



THE UNIVERSITY OF BRITISH COLUMBIA

# Symmetric Crypto Systems

EECE 412

Copyright © 2004-2012 Konstantin Beznosov

# Module Outline

- Stream ciphers “under the hood”
- Block ciphers “under the hood”
- Modes of operation for block ciphers

# learning objectives

- explain main properties of block and stream ciphers,
- match a cipher type and mode of operation to the system at hand,
- explain how ECB, CBC, OFB, and CTR modes of operation work and draw diagrams showing that,
- given a mode of operation, identify its advantages and shortcomings.

# Stream Ciphers



# Random Generator (Stream Cipher)

as Random Oracle

- In:
  - short string (**key**)
  - length of the output



- Out: **long random** stream of bits (**keystream**)
- Applications:
  - Communications encryption
  - Storage encryption

## Properties

- Should not reuse
  - Use **seed**

# Stream Ciphers

- Not as popular today as block ciphers
- A5/I
  - Designed for hardware implementations
  - Based on shift registers
  - Used in GSM mobile phone system
- RC4
  - Designed for software implementations
  - Based on a changing lookup table
  - Used many places

# A5/I

- A5/I consists of 3 shift registers
  - X: 19 bits ( $x_0, x_1, x_2, \dots, x_{18}$ )
  - Y: 22 bits ( $y_0, y_1, y_2, \dots, y_{21}$ )
  - Z: 23 bits ( $z_0, z_1, z_2, \dots, z_{22}$ )

# A5/I

- At each step:  $m = \text{maj}(x_8, y_{10}, z_{10})$ 
  - Examples:  $\text{maj}(0,1,0) = 0$  and  $\text{maj}(1,1,0) = 1$
- If  $x_8 = m$  then X steps
  - $t = x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18}$
  - $x_i = x_{i-1}$  for  $i = 18, 17, \dots, 1$  and  $x_0 = t$
- If  $y_{10} = m$  then Y steps
  - $t = y_{20} \oplus y_{21}$
  - $y_i = y_{i-1}$  for  $i = 21, 20, \dots, 1$  and  $y_0 = t$
- If  $z_{10} = m$  then Z steps
  - $t = z_7 \oplus z_{20} \oplus z_{21} \oplus z_{22}$
  - $z_i = z_{i-1}$  for  $i = 22, 21, \dots, 1$  and  $z_0 = t$
- Keystream bit is  $x_{18} \oplus y_{21} \oplus z_{22}$

