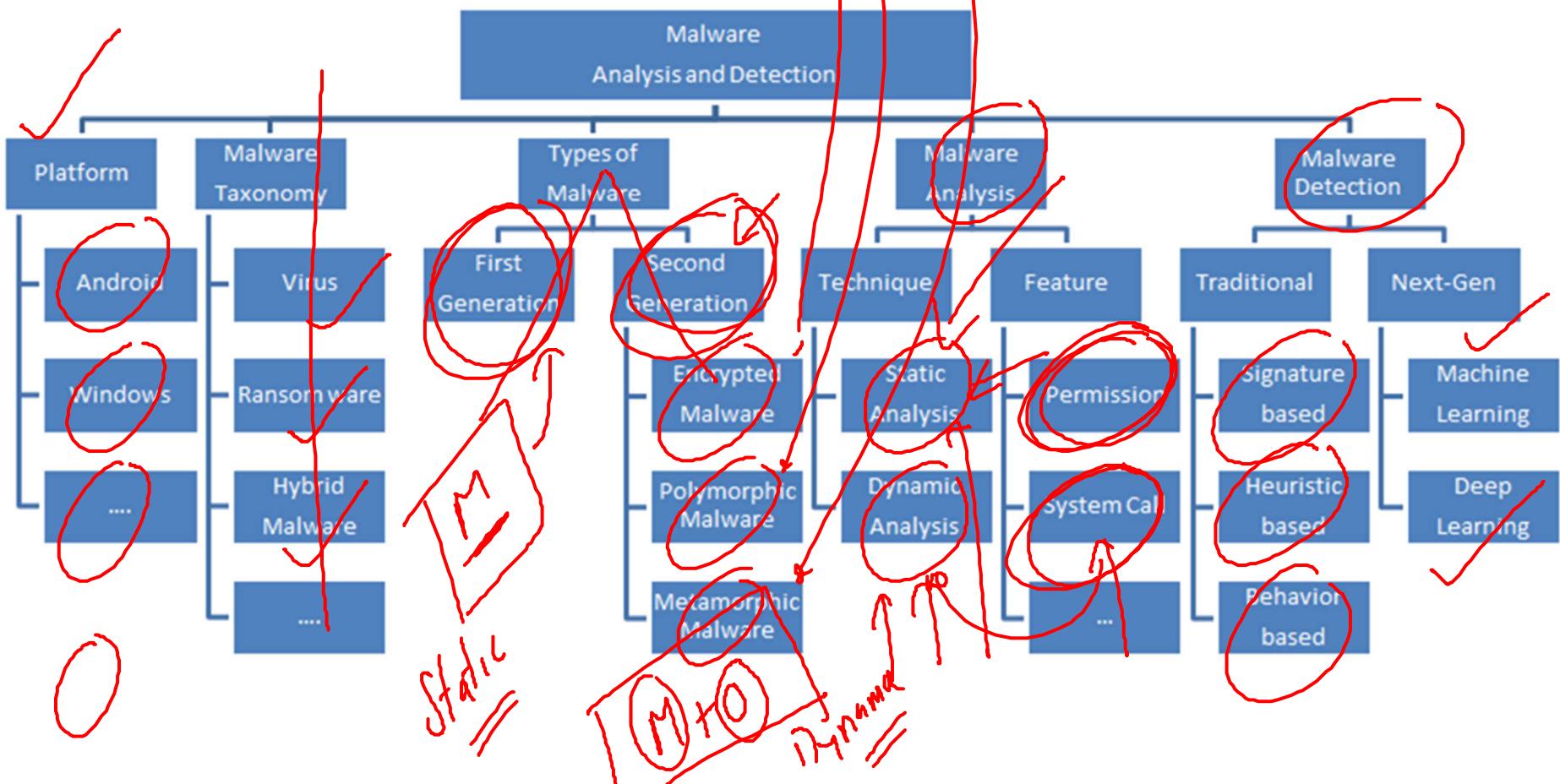
- True Positive (TP)
    - correct prediction
    - "malware detected as malware".
  - True Negative (TN)
    - correct prediction
    - "benign detected as benign".
  - False Positive (FP)
    - incorrect prediction
    - "benign detected as malware".
  - False Negative (FN)
    - incorrect prediction
    - "malware detected as benign".
- 
- Accuracy
    - percentage of correct predictions to the total number of predictions.
- Accuracy =  $\frac{TP + TN}{TP + TN + FP + FN} \times 100$*
- 
- Fooling Rate
    - a.k.a Forced Misclassification Rate
- Fooling Rate =  $\frac{M' - FN}{M - FN}$*
- where  $M'$  is the number of adversarial malware applications and  $M$  is the number of malware applications.
- $M \rightarrow AA$



# Overview

- 
- 1. Introduction
  - 2. Preliminaries
  - ~~3. Literature Review~~
  - 4. Permission based Malware Detection Models using Machine and Deep Learning
  - 5. Robust Malware Detection Models: Adversarial Attacks Using Q-Learning

# Overview



# Static Malware

- Virus is a piece of code that attaches itself to host (benign) program and is activated by host program. [Spafford 1989]
- Worm is a standalone malicious program which can run independently. [Spafford 1989]
- Trojan is a software program that pretends to be useful but performs malicious actions in the background with user's knowledge or consent. [Schultz et al. 2001]
- Spyware is a type of malicious program that spies on user activities without the users' knowledge or consent. [Borders and Prakash 2004]
  - Spam-ware, Adware are on same lines.
- Ransomware installs covertly on a victim's computer and executes a crypto-virology attack that adversely affects it.
- Hybrid malware combines two or more other forms of malicious codes into a new type to achieve more powerful attack functionalities.

# Obfuscation Techniques

- Garbage Code Insertion
  - “nop” instruction
- Compression
  - while storing the malware, compress it
- Control / Data Flow Permutation
  - Jump Instruction Insertion
- Package, Class or Method Renaming
- String / Class / Resource Encryption
- Using Reflection APIs





# Malware Detection System

- Signature-based Detection Technique
- Heuristic-based Detection Technique
- Machine Learning and Deep Learning based Technique

# Signature-based Malware Detection Technique

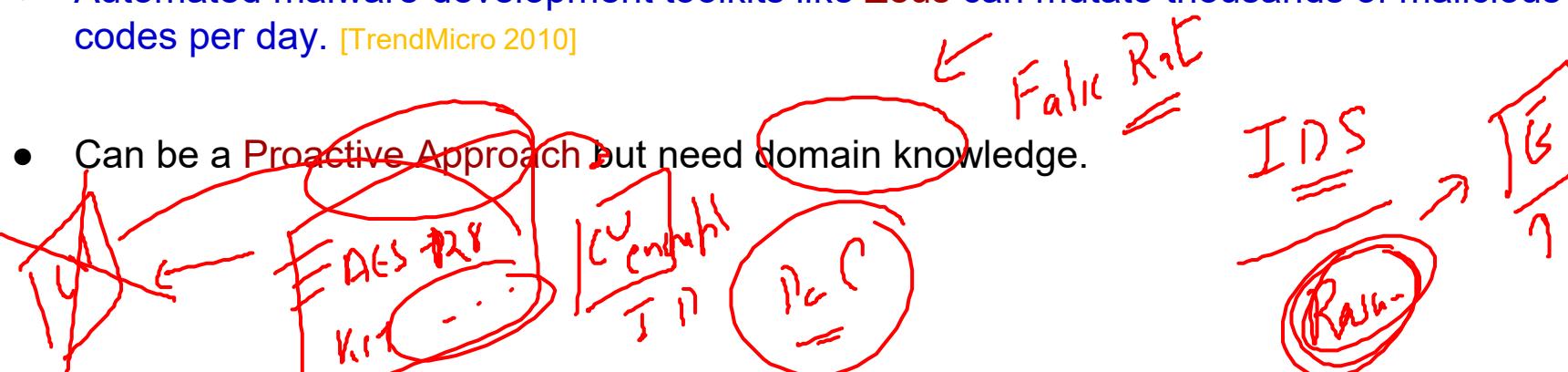
- Signature unique strings that identifies a malware sample. [Moskovich et al. 2009]
- Signatures are often manually generated, disseminated, and maintained by domain experts.  
*human drive*
- Typical time window between a malware's release and its detection by anti-malware software is about 54 days. 15% of samples are still undetected even after 180 days.
- Malware can easily bypass signature-based identification by changing small pieces of its code without affecting the semantics. [Rastogi et al. 2013]
- Reactive Approach (no savior against zero day attack)

