Block Ciphers and Modes of Operation

Gerardo Pelosi

Department of Electronics, Information and Bioengineering (DEIB)
Politecnico di Milano

gerardo.pelosi - at - polimi.it

Overview

Lesson contents

- Design structures for block ciphers: Feistel Network (FN) and Substitution & Permutation Network (SPN)
- The Feistel Network structure
- Data Encryption Standard (DES)
- Security of DES, Double-DES, Triple-DES, DES-X
- Modes of Operation to guarantee Confidentiality: ECB, CBC, OFB, CFB, CTR

Design of Symmetric-key Ciphers: Confusion & Diffusion

Building on the weaknesses of Historical ciphers, C. Shannon stated the following (very general and informal) design principles to thwart cryptanalysis based on statistical properties of ptxs and ctxs

A symmetric cipher should be composed as the iterative application of operations that realize *Confusion & Diffusion* of the plaintext symbols

Confusion

make the relation between the key, plaintext and ciphertext as complex as possible. Ideally, **each digit of the key influences the correspondence between pxt and cxt letters** in a non-predictable way

- Replacing every letter with the one next to it on the typewriter keyboard is an insecure example of confusion by *substitution*
- Ciphers that do not offer effective confusion are vulnerable to frequency analysis

Design of Symmetric-key Ciphers: Confusion & Diffusion

Diffusion

refers to the property that the **statistical distribution of "groups of pxt letters" frequencies** (due to the redundancy of the ptx language) **should be dissipated**, as much as possible, **into flat distribution statistics**, i.e. the ctx should appear as random data.
Ideally, keeping the same key, the change of a single bit in the plaintext drives the change of all bits in ciphertext

- In contrast to confusion, **diffusion spreads (diffuse) the influence of a single plaintext letter over many (or every) cxt letters**
- Ciphers suffering from poor diffusion can usually be broken by means of Known Plaintext Attacks (e.g., simple permutation ciphers)

Modern Block Ciphers

Block ciphers operate on a block of plaintext $m \in \mathcal{M}$ ($m = \langle m_1, \ldots, m_n \rangle$, with $m_i \in \{0,1\}$), to produce a block of ciphertext ($c = \langle c_1, \ldots, c_n \rangle \in \mathcal{C}$, with $c_i \in \{0,1\}$) through a key-parametric transformation (either $\mathbb{E}_k()$ or $\mathbb{D}_k()$)

- The block size n is in the [64, 256] bit range
- ptx size might not be multiple a of block size $\rightarrow$ padding needed! Possible padding strategies:
  - Known *non-data* values (e.g. nulls)
  - A number indicating the size of the pad (may require an extra block)
  - A number indicating the size of the ptx (may require an extra block)
- For ptxs longer than a single block, the scheme used to apply $\mathbb{E}_k()$ (or $\mathbb{D}_k()$) is called mode of operation

Purposes of a block cipher

Security desiderata

- Provide nontrivial blending of plaintext and ciphertext (confusion)
- Provide statistical flatness of the output (diffusion)
- Be analyzable in a clear and formal way to ensure sound security
  - A structure based on the repetition of a short sequence of steps (round) is preferable to a monolithic design

Efficiency desiderata

- Provide efficient encryption and decryption
- Possibly reuse SW/HW resources for both $\mathbb{E}_k()$ and $\mathbb{D}_k()$ (lower code size or silicon area)

Modern Design Strategies

Glossary

- Cipher **state**: The result of each operation performed by the cipher
  - Initialized with the ptx; contains the ctx at the end of the computation
- **Round**: basic sequence of operations applied to the cipher state, a number of times (involves blending the key into the state)
- **Key schedule**: procedure expanding the original user key into key material to be used in each round

High level structure

- **Expand** the user key into a set of subkeys and combine them with the cipher state during the execution of the round primitive
- **Iterate** the application of the *Round*.
  Repeating the round increases the complexity of the dependency relations among the user key bits and the bits of the cipher state

Modern Design Strategies: Feistel Networks and SPNs

Feistel Networks

- Invented by Horst Feistel (in '50-'60), it splits the cipher state in two parts and acts on one of them per round
- The decryption employs the same cipher structure, except for a reversal in the key schedule

Substitution Permutation Networks (SPNs)

- A SPN implements the confusion-diffusion principles suggested by Shannon with distinct enc/dec transformations
- The round of a SPN acts on the whole cipher state with:
  - A "non-linear" function providing Confusion represented as a lookup table, a.k.a.: Substitution-Box (S-box)
  - A "linear" function providing Diffusion, f.i., a bitwise permutation, or pairs of `rotate` & `xor` operations
  - The addition of a part of the key schedule

Feistel Network

Definition

A Feistel network transforms an n-bit ptx block $m{=}\langle L_0, R_0\rangle$, into an n-bit ctx block $c{=}\langle R_r, L_r\rangle$ through an r-round process ($r{\geq}1$); where the sub-blocks L_i, R_i are $n/2$-bit long.

FEISTEL($\langle L, R\rangle, k$)

1 **for** $i \leftarrow 0$ **to** $r-1$
2 $\quad temp \leftarrow L$
3 $\quad L \leftarrow R$
4 $\quad R \leftarrow temp \oplus \mathcal{F}(k_i, R)$ // $L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus \mathcal{F}(k_i, R_{i-1})$
5 $R \leftarrow R \oplus \mathcal{F}(k_r, L)$
6 **return** $\langle R, L\rangle$ // Note: the last round block halves are swapped

where $\mathcal{F}$ is an arbitrary function (non-linear and possibly non-invertible) and each subkey k_i, $0{\leq}i{\leq}r$, is computed from key schedule of the cipher key k

Structural Properties

Properties of a Feistel network:

- the round transformation is invertible **regardless** of the choice of the function $\mathcal{F}$. Indeed, for the i-th round:

$$L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus \mathcal{F}(k_i, R_{i-1})$$

  we can also write

$$R_{i-1} = L_i, \quad L_{i-1} = R_i \oplus \mathcal{F}(k_i, L_i)$$

- Note that the last round block halves are swapped $\langle R_r, L_r \rangle$, and not in their usual left-right order $\langle L, R \rangle$
- Applying the Feistel network on a ctx (using the subkeys in reverse order, i.e. k_r through k_0) provides the ptx

Feistel Network

Block Diagram

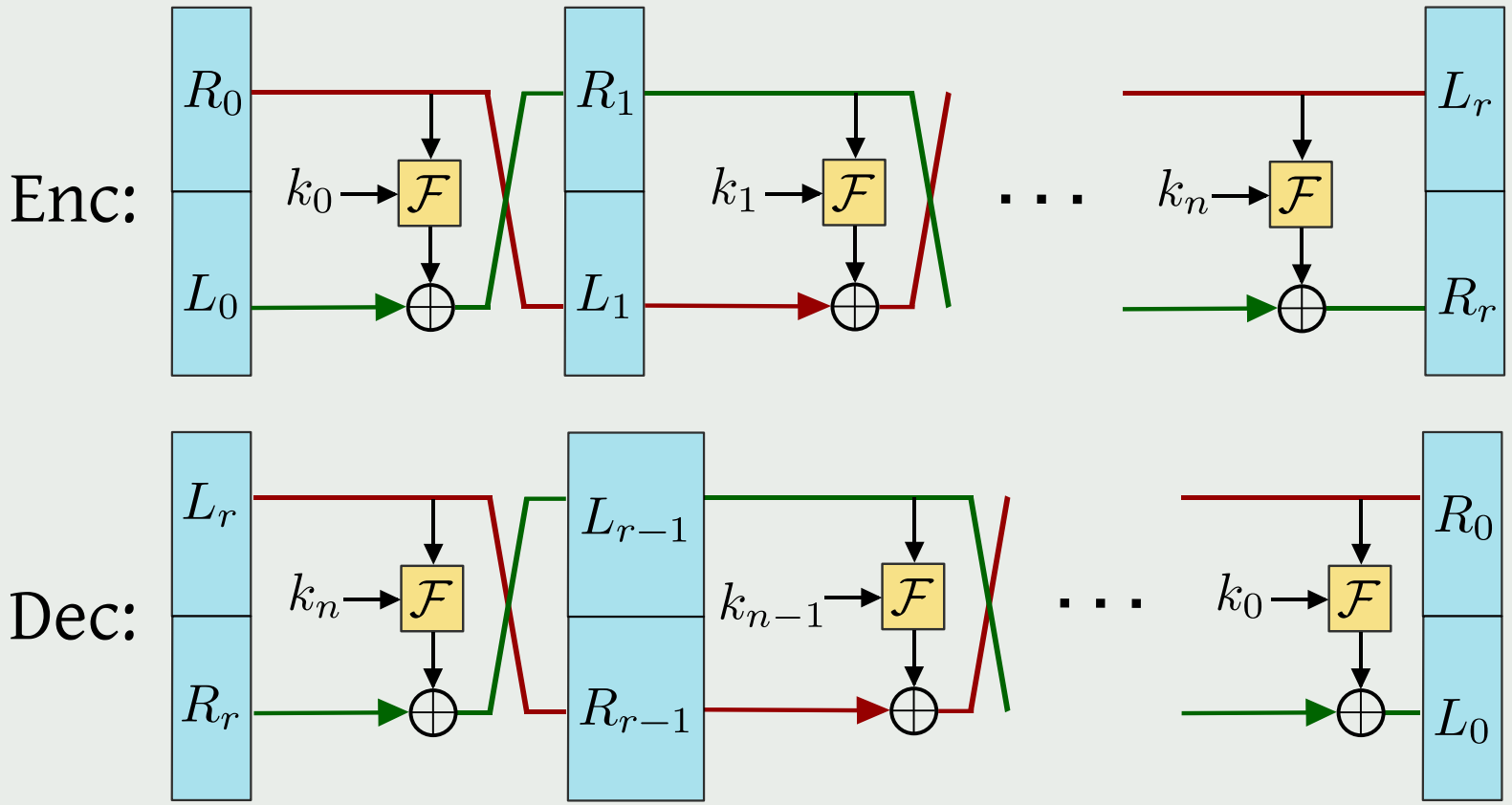


Figure: The structure of a Feistel Network

Feistel Network

Description

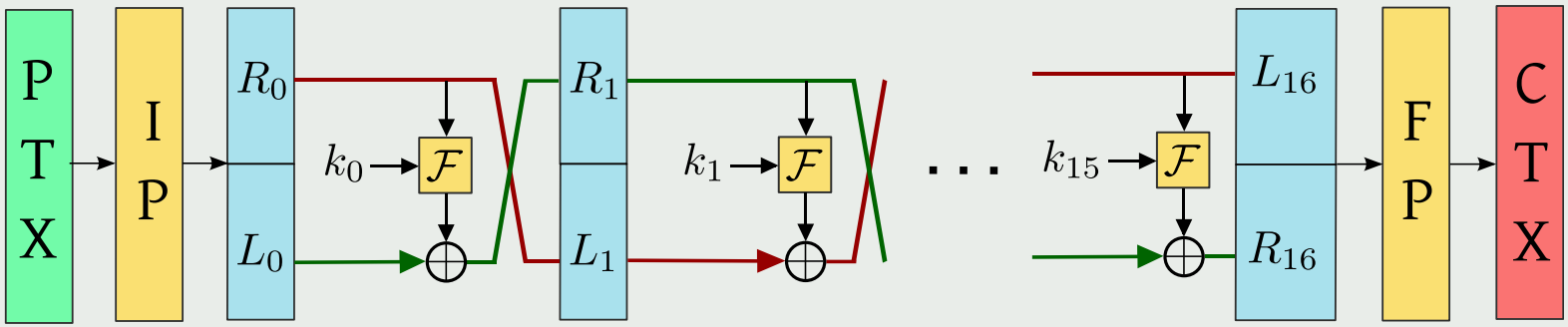
- Confusion: provided by the key-dependent $\mathcal{F}$ function
- Diffusion: obtained adding the $\mathcal{F}$-processed part R_i to L_i