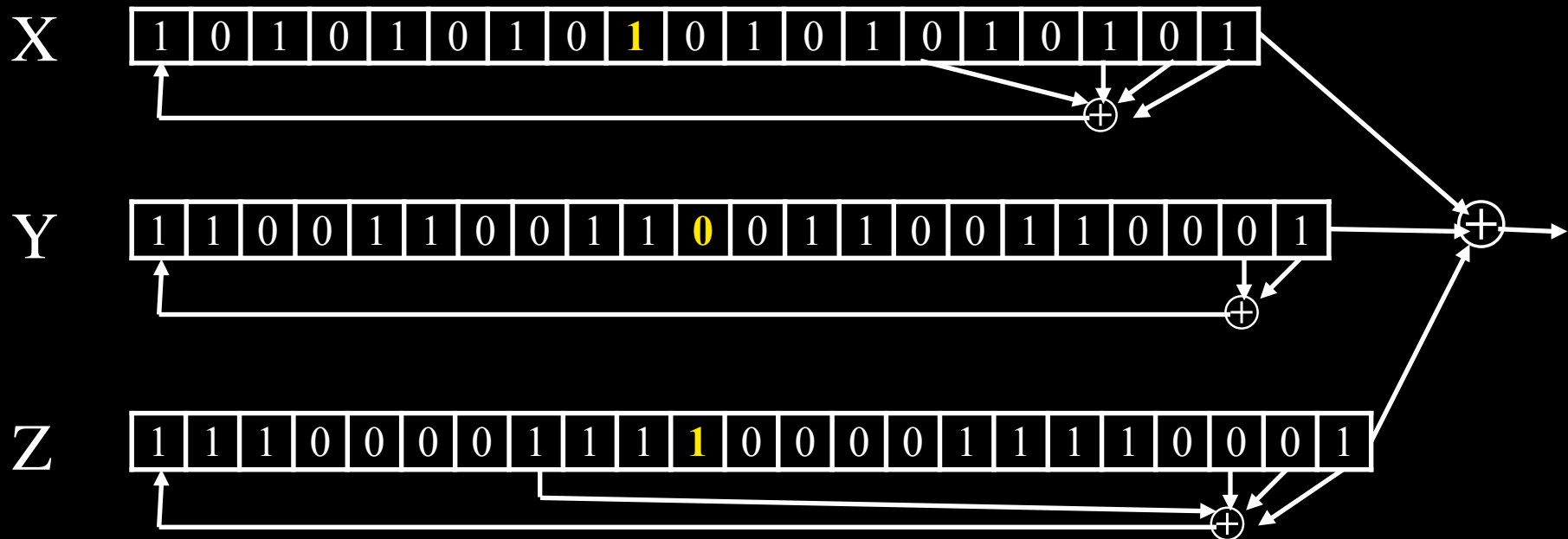


# A5/1



- Each value is a single bit
- Key is used as **initial fill** of registers
- Each register steps or not, based on  $(x_8, y_{10}, z_{10})$
- Keystream bit is XOR of right bits of registers

# A5/1: example



- In this example,  $m = \text{maj}(x_8, y_{10}, z_{10}) = \text{maj}(1, 0, 1) = 1$
- Register X steps, Y does not step, and Z steps
- Keystream bit is XOR of right bits of registers
- Here, keystream bit will be  $0 \oplus 1 \oplus 0 = 1$

# Use of Stream Ciphers

- Stream ciphers were big in the past
  - Efficient in hardware
  - Speed needed to keep up with voice, etc.
- Today, processors are fast, so software-based crypto is fast enough

# Block Ciphers “Under the Hood”

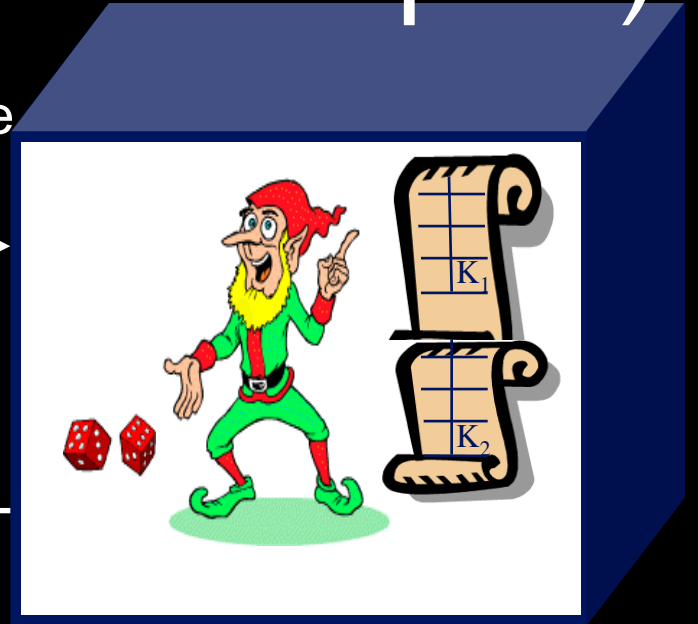
# Random Permutation (Block Cipher)

as Random Oracle

- In
  - fixed size short string (**plaintext**)  $M$ ,
  - DES -- 64 bits
  - **Key**  $K$