Malware

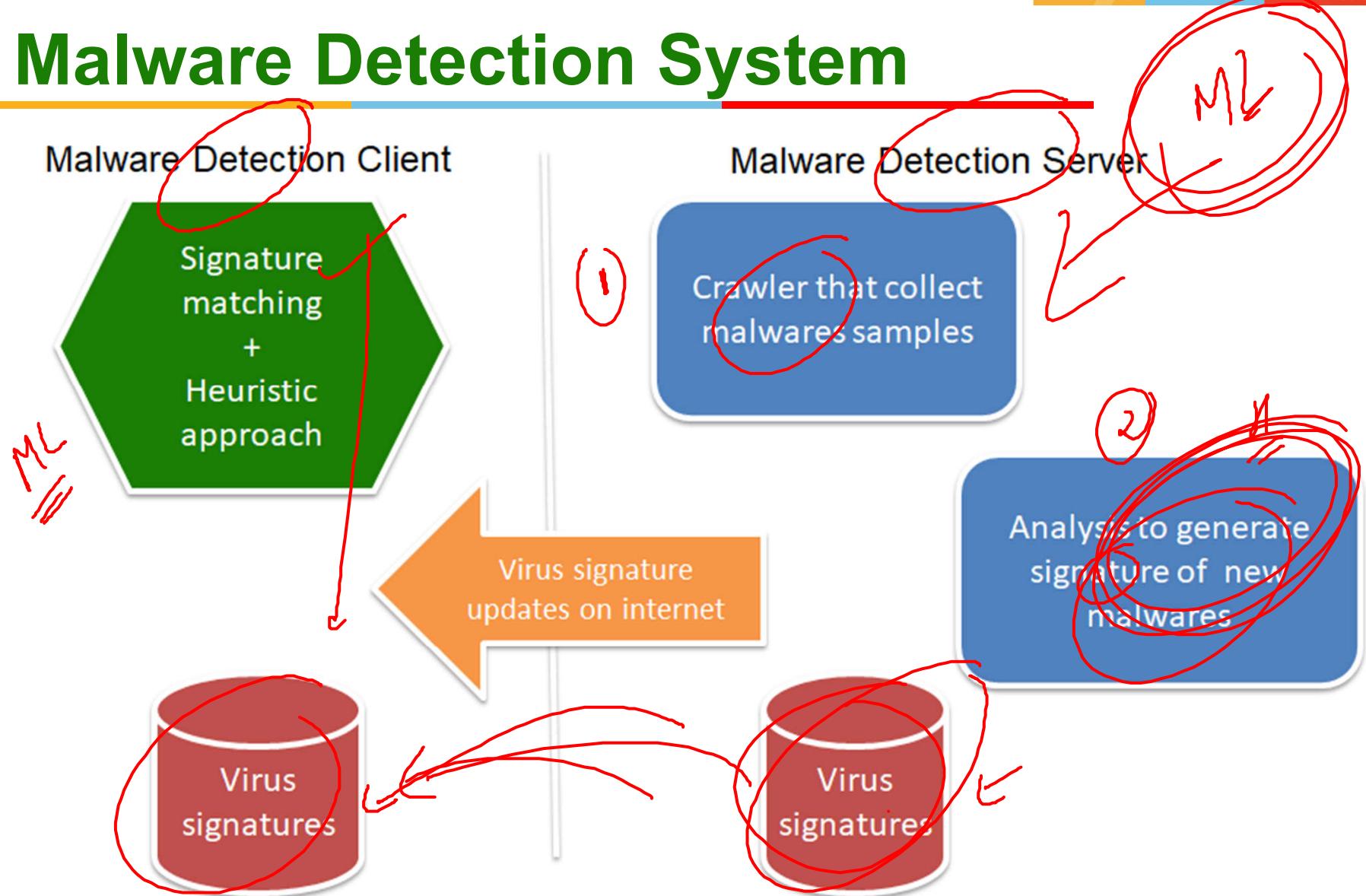
Signer

# Heuristic-based Malware Detection Technique

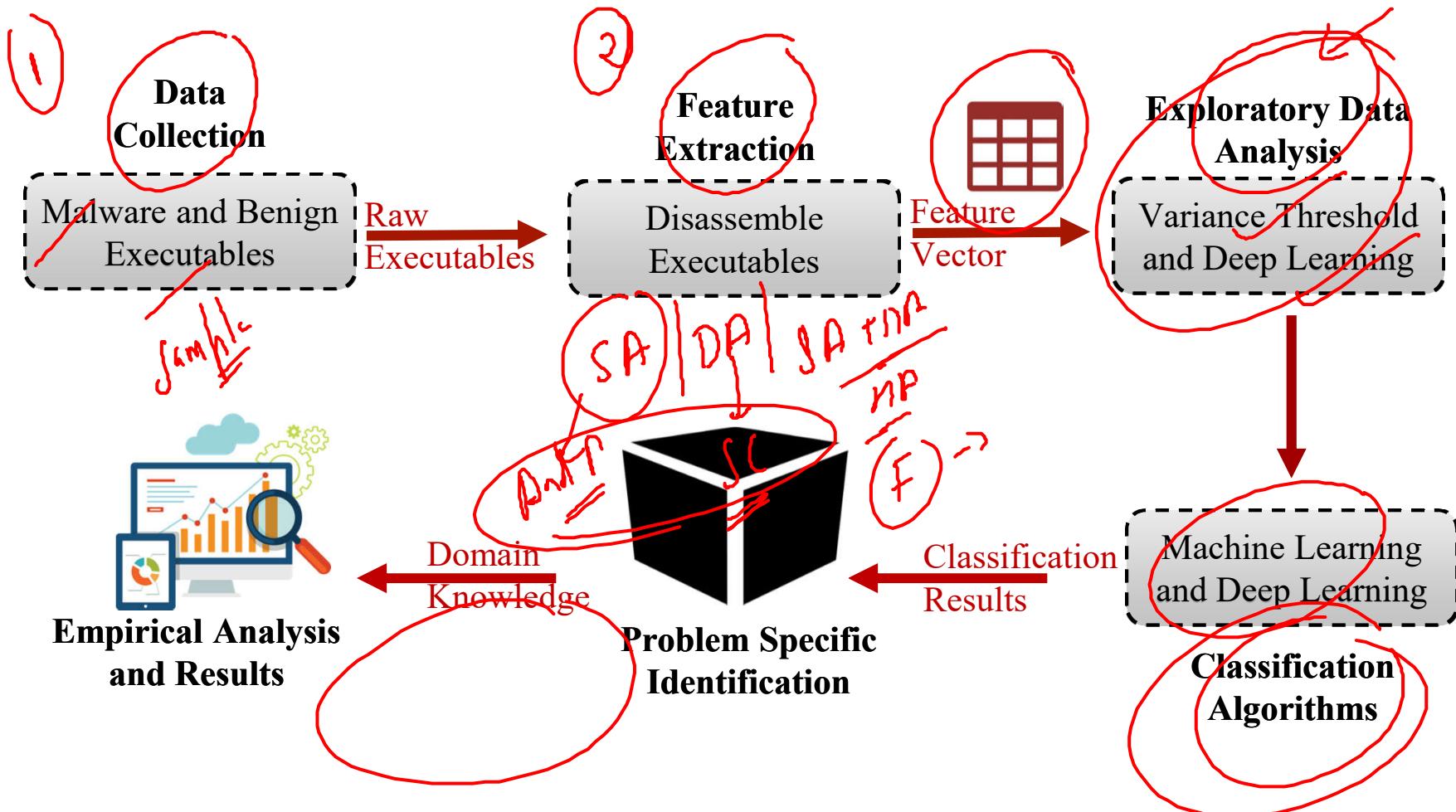
- Encrypted, polymorphic, and metamorphic malware can easily bypass the signature-based detection.
- So domain experts make rules/patterns to discriminate malware and benign files.
- Rules/patterns should be generic to detect variants of the same malware family, but not benign files. [Egele et al. 2012]
- Automated malware development toolkits like Zeus can mutate thousands of malicious codes per day. [TrendMicro 2010]



# Traditional Malware Detection System



# Malware Detection Models using Machine and Deep Learning





# Overview

- 
1. Introduction
  2. Preliminaries
  3. Literature Review
  4. Permission based Malware Detection Models using Machine and Deep Learning
  5. Robust Malware Detection Models: Adversarial Attacks Using Q-Learning

# Research Questions

- ① • Research Question-1: Do malware designers use specific android permission set to perform malicious activities?
- ② • Research Question-2: Can a reduced android permission set be constructed for efficient android malware detection?
- ③ • Research Question-3: Do android malware detection model(s) based on machine learning tend to use less computational resources as compared to the deep learning algorithms?

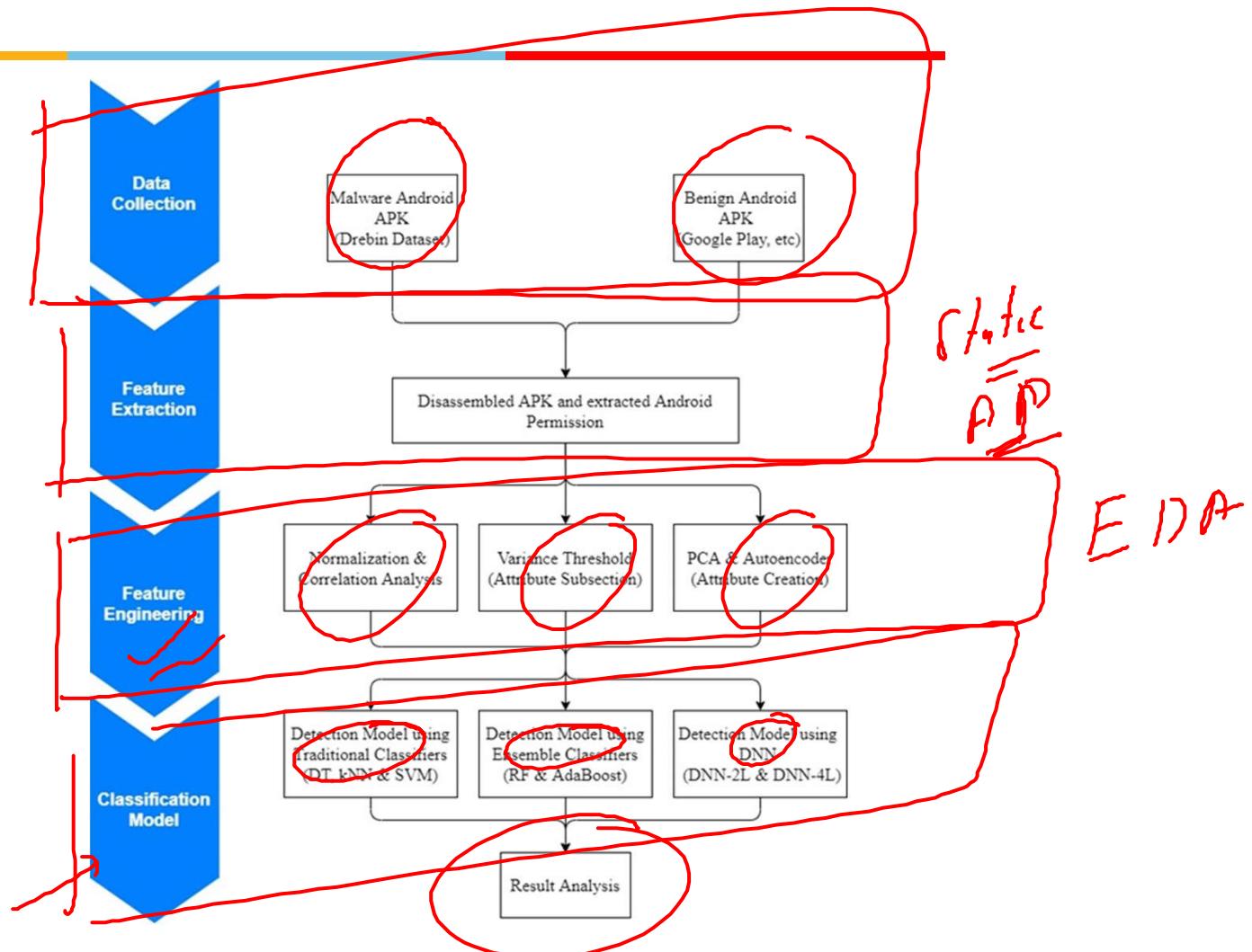
MD  
ML  
TCC  
MD  
(DL)

# Problem Definition

- Given a dataset ( $D$ ) containing  $n$  android applications:
  - subset  $M$  contain malware applications,
  - subset  $B$  contain benign applications,
  - and  $|M| \approx |B|$
$$D = \{(App_i, Class_i), \forall i = 1 \dots n\}$$

where  $App_i$  represents  $i^{th}$  android application and  $Class_i$  corresponds to the class label of the application. The  $Class_i \in \{0, 1\}$  where 0 signifies benign and 1 signifies malicious application.
- An android application can be represented using features (permission, intent, etc.).
- Malware detection model ( $f$ ) can be constructed using feature and various machine learning and deep learning algorithms.
- The performance of the malware detection model ( $f$ ) can be used to measured using evaluation metrics like accuracy, FP, TP, AUC etc.

# Framework Design



A schematic for the construction of an efficient and effective android malware detection system.