Ciphers based on a Feistel Network

- **DES**: block-size: 64-bit, key-size:56-bit, rounds: 16.
  Standard encryption algorithm employed by the US-government for unclassified documents from 1976 to 2000
- **Blowfish**: block-size: 64-bit, key-size: 32–448 bits (4–56 bytes), rounds: 16.
  Part of the key schedule used to generate S-Boxes in the $\mathcal{F}$, very demanding key-schedule
- **Twofish**: evolution of blowfish, block-size: 128-bit, key-size: 128, 192 o 256 bits, rounds: 16
- **CAST5**: block-size: 128-bit, key-size: 40–128 bits (5–16 bytes), rounds: 12 (16, with key-size $\geq$ 80 bits). Only arithmetic operations and key-dependent bit-rotations employed in the $\mathcal{F}$ map. Efficient SW implementation

Feistel Network

DES - structure



16-round Feistel cryptosystem with 64-bit wide cipher state:

- Cipher key: 64 bits; only 56 bits are used. One bit per byte is a parity bit.
- Key schedule: produces $\langle k_1, \ldots, k_{16} \rangle$, 48 bits each. Each round key is obtained through bitwise permutation and selection of the initial 56 bits. DES-peculiar feature: $k_0 = k_{15}$
- The fixed permutations IP, FP=IP^{-1}, have no effect on the security; IP and FP were motivated by the ease of laying out the circuit wires

DES Structure

DES Round

It takes a 64-bit block $\langle L_{i-1}, R_{i-1} \rangle$ and a 48-bit subkey k_i to compute $L_i = R_{i-1}, R_i = L_{i-1} \oplus \mathcal{F}(k_i, R_{i-1})$, where

$$\mathcal{F}(k_i, R_{i-1}) = \texttt{P-box}(\texttt{S-box}(k_i \oplus \texttt{E-box}(R_{i-1}))$$

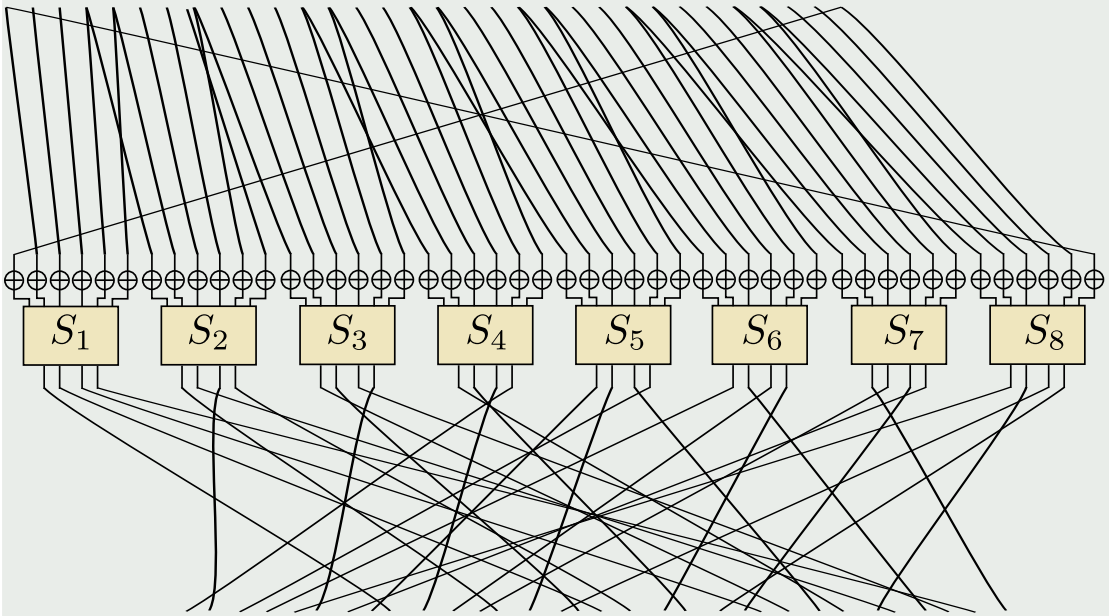
1. $\texttt{E-box}(R_{i-1})$ expands R_{i-1} from 32 into 48 bits ($\langle b_1 b_2 \dots b_8 \rangle$, each b_i is 6 bits) via a *fixed expansion* that simply duplicates some bits
2. Adds the 48-bit round key: $\langle b_1 b_2 \dots b_8 \rangle \leftarrow k_i \oplus \langle b_1 b_2 \dots b_8 \rangle$
3. Map the 48-bit word into a 32-bit one via applying 8 fixed S-boxes.
   - For each b_i, $c_i = S_i(b_i)$, where S_i maps 6 into 4 bits
   - S_i are the only non-linear component of DES.
   - S_i described as a 4×16 look-up tables: 1st and 2nd input bit $\rightarrow$ row index, 3rd to 6th bit $\rightarrow$ column index, each cell contains 4 output bits
4. Apply a fixed bitwise permutation specified by the P-box

DES Round

$\mathcal{F}$-function

$$\mathbf{L_i} = \mathbf{R_{i-1}}, \mathbf{R_i} = \mathbf{L_{i-1}} \oplus \mathcal{F}(\mathbf{k_i}, \mathbf{R_{i-1}}) \quad //\text{half} - \text{word } (L_i, R_i) \text{ size :32-bit}$$

$$\mathcal{F}(\mathbf{k_i}, \mathbf{R_{i-1}}) = \texttt{P-box}(\texttt{S-box}(\mathbf{k_i} \oplus \texttt{E-box}(\mathbf{R_{i-1}})) \ //\text{output size :32-bit}$$