Queries →

← Responses



- Out
  - same fixed size short string (**ciphertext**)  $C$

## Notation

- $C = \{ M \}_K$
- $M = \{ C \}_K$

## Properties

- Invertible

# Related Notes

- Main properties of block ciphers
  - invertible
  - confusing
  - diffusing
- Main block ciphers
  - Data Encryption Standard (**DES**)
  - Advanced Encryption Standard (**AES**) a.k.a., Rijndael

# (Iterated) Block Cipher

- Plaintext and ciphertext consists of fixed sized blocks
- Ciphertext obtained from plaintext by iterating a **round function**
- Input to round function consists of key and the output of previous round
- Usually implemented in software

# Feistel Cipher

- **type** of block cipher design, not a specific cipher
- Split plaintext block into left and right halves:  
Plaintext =  $(L_0, R_0)$

- For each round  $i=1, 2, \dots, n$ , compute

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

where  $F$  is **round function** and  $K_i$  is **subkey**

- Ciphertext =  $(L_n, R_n)$



# Feistel Cipher

- Decryption: Ciphertext =  $(L_n, R_n)$
- For each round  $i = n, n-1, \dots, 1$ , compute

$$R_{i-1} = L_i$$

$$L_{i-1} = R_i \oplus F(R_{i-1}, K_i)$$

where  $F$  is round function and  $K_i$  is subkey

- Plaintext =  $(L_0, R_0)$
- Formula “works” for any function  $F$
- But only secure for certain functions  $F$ 
  - silly round function example:  $F(x, y) == 0$  for any  $x$  and  $y$ .

# Advanced Encryption Standard

- Replacement for DES
- AES competition (late 90's)
  - NSA openly involved
  - Transparent process
  - Many strong algorithms proposed
  - Rijndael Algorithm ultimately selected
    - Pronounced like “Rain Doll” or “Rhine Doll”
    - invented by Joan Daemen and Vincent Rijmen
- Iterated block cipher (like DES)

# AES Overview

- **Block size:** 128 bits (Rijndael had also 192 or 256)
- **Key length:** 128, 192 or 256 bits (independent of block size)
- 10 to 14 rounds (depends on key length)
- Each round uses 4 functions (in 3 “layers”)
  - ByteSub (nonlinear layer)
  - ShiftRow (linear mixing layer)
  - MixColumn (nonlinear layer)
  - AddRoundKey (key addition layer)



# AES demonstration

# review questions

- in A5/1, how is the keystream bit is used after its been obtained (after all it's only a single bit)?
- how would you define “confusion” and “diffusion” in the context of ciphers?
  - confusion -- obscuring the relationship between the plaintext and ciphertext
  - diffusion -- spreading the plaintext statistics through the ciphertext