# Dataset (Malware)

- Drebin Dataset: Daniel Arp and others downloaded more than one lakh android applications from Google Play Store and other sources.
- VirusTotal service to label each application (malware/benign).
  - VirusTotal is a subsidiary of Alphabet Inc., which aggregates the result of many antivirus products.
- Contains 5,560 malicious android applications.
- Include samples from Android Malware Genome Project.
- Samples from malware families like FakelInstaller, DroidKungFu, GoldDream, etc.
- Drebin is a benchmark dataset of android malware applications with more than 2200 citations.



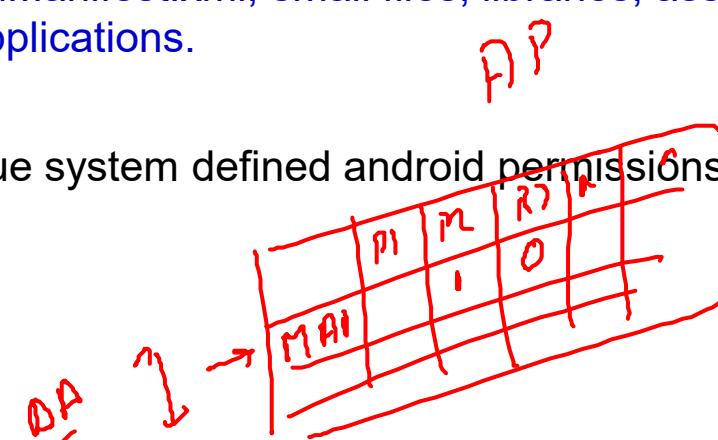
# Dataset (Benign)

- We downloaded ~8,000 android applications from the Google Play Store.
- Verified each downloaded application using VirusTotal.
  - virustotal.com: aggregator of 50+ antivirus software.
  - Android application was labelled benign ONLY when all AV declare it benign.
  - Malicious applications were discarded.
- Final Dataset contains:
  - Benign Android Applications: 5,721
  - Malware Android Applications: 5,560

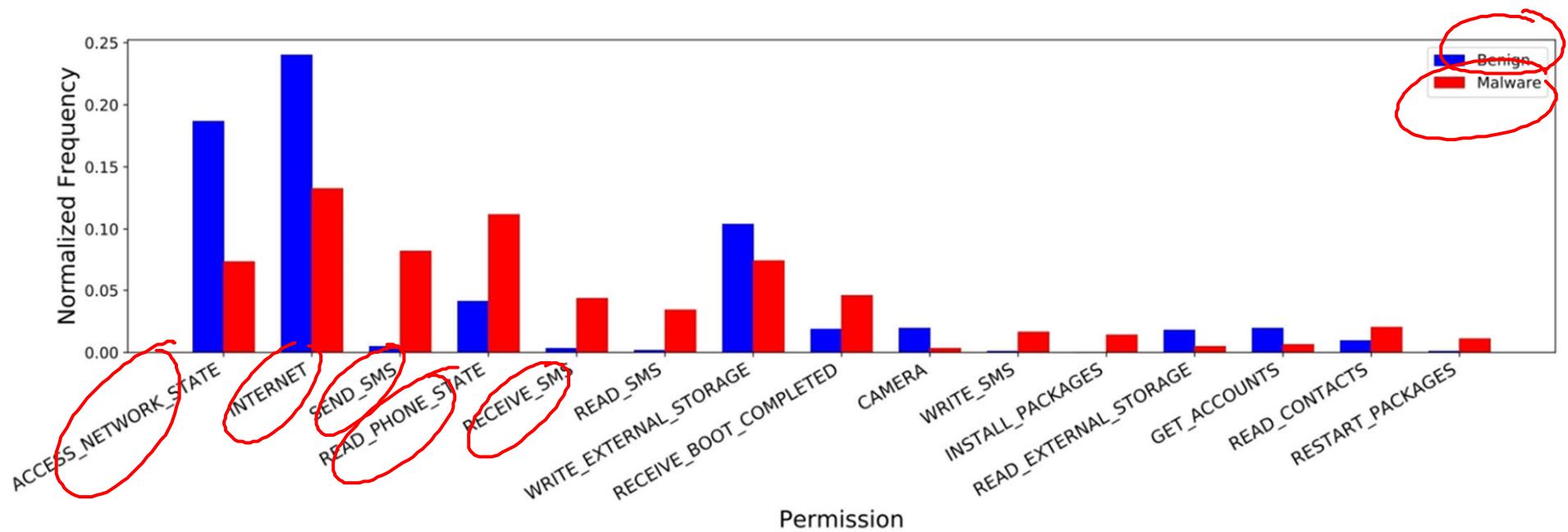
AMD  
K  
DL  
T  
B  
W  
C  
AD  
P  
N  
R  
M  
X

# Feature Extraction

- Total: 11,281 android applications (malware and benign).
- Principal security protection on android platform is derived from the android permission system.
- Static Analysis to extract android permissions for malware detection.
- Disassemble android applications using Apktool.
  - android application (.apk) to AndroidManifest.xml, smali files, libraries, assets, etc.
  - Removed corrupted or encrypted applications.
- Master permission list contains 197 unique system defined android permissions.
- Final android permission feature vector:
  - Benign: 5,721 X 197
  - Malware: 5,560 X 197

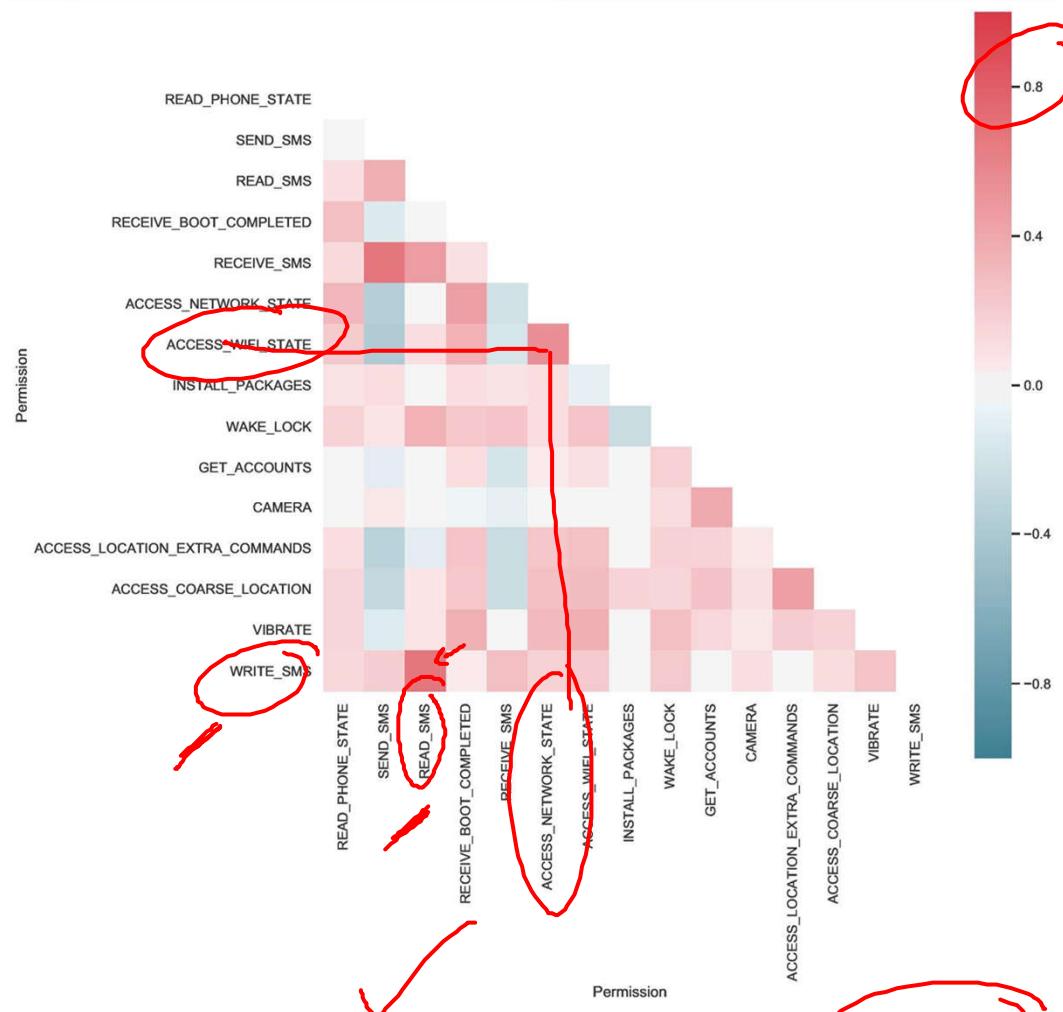


# Permission Usage



Normalized frequency of the permission usage in malicious and benign applications.

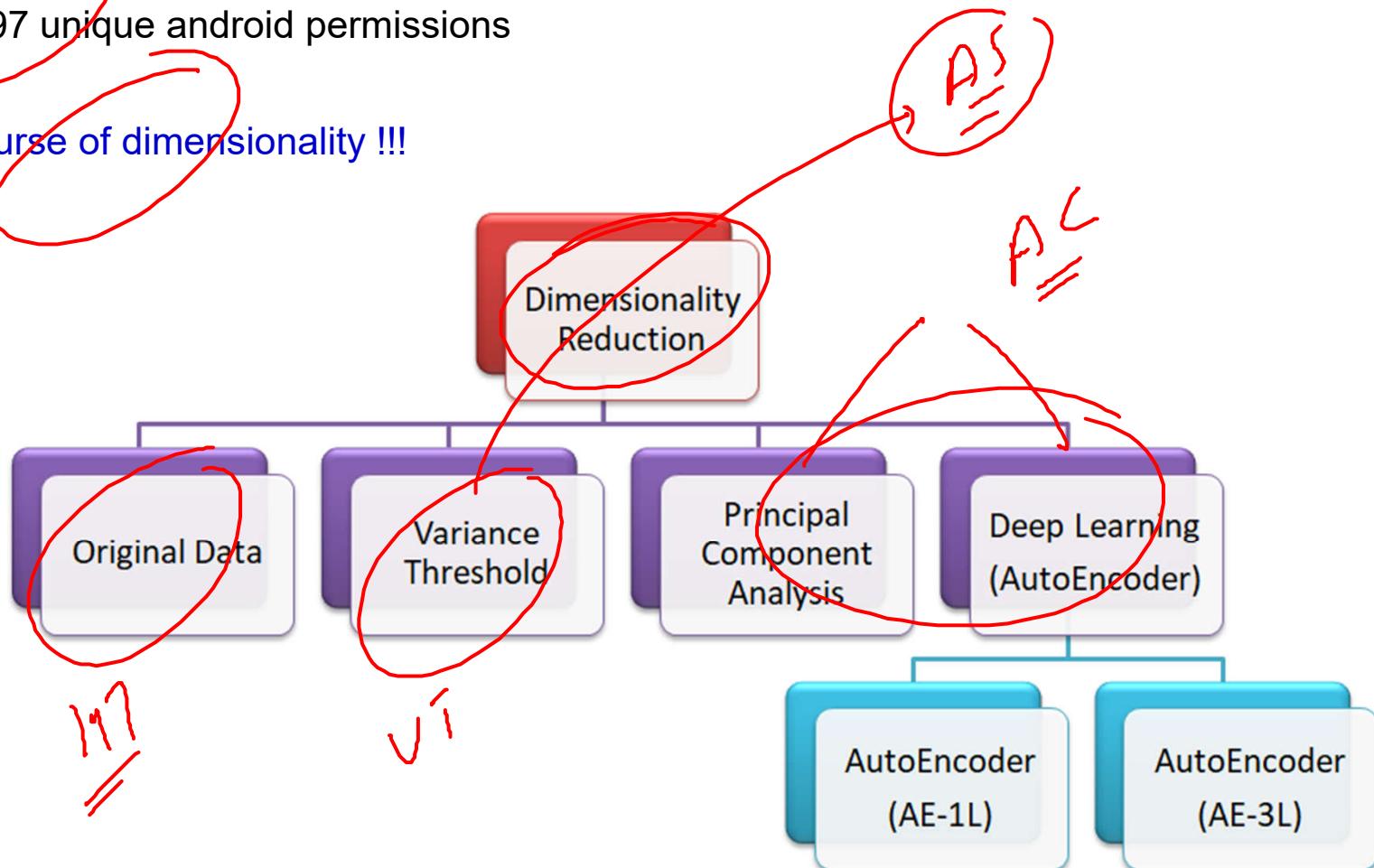
# Correlation Grid



Correlation matrix of permissions used in the malicious applications.