- E-box stage
  $4 \times 8 = \mathbf{32\ bits} \rightarrow$
  $6 \times 8 = 48$ bits
- bitwise XOR with the $6 \times 8 = 48$-bit round subkey
- S-box stage
  $6 \times 8 = 48$ bits $\rightarrow$
  $4 \times 8 = 32$ bits
- P-box stage
  32 bits $\rightarrow$ **32 bits**

DES Properties

Complementation property

Inverting the bit values of the input ptx m and key k, yields a ctx equal to bitwise inversion of the result of DES(k, m)

$$\mathrm{DES}(k, m) = \overline{\mathrm{DES}(\overline{k}, \overline{m})}, \quad \forall\, m \in \mathcal{M},\ \forall\, k \in \mathcal{K}$$

Chosen-ptx attack

Collect ptx-ctx pairs (m_1, c_1), $(\overline{m_1}, c_2)$, with $c_1 = \mathrm{DES}(k, m_1)$, $c_2 = \mathrm{DES}(k, \overline{m_1})$
Noting that

$$c_2 = \mathrm{DES}(k, \overline{m_1}) \Leftrightarrow \overline{c_2} = \mathrm{DES}(\overline{k}, m_1)$$

Test for any $\widetilde{k}$, if either DES$(\widetilde{k}, m_1)$ yields c_1 or $\overline{c_2}$.
If not, discard both $\widetilde{k}$ and $\neg\widetilde{k}$), with a single DES computation.
Exhaustive key search down to 2^{56-1} trials (...not that useful in practice!)

DES Properties

The DES secret key should be randomly chosen, but there are some particular values that should not be used as the key schedule creates 16 identical subkeys (**Weak Keys**) or only two different values for 16 subkeys (**Semi-weak keys**)

Weak Keys

Encrypting a ptx twice with a weak key k, yields the original ptx.

$$\mathrm{DES}(k, \mathrm{DES}(k, m)) = m, \ \forall \ m \in \mathcal{M}$$

There are 4 weak keys ($\texttt{Group}_1$) written with 64-bit, including the 8 rightmost parity bits

If the implementation does not check the parity bits of k,
4 extra key values are weak ($\texttt{Group}_2$)

$\texttt{Group}_1$	$\texttt{Group}_2$
0x 0101010101010101	0x 0000000000000000
0x FEFEFEFEFEFEFEFE	0x FFFFFFFFFFFFFFFF
0x E0E0E0E0F1F1F1F1	0x E1E1E1E1F0F0F0F0
0x 1F1F1F1F0E0E0E0E	0x 1E1E1E1E0F0F0F0F

DES Properties

Semi-Weak Keys

Also a Semi-Weak key pair $\langle k, k' \rangle$ causes the composition of two DES encryptions employing k and k' to compute the original ptx

$$\mathrm{DES}(k, \mathrm{DES}(k', m)) = m, \ \forall \ m \in \mathcal{M}$$

There are 6 semi-weak key pairs:

$$\langle \texttt{0x 011F011F010E010E}, \texttt{0x 1F011F010E010E01} \rangle$$
$$\langle \texttt{0x 01E001E001F101F1}, \texttt{0x E001E001F101F101} \rangle$$
$$\langle \texttt{0x 01FE01FE01FE01FE}, \texttt{0x FE01FE01FE01FE01} \rangle$$
$$\langle \texttt{0x 1FE01FE00EF10EF1}, \texttt{0x E01FE01FF10EF10E} \rangle$$
$$\langle \texttt{0x 1FFE1FFE0EFE0EFE}, \texttt{0x FE1FFE1FFE0EFE0E} \rangle$$
$$\langle \texttt{0x E0FEE0FEF1FEF1FE}, \texttt{0x FEE0FEE0FEF1FEF1} \rangle$$

DES Properties

DES is not a group

Given a ptx m and a pair of keys k_1, k_2, the set of DES (biijective) transformations is not closed under composition:

$$\forall m \ \nexists k_3 \text{ s.t. } \mathrm{DES}(k_3, m) = \mathrm{DES}(k_2, \mathrm{DES}(k_1, m))$$

The above condition means that encrypting a ptx m through applying DES twice with two different keys, is not the same as encrypting m once with a third key

- K.W. Campbell and M.J. Wiener, DES is not a group, Advances in Cryptology - Crypto'92, Springer-Verlag (1993), pp. 512-520

N.B.: Given the ptx space $\{0,1\}^{64}$, which has $n{=}2^{64}$ elements, a DES with a fixed key is one out of $(2^{64})!$ permutations $\mathcal{S}_n{:}\{0,1\}^{64}{\mapsto}\{0,1\}^{64}$.
The composition of two DES with distinct keys is still a permutation $\mathcal{S}_n$, but in general it **cannot be thought as computed by a DES with another key**

DES Breaking

Due to the computational power available today, that employs also specialized parallel hardware, DES has become insecure

Brute Forcing DES

Given a ptx-ctx pair, an *exhaustive key search* needs to perform on average 2^{55} DES encryptions:

- Computing 1 encryption per micro-second
  $\Rightarrow 2^{55} \approx 3.6\,10^{10}$ [DES/s] ≈ 1142 [DES/year]
- Employing 10^6 devices in parallel gives a break in ≈ 10 hours

DES Breaking

Brute Forcing DES

- 1998: Electronic Frontier Foundation (EFF) broke the DES employing ad-hoc ASIC hw (EFF DES Cracker: [92 10^9 [keys/sec]) in 56 hours for less than 250K USD
- 1999: EFF provided a DES breaking in 23 hours (245 10^9 [key/s]) employing a distributed breaker with 10^5 computers
- 2008: a German company, *SciEngines GmbH*, reduced the time to break DES in about **20 hours** (292 10^9 [key/s]), using a cluster of 16 devices, each equipped with 8 FPGAs Spartan-3 5000's, for a total cost of about than **5K Euros**, with a power consumption $\leq$ **850 watts** (peak)