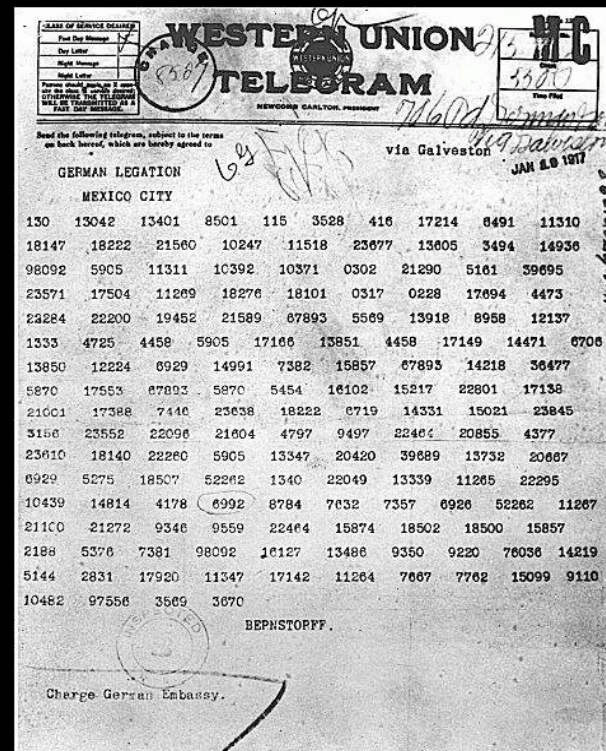


# Modes of Operation

# Code book

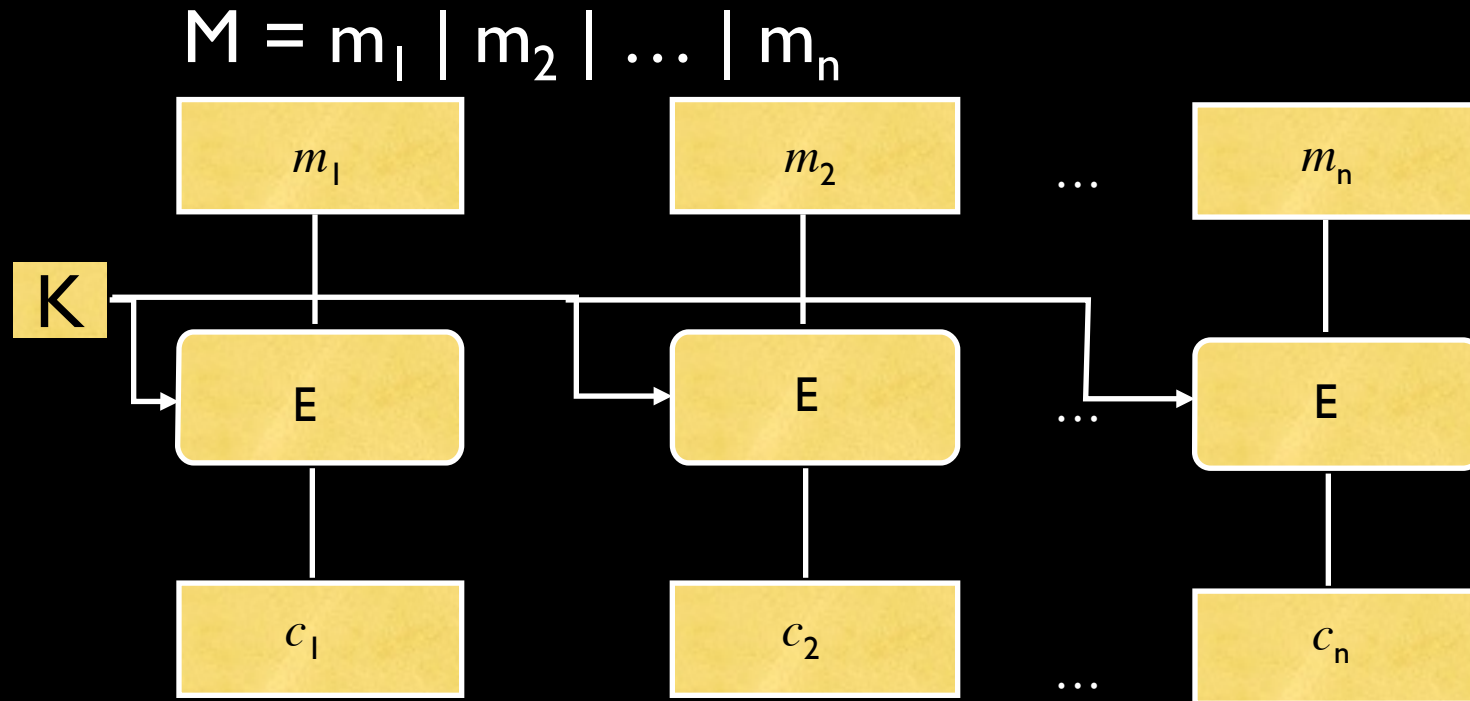
- Literally, a book filled with “codewords”

Februar	13605
fest	13732
finanzielle	13850
folgender	13918
Frieden	17142
Friedenschluss	17149
:	:



- Modern block ciphers are code books!