# Dimensionality Reduction

- 197 unique android permissions
- Curse of dimensionality !!!



# Feature Reduction

Feature Reduction	Number of features
OD <sup>1</sup>	197
VT <sup>2</sup>	16
PCA <sup>3</sup>	16
AE-1L <sup>4</sup>	64 (code layer)
AE-3L <sup>5</sup>	16 (code layer)

<sup>1</sup>Original Data <sup>2</sup>Variance Threshold <sup>3</sup>Principal Component Analysis

<sup>4</sup>AE with 1 Layer <sup>5</sup>AE with 3 Layers

Feature reduction technique and corresponding reduced vector size.



# Classification Algorithms

- Detection Model using Traditional Classifiers
  - DT, kNN, SVM
- Detection Model using Ensemble Classifiers
  - RF, AdaBoost
- Detection Model using DNNs
  - DNN-2L, DNN-4L, DNN-7L

# Performance Analysis

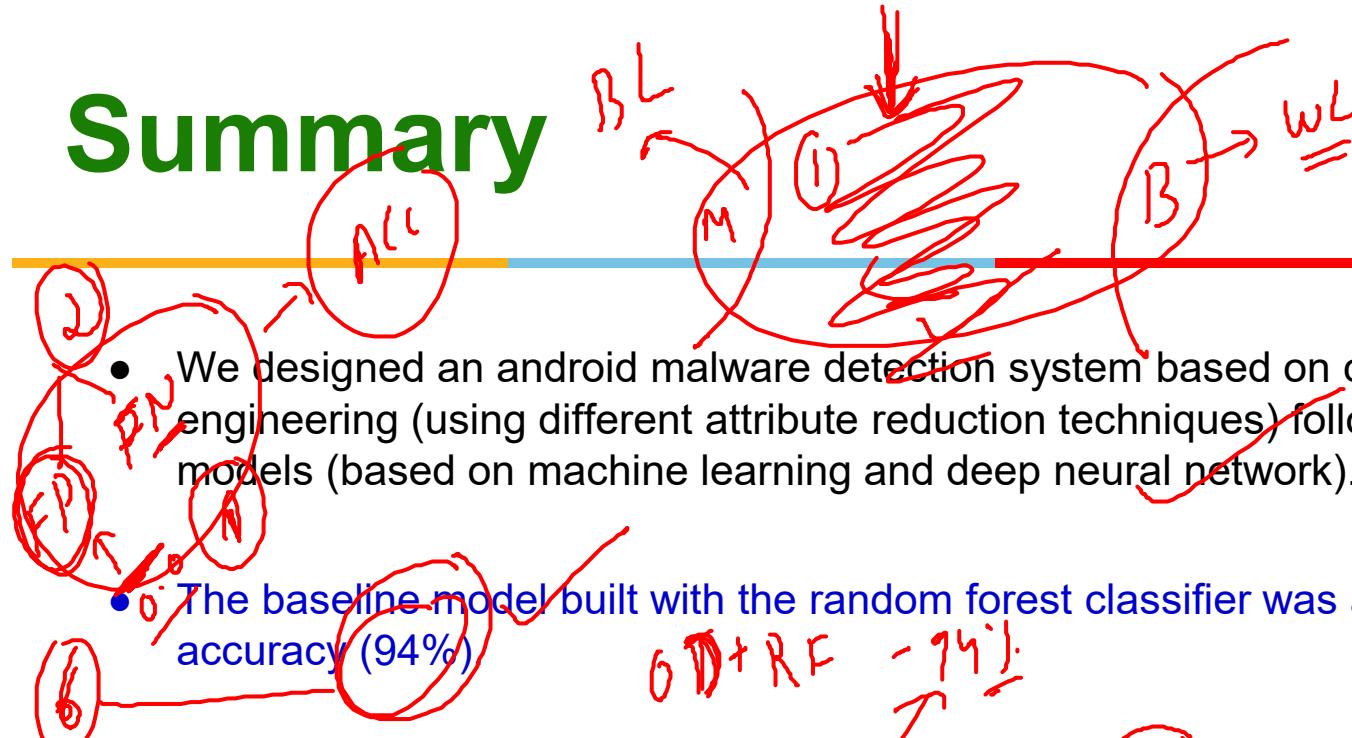
Feature Reduction	Classifier	Accuracy	TPR	AUC	Train Time (sec)	Test Time (sec)
OD <sup>1</sup>	DT <sup>6</sup>	92.6	91.9	94.5	0.052	0.003
VT <sup>2</sup>	DT	92.3	90.9	94.7	0.012	<b>0.001</b>
PCA <sup>3</sup>	DT	91.2	90.7	92.6	0.059	0.002
AE-1L <sup>4</sup>	DT	91.4	91.1	92.7	0.258	0.003
AE-3L <sup>5</sup>	DT	90.5	89.8	91.6	0.476	0.002
OD	kNN <sup>7</sup>	78.7	<b>97.8</b>	84.2	0.133	4.044
VT	kNN	78.4	96.4	84.5	0.080	0.421
PCA	kNN	91.4	91.3	<b>95.7</b>	<b>0.005</b>	0.119
AE-1L	kNN	90.8	93.0	95.4	0.100	1.950
AE-3L	kNN	88.4	90.3	94.7	0.084	0.327
OD	SVM <sup>8</sup>	91.0	89.4	96.3	5.634	0.929
VT	SVM	89.1	87.3	95.2	1.186	0.116
PCA	SVM	87.3	87.6	94.1	0.624	0.088
AE-1L	SVM	89.5	86.9	95.4	6.372	1.061
AE-3L	SVM	86.6	86.2	93.9	7.200	1.185
OD	RF <sup>9</sup>	<b>94.0</b>	93.0	<b>98.1</b>	0.674	0.036
VT	RF	93.3	92.0	97.7	0.328	0.028
PCA	RF	93.1	91.7	97.6	0.870	0.026
AE-1L	RF	93.2	91.7	97.7	1.281	0.029
AE-3L	RF	91.9	91.1	97.2	2.087	0.028

Feature Reduction	Classifier	Accuracy	TPR	AUC	Train Time (sec)	Test Time (sec)
OD	AdaBoost <sup>10</sup>	90.6	90.6	96.4	1.488	0.063
VT	AdaBoost	89.1	88.7	95.3	0.352	0.031
PCA	AdaBoost	89.9	89.1	96.1	0.828	0.029
AE-1L	AdaBoost	91.1	89.8	96.6	2.881	0.048
AE-3L	AdaBoost	89.0	87.6	95.4	5.029	0.048
OD	DNN-2L <sup>11</sup>	93.0	93.5	93.0	162.624	0.195
VT	DNN-2L	92.6	91.7	92.6	175.512	0.225
PCA	DNN-2L	86.9	86.1	86.9	160.566	0.259
AE-1L	DNN-2L	93.1	91.1	93.2	162.066	0.200
AE-3L	DNN-2L	87.6	87.0	87.6	165.436	0.215
OD	DNN-4L <sup>12</sup>	93.0	93.0	94.2	191.264	0.310
VT	DNN-4L	93.1	92.0	93.2	207.849	0.400
PCA	DNN-4L	90.2	87.3	90.2	207.461	0.434
AE-1L	DNN-4L	93.1	91.0	93.1	193.373	0.333
AE-3L	DNN-4L	88.9	86.0	88.9	200.711	0.355
OD	DNN-7L <sup>13</sup>	93.0	92.8	94.0	254.377	0.521
VT	DNN-7L	92.6	90.5	92.6	269.323	0.664
PCA	DNN-7L	89.5	89.2	89.5	264.253	0.717
AE-1L	DNN-7L	92.5	89.3	92.6	247.947	0.557
AE-3L	DNN-7L	88.3	83.6	88.4	255.443	0.598

$$\begin{aligned}
 197 + RF &= 94.0 \\
 &= 98.1 \\
 197 + RF &= 98.1
 \end{aligned}$$

Performance of various malware detection models based on different classification algorithms with feature reduction techniques.