To increase the resistance of the standard cipher against brute force attacks, in 1998 the **Triple DES** was standardized by the US institutions: ANSI and NIST

Double DES (2DES)

DES is not a group $\rightarrow$ multiple applications of the primitive with different keys seems to be a good strategy to improve the security

2DES

Double DES cipher consists of applying the DES primitive twice

$$c = 2\mathrm{DES}(k_1, k_2, m) \stackrel{\text{def.}}{=} \mathrm{DES}(k_1, \mathrm{DES}(k_2, m))$$

nevertheless, this structure is vulnerable to *meet-in-the-middle attacks*:

$$c = \mathrm{DES}(k_1, \mathrm{DES}(k_2, m)) \Longleftrightarrow \mathrm{DES}^{-1}(k_1, c) = \mathrm{DES}(k_2, m)$$

Double DES (2DES)

Meet-in-the-middle

This analysis technique trades the number of DES encryptions employed in an exhaustive key search with storage. Given a ptx-ctx pair, $\langle m, c \rangle$

$$c = \mathrm{DES}(k_1, \mathrm{DES}(k_2, m)) \Leftrightarrow \mathrm{DES}^{-1}(k_1, c) = \mathrm{DES}(k_2, m)$$

- for all k_2 candidate key values $k_{(2,i)} \in \{0, \ldots, 2^{56}-1\}$ compute $A_i = \mathrm{DES}(k_{(2,i)}, m)$ and store A_i
  - this costs 2^{56} DES encryptions and 2^{56} memory cells (with 64-bit size)
- for every k_1 candidate key values $k_{(1,j)} \in \{0, \ldots, 2^{56}-1\}$ compute $B_j = \mathrm{DES}^{-1}(k_{(1,j)}, c)$ and check all the equalities $A_i = B_j$.
  If an equality $A_i = B_j$ holds, then store the key pair $\langle k_{(2,i)}, k_{(1,j)} \rangle$
  - in the worst case, this costs 2^{56} decryptions

Double DES (2DES)

Key pair sieving

with the meet-in-the-middle analysis, there is a set of candidate key pairs $S=\{\langle k_{(2,i)}, k_{(1,j)}\rangle, \ldots, \}$

- Obs: given m, $c=\mathrm{DES}(k_{(1,j)}, \mathrm{DES}(k_{(2,i)}(m))$, there are 2^{112} possible key values and 2^{64} possible ctxs values c, therefore there are up to $\frac{2^{112}}{2^{64}}=2^{48}$ key values $\langle k_{(1,j)}, k_{(2,i)}\rangle$ associated to the pair $\langle m, c\rangle$

Through employing a second ptx-ctx pair $\langle m', c'\rangle$, it is possible to check which key pair in S is the correct one

- Obs. c' may have a value in the range $\{0, \ldots, 2^{64}-1\}$, but the set S limits the number of such values to 2^{48} (one for each possible key pair in S). Thus, the probability to have more than one key pairs in S that yield to the same c' is given by: $\text{possible mappings/tot. mappings}=\frac{2^{48}}{2^{64}}=\frac{1}{2^{16}}$.

The 2DES cipher employs a 112-bit key but it is vulnerable to a *Known ptx attack* with a cost of $\approx 2^{57}$ DES enc.s, thus, it is not better than the single DES !!!

Triple DES (TDES)

The standardized TDES provides a simple method to effectively increase the key size w.r.t. DES without designing a completely new block cipher

3DES

The improved primitive employs:

- a "key bundle" with three DES keys: k_1, k_2 and k_3, each of 56 bits (excluding parity bits)
- an iterated application of the DES primitive to define both the encryption and the decryption transformation

$$c = 3\mathrm{DES}(k_1, k_2, k_3, m) \stackrel{\text{def.}}{=} \mathrm{DES}(k_1, \mathrm{DES}^{-1}(k_2, \mathrm{DES}(k_3, m)))$$

$$m = 3\mathrm{DES}^{-1}(k_1, k_2, k_3, m) \stackrel{\text{def.}}{=} \mathrm{DES}^{-1}(k_3, \mathrm{DES}(k_2, \mathrm{DES}^{-1}(k_1, c)))$$

where DES^{-1} indicates the inverse DES transformation (i.e., the application of the key schedule in reverse order)

Triple DES (TDES)

$$c = \mathrm{DES}(k_1, \mathrm{DES}^{-1}(k_2, \mathrm{DES}(k_3, m)))$$

The standards define three keying options:

- Keying opt. 1 (3DES3): All three keys are independent
- Keying opt. 2 (3DES2): k_1 and k_2 are independent, and $k_3{=}k_1$
- Keying opt. 3 (no longer recommended): All three keys are identical, i.e., $k_1{=}k_2{=}k_3$
  - it used to provide backward compatibility with DES, because the 1st and 2nd applications of DES cancel out

Triple DES (TDES)

3DES3

$$c = \mathrm{DES}(k_1, \mathrm{DES}^{-1}(k_2, \mathrm{DES}(k_3, m)))$$

The 3DES with $k_1 \neq k_2 \neq k_3$ can also be attacked with the *meet-in-the middle* strategy considering a single $\langle m, c \rangle$ pair:
$\mathrm{DES}^{-1}(\mathbf{k_1}, \mathbf{c}) = \mathrm{DES}^{-1}(\mathbf{k_2}, \mathrm{DES}(\mathbf{k_3}, \mathbf{m}))$

- storage for 2^{112} values: $\{\langle\ \langle k_2, k_3 \rangle, \mathrm{DES}^{-1}(k_2, \mathrm{DES}(k_3, m))\ \rangle\}$
- for each value of k_1, 2^{56} DES^{-1} computations and lookups in the previous table to find a match