# Electronic Code Book (ECB)



$$c_i = E_K(m_i) \quad C = c_1 | c_2 | \dots | c_n$$

## Drawbacks

- Same message has same ciphertext
- Redundant/repetitive patterns will show through
- Subject to “cut-and-splice” attacks



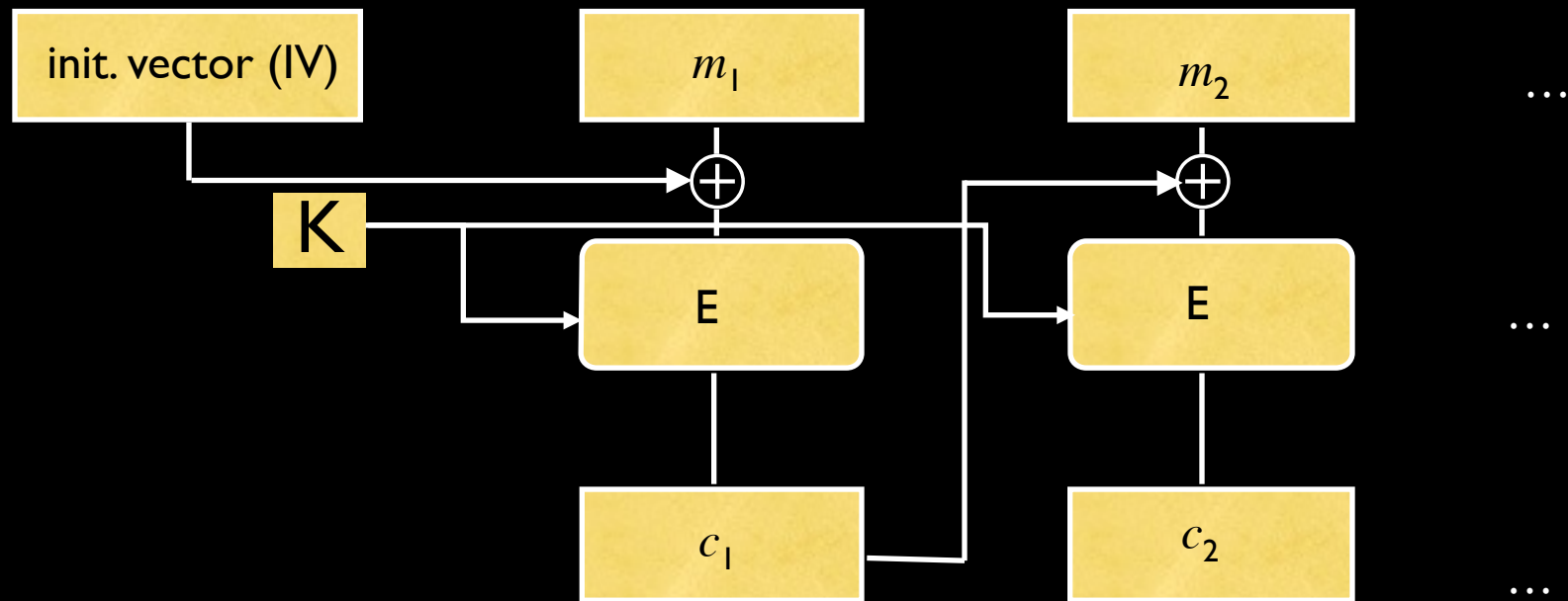