# Summary



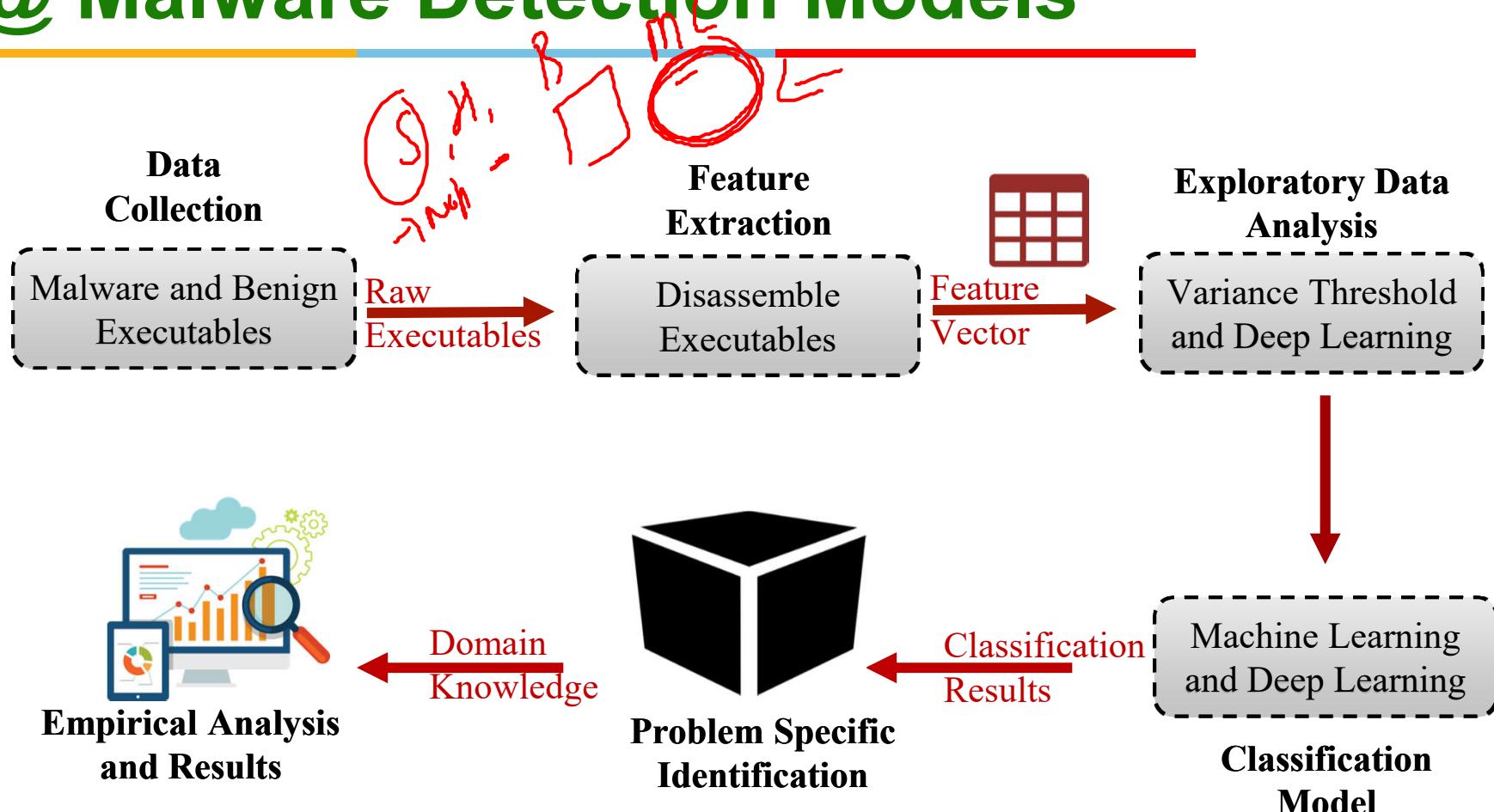
- We designed an android malware detection system based on comprehensive feature engineering (using different attribute reduction techniques) followed by classification models (based on machine learning and deep neural network).
- The baseline model built with the random forest classifier was able to achieve the highest accuracy (94%).  
 $DT + RF - 94\%$
- The reduced feature models constructed with only 8% android permissions attained comparable accuracy with appreciable time-saving. It used only 16 android permissions and achieved an accuracy of 93.3% with random forest classifier (~ 1% less than the baseline model). The decrease in android permissions reduced the train and test time by half and tenth, respectively.
- Deep neural network models achieve comparable accuracy against machine learning models but have a massive computational penalty (hundred and ten times more train and test time respectively compared to random forest models).



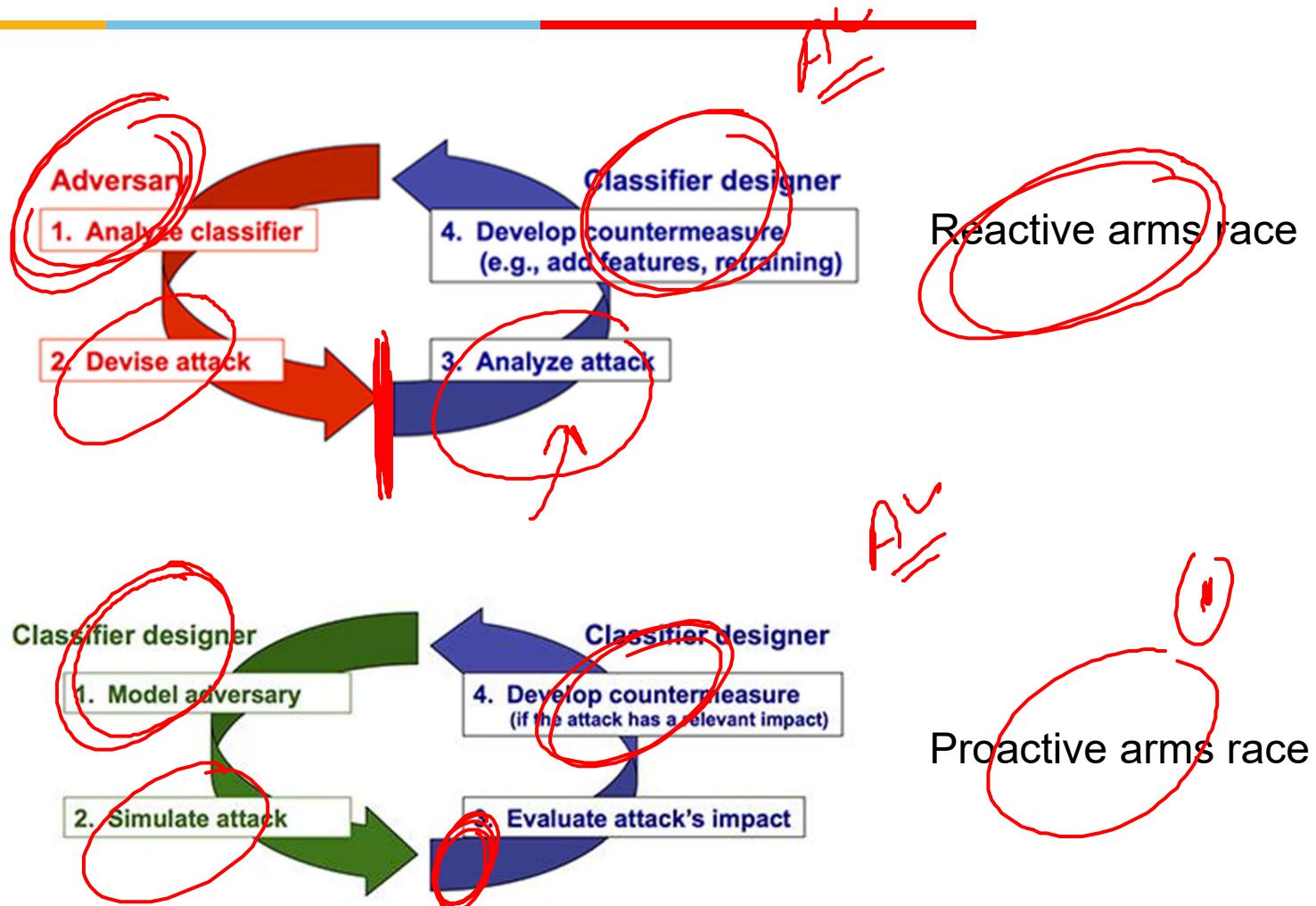
# Overview

- 
- 1. Introduction
  - 2. Preliminaries
  - 3. Literature Review
  - 4. Permission based Malware Detection Models using Machine and Deep Learning
  - 5. Robust Malware Detection Models: Adversarial Attacks Using Q-Learning**

# Framework @ Malware Detection Models



# Rat Race



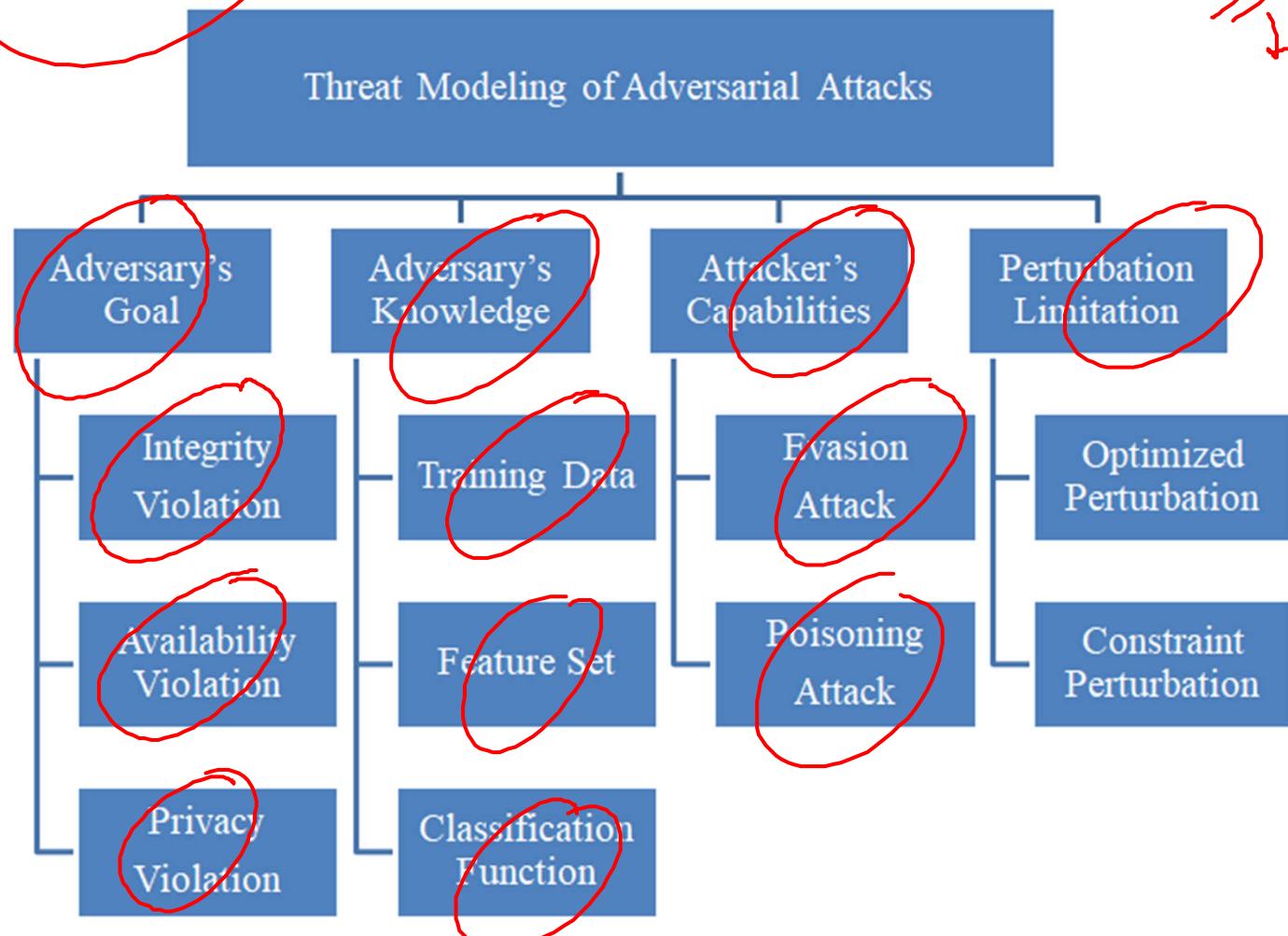
Source: [https://en.wikipedia.org/wiki/Adversarial\\_machine\\_learning](https://en.wikipedia.org/wiki/Adversarial_machine_learning)