$\mathrm{DES}(k_2, \mathrm{DES}^{-1}(k_1, c)) = \mathrm{DES}(k_3, m)$

- storage for 2^{56} values: $\{\langle k_3, \mathrm{DES}(k_3, m)) \rangle\}$
- 2^{112} $\mathrm{DES}(k_2, \mathrm{DES}^{-1}(k_1, c))$ comp.s and lookups to find a match

The 3DES3 cipher employs a 168-bit key and requires $\approx 2^{112}$ DES encryptions for a *known ptx attack*

Triple DES (TDES)

3DES2

$$c = \text{3DES2}(k_1, k_2, k_1, m) \stackrel{\text{def.}}{=} \text{DES}(k_1, \text{DES}^{-1}(k_2, \text{DES}(k_1, m)))$$

The 3DES2 with $k_1 \neq k_2$ can also be attacked with a *meet-in-the middle* strategy having the same costs of attacking the 3DES3

The 3DES2 cipher is the preferred choice as it employs a 112-bit key and requires 2^{112} DES encryptions for a *known ptx attack*

Triple DES (TDES)

- 2DES is 2× slower than DES, 2× key length and the same security margin ($\approx 2^{56}$ DES enc.s)
- 3DES3 is 3× slower than DES, 3× key length and a double security margin ($\approx 2^{112}$ DES enc.s)
- 3DES2 is 3× slower than DES, 2× key length and a double security margin ($\approx 2^{112}$ DES enc.s)

DES-X

It is a variant on the DES block cipher to increase the complexity of a known ptx attack using a technique called *pre-whitening*. Three independent keys are used. The fist key k has 56-bit size, while the other keys k_1, k_2 have 64-bit size

$$c = \text{DES-X}(k, k_1, k_2, m) \stackrel{\text{def.}}{=} k_2 \oplus \text{DES}(k, m \oplus k_1)$$

$$m = \text{DES-X}^{-1}(k, k_1, k_2, c) \stackrel{\text{def.}}{=} k_1 \oplus \text{DES}^{-1}(k, c \oplus k_2)$$

DES-X is ≈ as fast as DES, with a 184-bit key and a security margin $\approx 2^{120}$ DES

Theoretical Attacks

Better techniques for the cryptanalysis of the DES are:

- linear cryptanalysis
- differential cryptanalysis

(Quantitative examples in next lectures)
These techniques are currently applied to test the robustness of every block cipher

Theoretical Attacks

Linear cryptanalysis

Finds approximated linear relations (i.e., Boolean equalities) among some bits of the ptx and some bits in input to the last round of the cipher. It recovers (in a subsequent step) the values of some bits of the secret key.

- 2^{43} known ptx-ctx pairs to find a DES key [Matsui, 1993]

Differential cryptanalysis

Finds how differences between two input ptxs propagate within the cipher up to the beginning of the last round. In a subsequent step, this knowledge is exploited to obtain the values of some bits in the last subkey.

- 2^{47} chosen ptx-ctx pairs to find a DES key, [Biham and Shamir, 1990]

Modes of Operation

Definition

A Mode of operation specifies the way to encrypt a message $m \in \mathcal{M}$ of **arbitrary length** through employing a block cipher

Currently, there are modes of operations to guarantee

- Confidentiality of messages:
  - Electronic CodeBook mode (ECB)
  - Cipher Block Chaining mode (CBC)–most popular–not entirely secure
  - Output FeedBack mode (OFB)
  - Cipher FeedBack mode (CFB)
  - Counter mode (CTR)
- Authentication of messages: CMAC
- Both confidentiality and authentication: CCM and GCM

We will present the last two categories after the study of Hash functions!

Modes of Operation

The plaintext message $m \in \mathcal{M}$ is broken into blocks of equal size
$m = \langle m_1, m_2, \ldots \rangle$
The last part of m might be shorter than one block, thus padding is needed
Possible padding:

- Known non-data values (e.g. nulls)
- A number indicating the size of the pad
- A number indicating the size of the plaintext

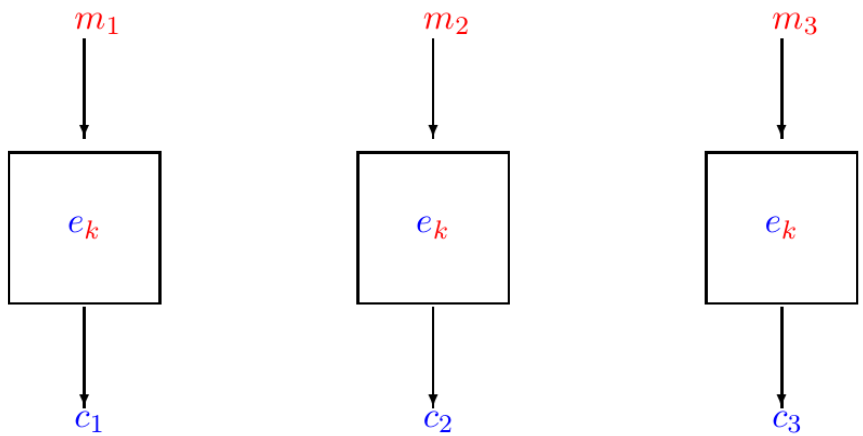
Note that the last two schemes may require an extra block

Electronic Code Book (ECB)

Each block is encrypted independently to compose the ciphertext message $c \in \mathcal{C}$ as $c = \langle\, c_1,\, c_2,\, \ldots \rangle$, where:

$$c_i = \mathbb{E}_k(m_i),\ \ i = 1, 2, \ldots$$

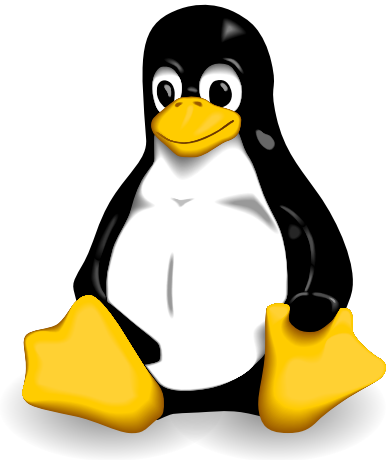
For a given key, this mode behaves like we have a gigantic codebook, in which each ptx block has an entry, hence the name *Electronic Code Book*