# Alice in ECB Mode



# Cipher Block Chaining (CBC)

$$c_i = E_K(m_i \oplus c_{i-1})$$

$$M = m_1 | m_2 | \dots | m_n$$



$$C = IV | c_1 | c_2 | \dots | c_n$$

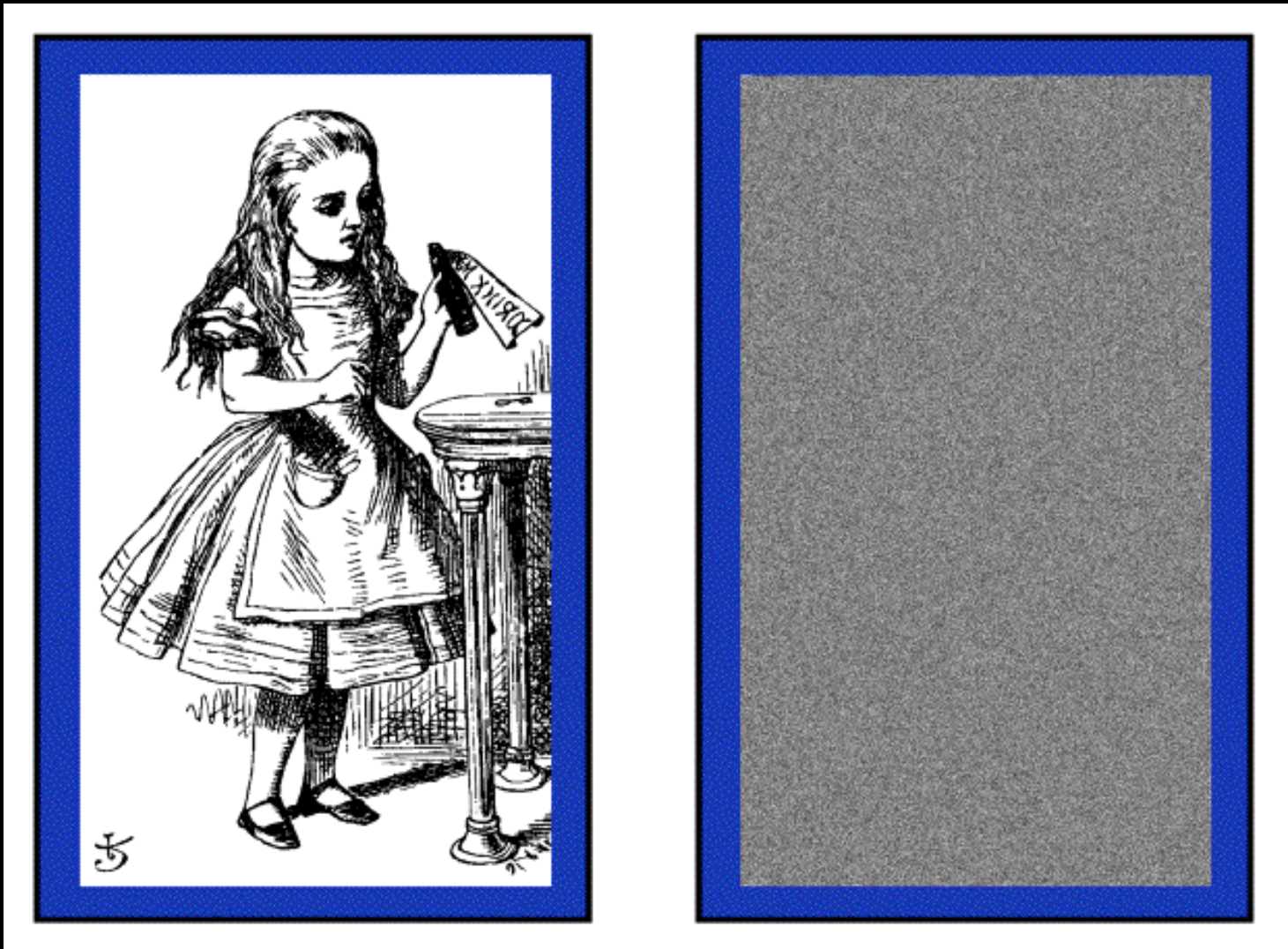
Decrypting with CBC:  $m_i = D_K(c_i) \oplus c_{i-1}$