# Threat Modeling of Adversarial Attacks



ML | DL

2016



# Problem Definition (continue from Slide-36)

- Android malware detection model can be constructed using machine learning and deep learning algorithms.
- The adversary can aim to attack the above malware detection model ( $f$ ) and reduce its performance.
- The adversary can design a false-negative evasion attack function ( $G$ ) that transforms malware application ( $App_{mal}$ ) into adversarial malware application ( $App_{adv}$ ) by adding noise/perturbation ( $\delta$ ) that can fool / force misclassification in the detection model ( $f$ ).

$$App_{adv} = G(App_{mal}) + \delta$$

$$\text{s.t. } f(App_{adv}) \neq f(App_{mal})$$

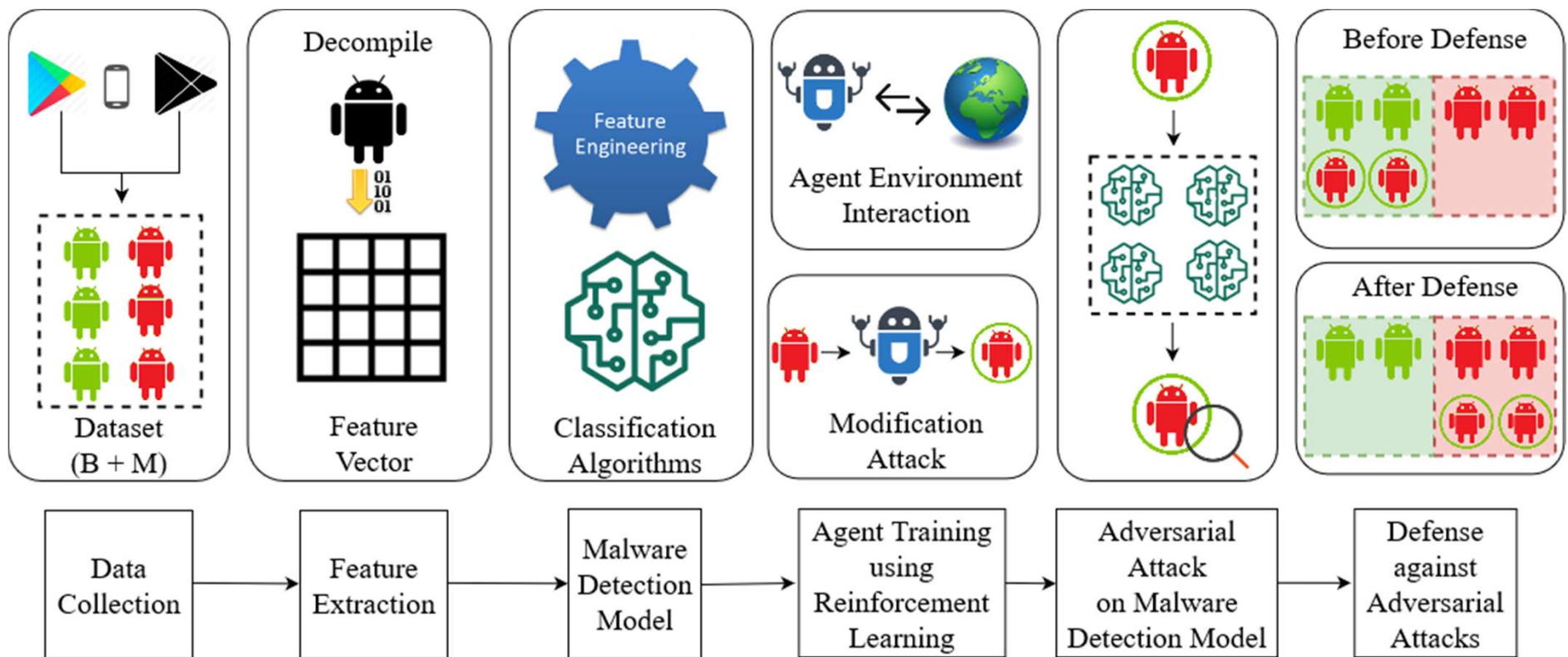
- The adversary can transform many malware applications into adversarial applications and thus drastically reduce the performance of the detection model.
- An adversary-aware anti-malware developer will foresee the above attack scenario(s) and beforehand design defense(s) to counter it.



# Threat Modelling

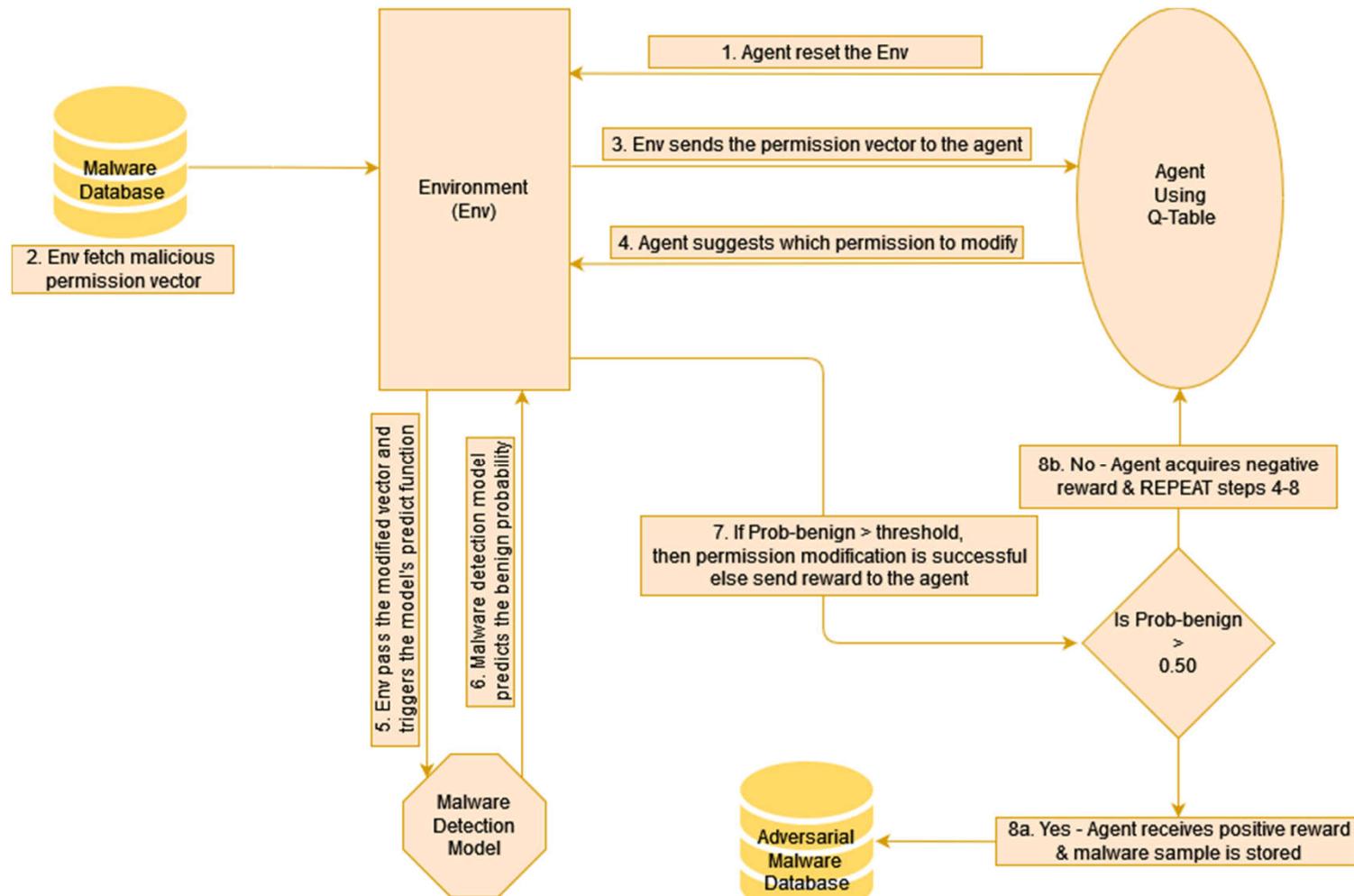
- Adversary's Goal
  - Integrity Attack
- Adversary's Knowledge
  - White Box Scenario (Single Policy Attack)
  - Grey Box Scenario (Multi Policy Attack)
- Adversary's Capabilities
  - Evasion Attack
  - Targeted Attack
  - False-Negative Attack
- Perturbation Limitation
  - Perturbation Optimization
    - Add Minimum Perturbations
  - Perturbation Constraint
    - Maintain App's syntactic, semantic and behavioral integrity

# Framework @ Robust Malware Detection Models



Framework for constructing robust malware detection model(s)  
using adversarial attack and defense strategies.

# Adversarial Attack Policy (SPA)



Adversarial attack on android malware detection system.



# Adversarial Attack Policy (MPA)

---

**Algorithm** Algorithm for multi-policy attack.

---

**Input :**

**X:** permission vector extracted from android application  
**Y:** class label (0 depicts benign and 1 depicts malware)  
**M:** set of malicious sample