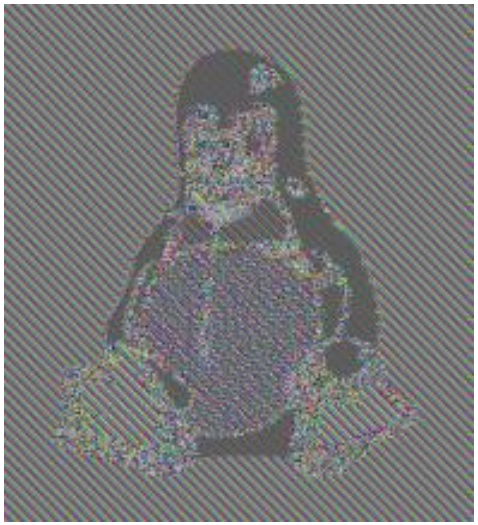
Electronic Code Book (ECB)

The major disadvantage of this method is that identical ptx blocks are encrypted into identical ctx blocks; thus, it does not hide data patterns well!

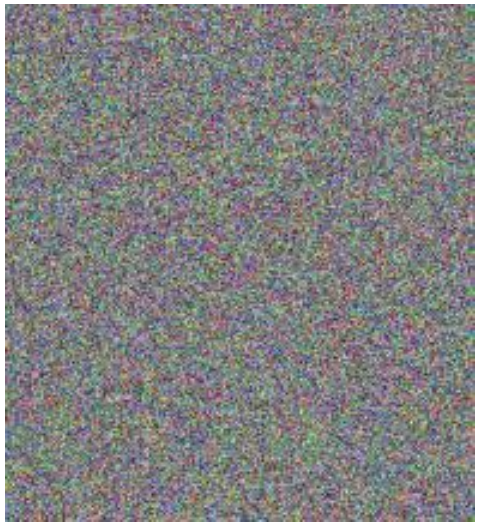
Original



ECB mode



Other modes



Remarks on ECB

Strengths:

- It's simple, fast and amenable to massive parallelization
- One-bit errors in the ctx cause a single block error in the ptx

Weaknesses:

- If the same message is encrypted (under the same key) twice, the ctxs are the same
- Repetitive information contained in the ptx may show in the ctx (expecially if aligned with blocks!)
- Suffers from block insertion or deletion attacks.
  - Can be compensated with a checksums over a number of ptx blocks

Typical application: secure transmission of short pieces of information (e.g., a temporary encryption key)

Cipher Block Chaining (CBC)

Blocks are encrypted/decrypted via the following equations:

$$\begin{cases} c_0 = \mathrm{IV} \\ c_i = \mathbb{E}_k(m_i \oplus c_{i-1}), \text{ for } i \geq 1 \end{cases} \quad \Bigg| \quad m_i = \mathbb{D}_k(c_i) \oplus c_{i-1}, \text{ for } i \geq 1$$

The *Initialization Vector* (IV) ensures that two encryptions of the same ptx produce different ctxs

- IV is a random value usually transmitted in clear from the encryptor to the decryptor as part of the ctx message
- IV may be fixed value if the key k of the cipher is used only once
- **IV can be sent encrypted in ECB mode before the rest of ctx blocks**

Cipher Block Chaining (CBC)

A *Chaining* strategy is the easiest way to add 'context' to each ctx block

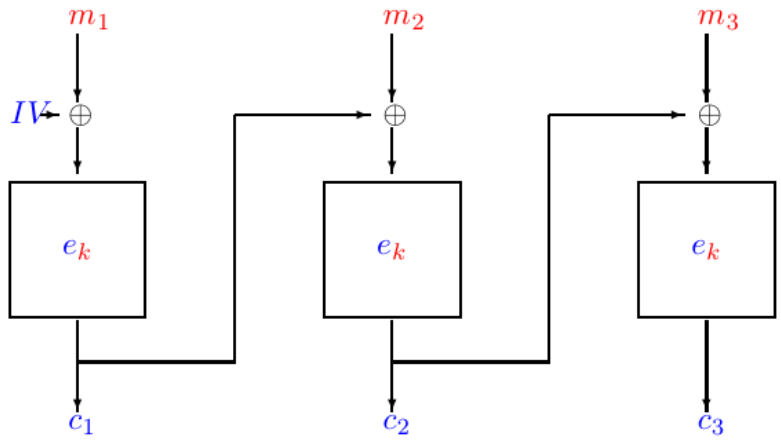
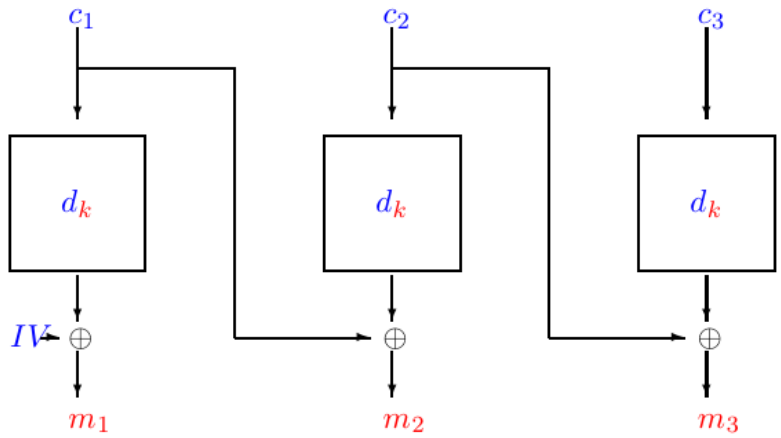


FIGURE 10. CBC decipherment



Remarks on CBC

Strengths:

- The encryption of a block depends on itself and all blocks before it. So, repeated ptx blocks are encrypted differently
- A ptx block can be recovered from two adjacent blocks of ctx $\Rightarrow$ decryption can be parallelized
- A one-bit change to the ctx corrupts the corresponding ptx block, and inverts the corresponding bit in the next ptx block.
  Further error propagation is avoided (CBC is self-recovering for intra-block errors)
- insertion or deletion attacks aiming to replace/insert/delete some blocks are detected!