Drawback: cannot precompute  $c_i$  without  $c_{i-1}$

# Alice in ECB Mode



# Alice in CBC Mode



# Output Feedback (OFB) Mode

- $K_0 = IV, K_1 = E_K(IV), K_2 = E_K(K_1), \dots, K_i = E_K(K_{i-1}) \dots$
- $C_i = m_i \oplus K_i$
- draw OFB diagram, similar to the one for CBC
- Purpose
  - use **block** cipher as a **stream** cipher
- Drawback
  - $K_1, \dots, K_i$  must be kept in memory

# TLS example

- CipherSuite TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA = { 0x00, 0x35 };
- CipherSuite TLS\_DH\_RSA\_WITH\_AES\_256\_CBC\_SHA = { 0x00, 0x37 };

# Counter Mode (CTR)

- Drawbacks of **feedback** modes
  - Hard to parallelize
    - CBC -- cannot pre-compute
    - OFB -- memory requirements
- Counter Encryption is easier to parallelize
  - $c_i = m_i \oplus E_K(IV+i)$
  - draw CTR diagram for decryption
    - $m_i = c_i \oplus E_K(IV+i)$

# IPSec example

Case #3: Encrypting 48 bytes (3 blocks) using AES-CBC with 128-bit key

Key : 0x6c3ea0477630ce21a2ce334aa746c2cd

IV : 0xc782dc4c098c66cbd9cd27d825682c81

Plaintext : "This is a 48-byte message (exactly 3 AES blocks)"

Ciphertext: 0xd0a02b3836451753d493665d33f0e886

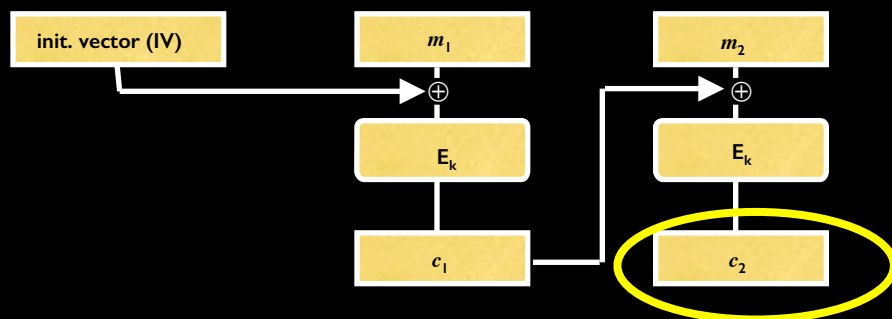
2dea54cdb293abc7506939276772f8d5

021c19216bad525c8579695d83ba2684



# message authentication code (MAC)

- Purpose
  - protect **message integrity** and **authenticity**
- How to do MAC with a block cipher?