**C:** set of malware detection models

**Function :**

**C<sub>P</sub>:** prediction function in classification models

**F<sub>ext</sub>:** feature vector extraction function

**F<sub>m</sub>:** feature vector modification function

**P:** permission to be modified

**Output :** *Sample(s<sub>g</sub>, n<sub>c</sub>, c)*

**s<sub>g</sub>:** modified permission vector ((goal state))

**n<sub>c</sub>:** number to permission modified

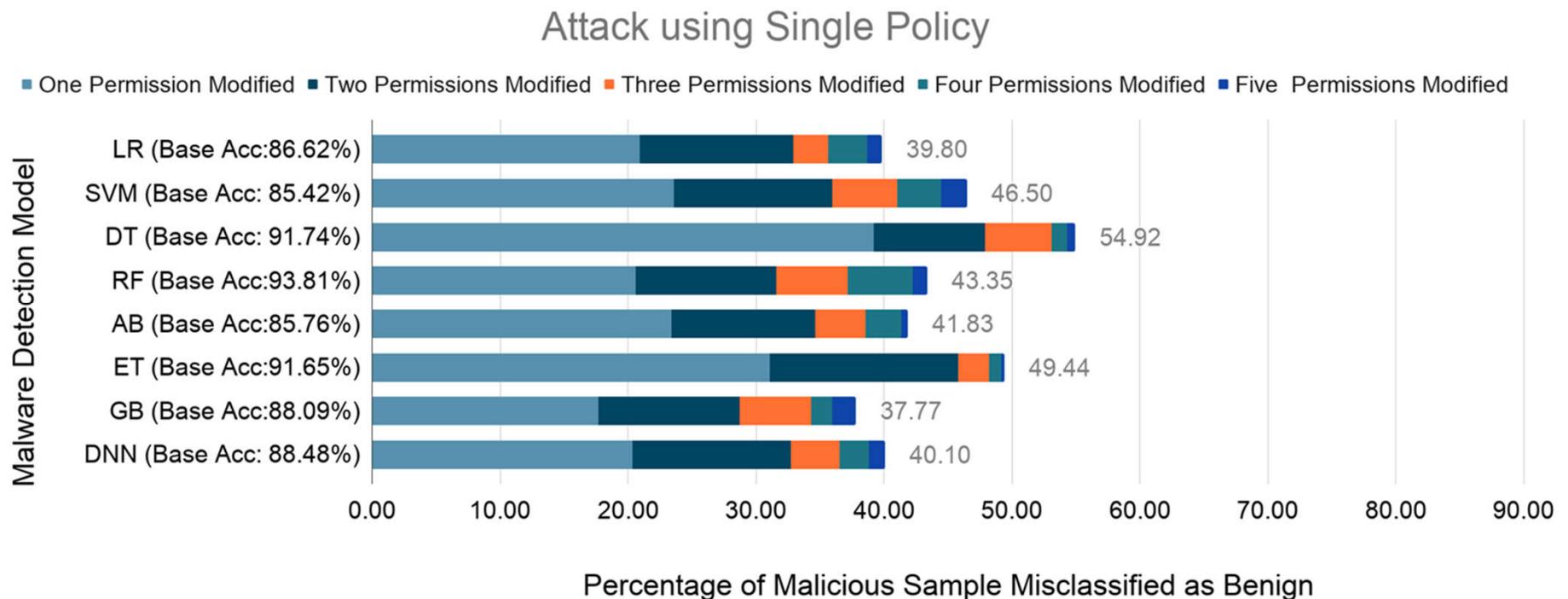
**c:** classification model

```
1: for each malware sample  $d \in M$  do
2:   for each malware detection model  $c \in C$  do
3:     for each policy  $\pi^*$  parallel do
4:        $s \leftarrow F_{ext}(d)$ 
5:        $p \leftarrow \pi^*(s)$ 
6:        $s_{modified} \leftarrow F_m(s, p)$ 
7:        $P_b, n_c \leftarrow C_p(c, s_{modified})$ 
8:       if ( $P_b \geq 0.5$ ) then
9:          $s_g \leftarrow s_{modified}$ 
10:         $Sample(s_g, n_c, c) \leftarrow (s_g, n_c, c)$ 
11:       else
12:         repeat steps 2 - 5 using  $s_{modified}$ 
13:       end if
14:     end for
15:   end for
16: end for
```

---

} in parallel

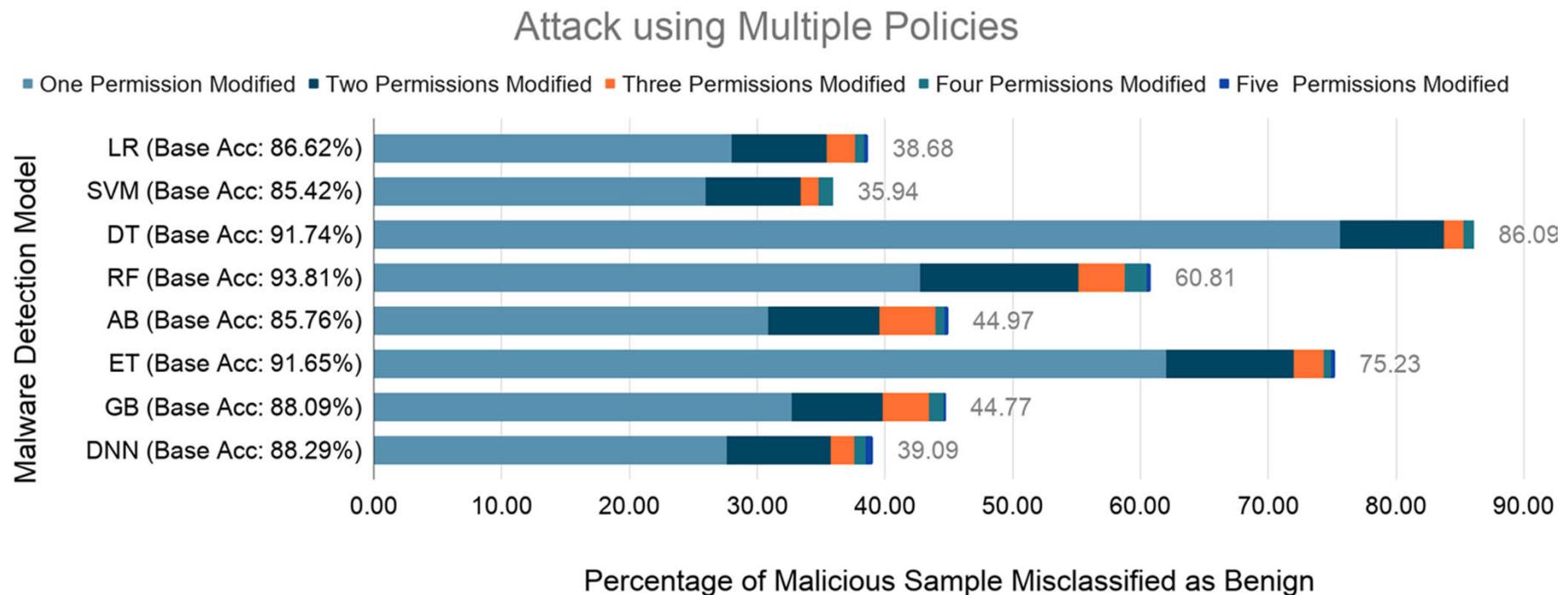
# Attack using Single Policy



Fooling rate achieved by SPA against different malware detection models.

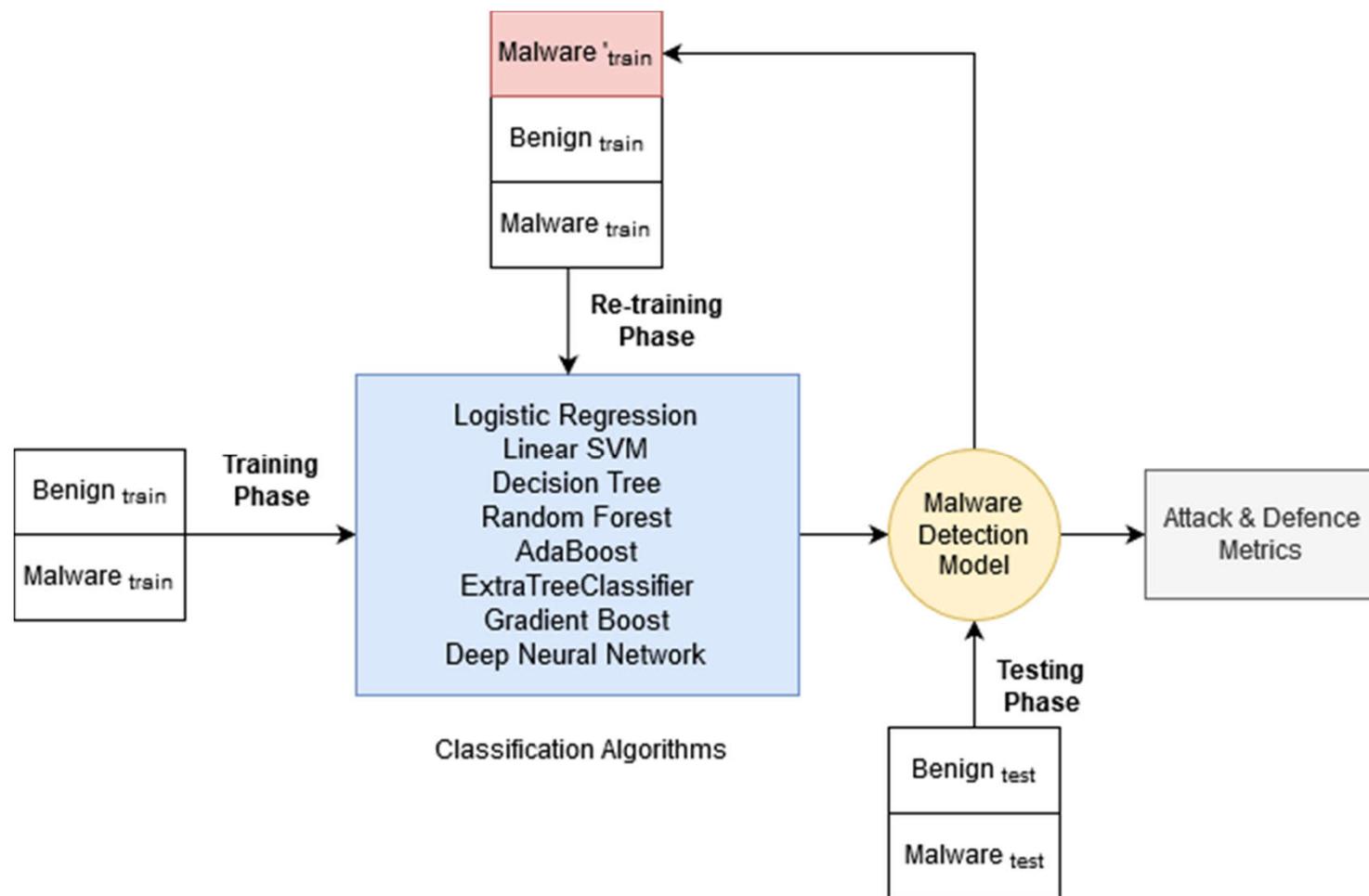


# Attack using Multi Policy



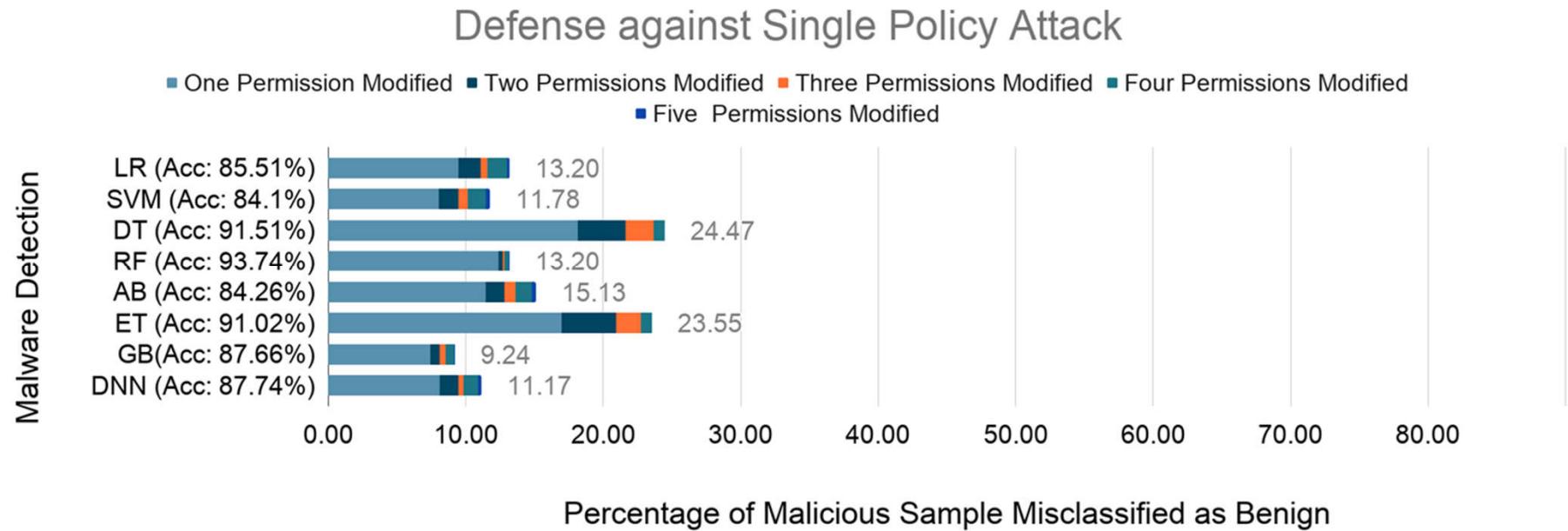
Fooling rate achieved by MPA against different malware detection models.