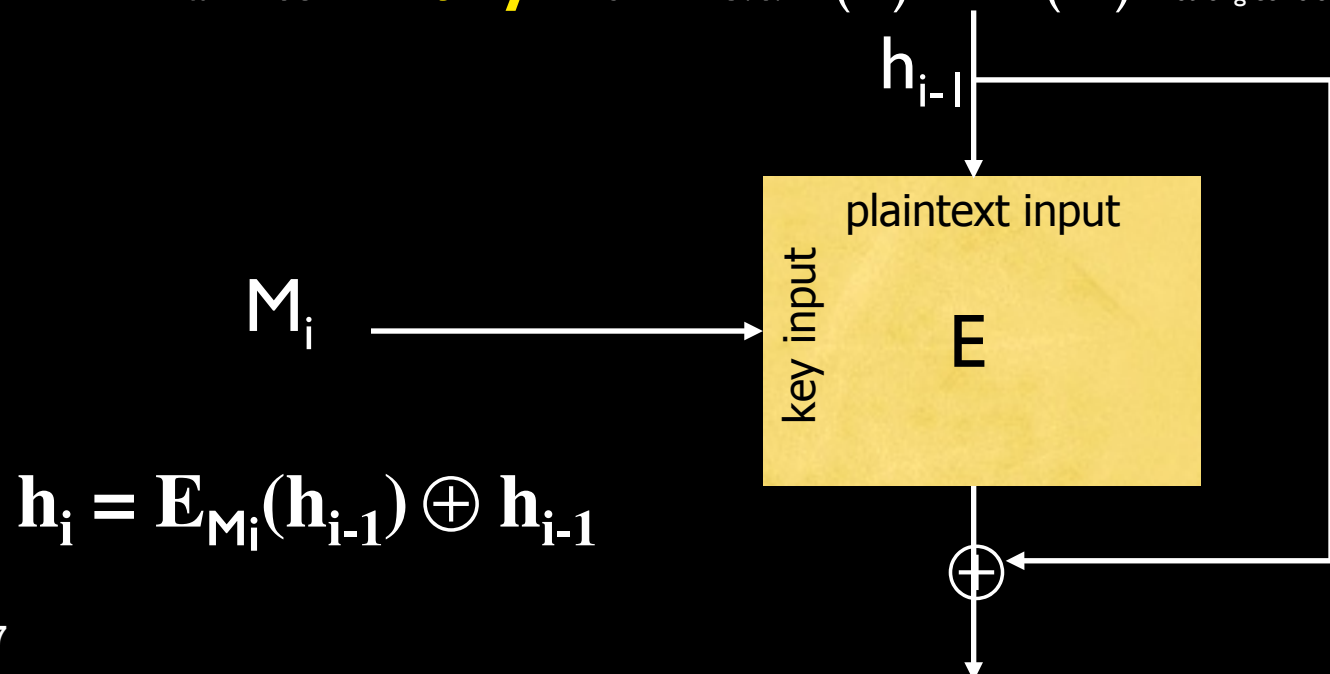


In CBC mode, the last block of cipher text serves as the MAC for the entire message

# Hash Function from a Block Cipher

$$h = H(M)$$

1. Easy to compute  $h$  from  $M$  - efficient
2. Hard to compute  $M$  from  $h$  - one way
3. For **given**  $M$ , hard to find another  $M'$  s.t.  $H(M) == H(M')$  - weak collision resistance
4. Hard to find **any**  $M$  &  $M'$  s.t.  $H(M) == H(M')$  - strong collision resistance



# Common Hash Functions and Applications

- Common hash functions
  - (Message Digest) MD5 value 128b
  - (Secure Hash Algorithm) SHA-1 180b value, SHA-256, SHA-512
- Applications
  - MACs
    - $MAC_K(M) = H(K, M)$
    - $HMAC_K(M) = H(K \oplus A, H(K \oplus B, M))$ ,  $A$  &  $B =$  magic (Section 5.7, Stamp)
  - Time stamping service
  - key updating
    - $K_i = H(K_{i-1})$
    - Backward security
  - Autokeying
    - $K_{i+1} = H(K_i, M_{i1}, M_{i2}, \dots)$
    - Forward security

# Key Points

- Ciphers are either **substitution**, transposition (a.k.a., **permutation**), or product
- Any block cipher should **confuse** and **defuse**
- Block ciphers are implemented in **SP-networks**
- Stream ciphers and hash functions are commonly implemented with block ciphers
- Hash functions used for
  - fingerprinting data, MAC, key updating, autokeying
  - Backward & forward security properties

# learning objectives

- explain main properties of block and stream ciphers,
- match a cipher type and mode of operation to the system at hand,
- explain how ECB, CBC, OFB, and CTR modes of operation work and draw diagrams showing that,
- given a mode of operation, identify its advantages and shortcomings,
- explain how MAC can be implemented and how it's different from just hash and from a cipher,
- explain backward and forward security and how they can be achieved.