# Defense Policy



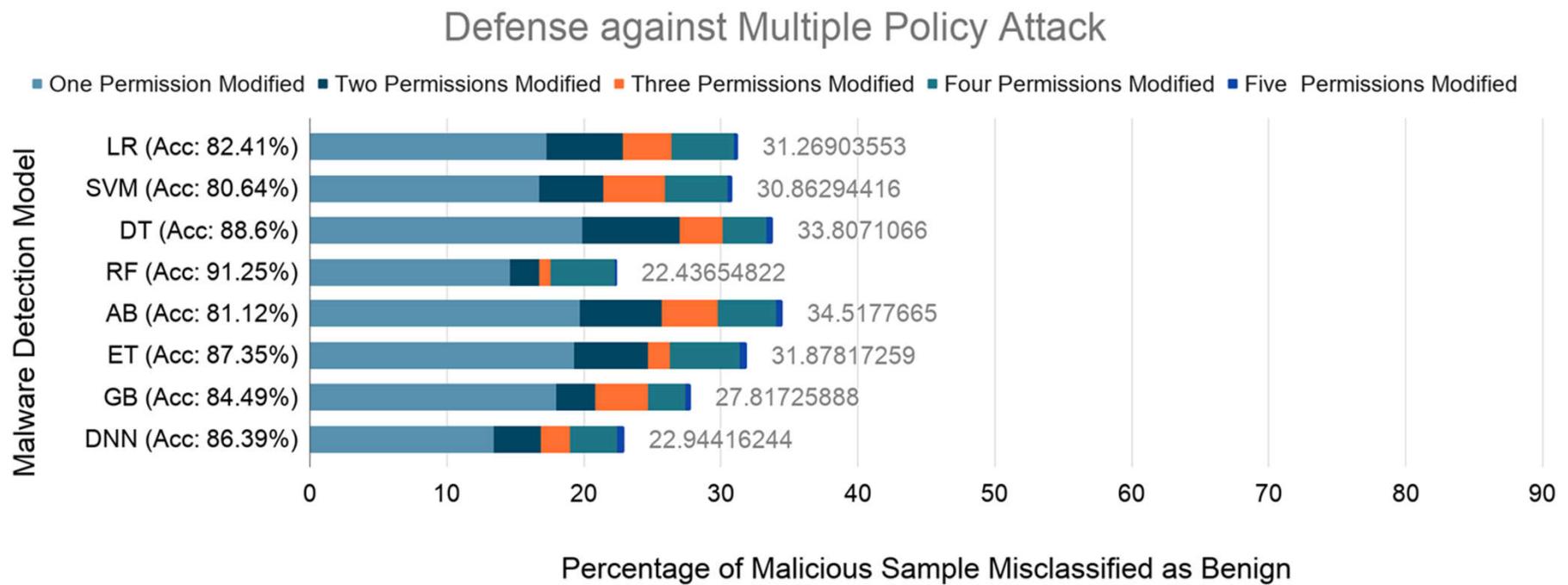
Defense against adversarial attacks.

# Defense against Single Policy Attack



Fooling rate achieved by SPA after the adversarial retraining defense of malware detection models.

# Defense against Multi Policy Attack



Fooling rate achieved by MPA after the adversarial retraining defense of malware detection models.



# Summary

---

- Literature suggests that malware detection models based on machine / deep learning have shown promising results. However, research in other domains suggests that the machine learning / deep learning models may be vulnerable to adversarial attacks.
- We develop a single-policy adversarial attack (SPA) for the white-box scenario, which achieved an average fooling rate of 44.28% with a maximum of five modifications across eight diverse set of detection models.
- We also devised a multi-policy attack (MPA) for the grey-box scenario, which obtained an average fooling rate of 53.20% with a maximum of five modifications across the same eight detection models.
- Finally, we designed a defence mechanism which reduces average fooling rate by threefold (to 15.22%) against single policy (SPA) and twofold (to 29.44%) against multiple policy attack (MPA) and thus improve the overall robustness of the android malware detection models.

[Home](#) > [Code](#) > [Text](#) > [ASCII table](#)

## ASCII Table

ASCII (American Standard Code for Information Interchange) character code chart with decimal,hex,binary,HTML and description:

- Collapse + Expand search character

0 00 NUL	23 17 ETB	46 2E :	69 45 E	92 5C \	115 7:
1 01 SOH	24 18 CAN	47 2F /	70 46 F	93 5D ]	116 74
2 02 STX	25 19 EM	48 30 0	71 47 G	94 5E ^	117 7!
3 03 ETX	26 1A SUB	49 31 1	72 48 H	95 5F _	118 70
4 04 EOT	27 1B ESC	50 32 2	73 49 I	96 60 :	119 7:
5 05 ENQ	28 1C FS	51 33 3	74 4A J	97 61 a	120 7:
6 06 ACK	29 1D GS	52 34 4	75 4B K	98 62 b	121 7:
7 07 BEL	30 1E RS	53 35 5	76 4C L	99 63 c	122 7:
8 08 BS	31 1F US	54 36 6	77 4D M	100 64 d	123 7:
9 09 HT	32 20 space	55 37 7	78 4E N	101 65 e	124 7:
10 0A LF	33 21 !	56 38 8	79 4F O	102 66 f	125 7:
11 0B VT	34 22 "	57 39 9	80 50 P	103 67 g	126 7:
12 0C FF	35 23 #	58 3A :	81 51 Q	104 68 h	127 7:
13 0D CR	36 24 \$	59 3B ;	82 52 R	105 69 i	
14 0E SO	37 25 %	60 3C <	83 53 S	106 6A j	
15 0F SI	38 26 &	61 3D =	84 54 T	107 6B k	
16 10 DLE	39 27 `	62 3E >	85 55 U	108 6C l	
17 11 DC1	40 28 (	63 3F ?	86 56 V	109 6D m	
18 12 DC2	41 29 )	64 40 @	87 57 W	110 6E n	
19 13 DC3	42 2A *	65 41 A	88 58 X	111 6F o	
20 14 DC4	43 2B +	66 42 B	89 59 Y	112 70 p	
21 15 NAK	44 2C ,	67 43 C	90 5A Z	113 71 q	
22 16 SYN	45 2D -	68 44 D	91 5B [	114 72 r	

[ASCII,Hex,Dec,Bin,Base64 converter ►](#)

## Extended ASCII table

Character encoding

UTF-8 (Unicode)

▼

128 80	151 97	174 AE ®	197 C5 Å	220 DC Ü	243 F:
129 81	152 98	175 AF ¯	198 C6 Æ	221 DD Ý	244 F:
130 82	153 99	176 B0 °	199 C7 Ç	222 DE þ	245 F:
131 83	154 9A	177 B1 ±	200 C8 È	223 DF ß	246 F:
132 84	155 9B	178 B2 ²	201 C9 É	224 E0 à	247 F:
133 85	156 9C	179 B3 ³	202 CA Ê	225 E1 á	248 F:
134 86	157 9D	180 B4 ́	203 CB Ë	226 E2 â	249 F:
135 87	158 9E	181 B5 µ	204 CC Ì	227 E3 ã	250 F:
136 88	159 9F	182 B6 ¶	205 CD Í	228 E4 ä	251 F:
137 89	160 A0	183 B7 ·	206 CE Î	229 E5 å	252 F:
138 8A	161 A1 i	184 B8 ¸	207 CF Ï	230 E6 æ	253 F:
139 8B	162 A2 ¢	185 B9 ¸	208 D0 Đ	231 E7 ç	254 F:
140 8C	163 A3 £	186 BA º	209 D1 Ñ	232 E8 è	255 F:
141 8D	164 A4 ☐	187 BB »	210 D2 Ò	233 E9 é	
142 8E	165 A5 ¥	188 BC ¼	211 D3 Ó	234 EA ê	
143 8F	166 A6 ¡	189 BD ½	212 D4 Õ	235 EB ë	
144 90	167 A7 §	190 BE ¾	213 D5 Õ	236 EC ï	
145 91	168 A8 ¨	191 BF ¸	214 D6 Ö	237 ED í	
146 92	169 A9 ©	192 C0 Á	215 D7 ×	238 EE î	
147 93	170 AA ª	193 C1 Á	216 D8 Ø	239 EF ï	
148 94	171 AB «	194 C2 Â	217 D9 Ù	240 F0 ð	
149 95	172 AC ¬	195 C3 Â	218 DA Ú	241 F1 ñ	
150 96	173 AD	196 C4 Ä	219 DB Û	242 F2 ö	

## What is ASCII code

ASCII (American Standard Code for Information Interchange) is a 7-bit characters code, with values from 0 to 127. The ASCII code is a subset of UTF-8 code. The ASCII code includes control characters and printable characters: digits, uppercase letters and lowercase letters.

## ASCII vs Unicode

ASCII is a 7-bit characters code, with values from 0 to 7F<sub>16</sub>. Unicode characters code is a superset of ASCII that contains the ASCII code with values from 0 to 10FFFF<sub>16</sub>

[Unicode character table ►](#)

## See also

[ASCII,Hex,Dec,Bin,Base64 converter](#)[ASCII to hex converter](#)[ASCII to binary converter](#)[Binary to ASCII converter](#)

[Hex to ASCII converter](#)[HTML char codes](#)[Unicode characters](#)[Windows ALT codes](#)[ASCII of 0](#)[ASCII of 'A'](#)[ASCII of enter](#)[ASCII of space](#)[ASCII code](#)[Hex,Dec,Bin converter with bit toggle](#)

## Write how to improve this page

Your message ...

Submit Feedback

### CODE TEXT

- [ALT codes](#)
- [ASCII table](#)
- [Text editor](#)
- [Unicode characters](#)

### RAPID TABLES

- [Recommend Site](#)
- [Send Feedback](#)
- [About](#)

[Home](#) | [Web](#) | [Math](#) | [Electricity](#) | [Calculators](#) | [Converters](#) | [Tools](#)

© RapidTables.com | [About](#) | [Terms of Use](#) | [Privacy Policy](#) | [Manage Cookies](#)