Remarks on CBC

Weaknesses:

- Its main drawback is that the **encryption** cannot be parallelized
- No recover against synchronization errors is possible:
  - If a bit is inserted/added or lost from the cipher-text string, then all subsequent blocks are garbled
- An adversary can alter a ctx block in such a way to arbitrarily modify the following ptx block. However this destroys the corresponding ptx block.
- Reusing the same IV/Key on two messages yields identical CTXs up to the first difference in ptxs

Stream based Modes

Given a plaintext message $m \in \mathcal{M}$ as a sequence of blocks $m = \langle m_1, m_2, \ldots, \rangle$ the CFB, OFB and CTR modes of operation generate a key stream $k_1, k_2, k_3 \ldots$ (each key has the size of a block) to mask the ptx m:

$$c_i = m_i \oplus k_i, \text{ for } i \geq 1$$

Cipher FeedBack mode (CFB)

The plaintext message $m \in \mathcal{M}$ is broken into blocks of j bits ($1 \leq j \leq n$):
$m = \langle m_1, m_2, m_3, \ldots \rangle$
A n-bit block cipher employed in CFB mode is provided with an n-bit Input Shift Register (`ISR`) and an n-bit Output Shift Register (`OSR`).

1. The `ISR` is initially filled with an initialization vector (IV)
2. The encryption algorithm is run once to output n bits into `OSR`
3. The leftmost j bits of `OSR` are then `xor`'ed with a group of j ptx bits
4. The result of this `xor` operation is sent over the network and fed back to the `ISR`, shifting the leftmost j bits out
5. The encryption algorithm is run again and the next group of j bits is encrypted in the same fashion

Cipher feedback mode (CFB)

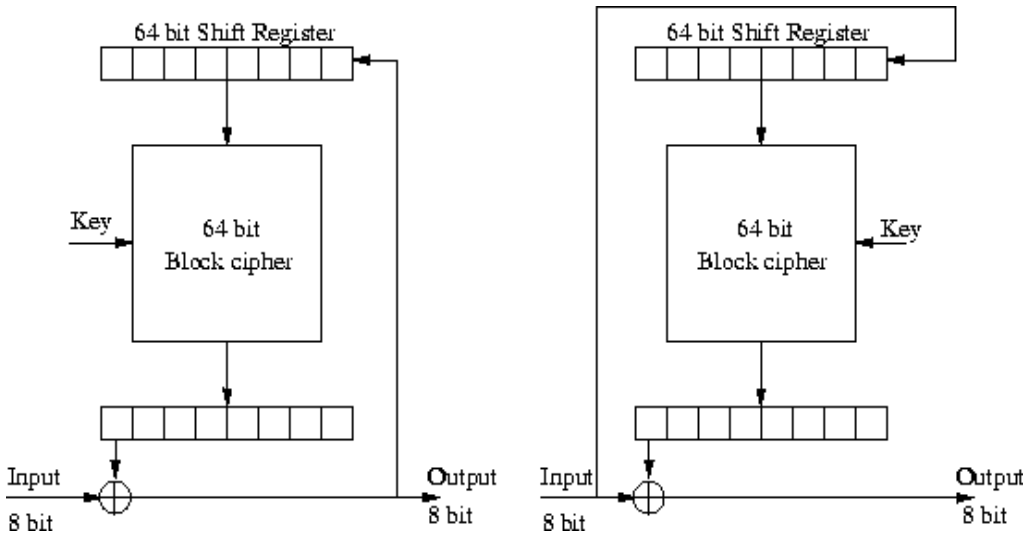
Encryption

$$\begin{cases} \mathtt{ISR}_0 \leftarrow \mathrm{IV} \\ \mathtt{OSR}_i \leftarrow \mathbb{E}_k(\mathtt{ISR}_{i-1}), \text{ for } i \geq 1 \\ c_i \leftarrow m_i \oplus j\text{-leftmost bits of } \mathtt{OSR}_i \\ \mathtt{ISR}_i \leftarrow (\mathtt{ISR}_{i-1} << j) \oplus c_i \end{cases}$$

Decryption

$$\begin{cases} \mathtt{ISR}_0 \leftarrow \mathrm{IV} \\ \mathtt{OSR}_i \leftarrow \mathbb{E}_k(\mathtt{ISR}_{i-1}), \text{ for } i \geq 1 \\ m_i \leftarrow c_i \oplus j\text{-leftmost bits of } \mathtt{OSR}_i \\ \mathtt{ISR}_i \leftarrow (\mathtt{ISR}_{i-1} << j) \oplus c_i \end{cases}$$

- The IV has the same properties discussed for the CBC mode
- It's useful in some applications such as encrypting interactive terminal inputs



Remarks on CFB

Strengths:

- The transmitted information comes in the form of arbitrarily size data
- Used with j=1-bit, a one bit de-synchronization is automatically recovered n+1 positions after the inserted or deleted bit

Weaknesses:

- Bit errors will corrupt the ptx block at the same bit positions. The corrupted cipher block will then be fed to the ISR and cause bit errors in the ptx for as long as the erroneous bits stay in ISR. After that, the system recovers, and all following bytes are decrypted correctly
- It is subject to insertion attacks/deletion attacks that spoil the block synchronization boundaries
- Cannot recover from lost ctx segments; if a ctx segment is lost, all following segments will be decrypted incorrectly (if the receiver is not aware of the segment loss)

Output Feedback mode (OFB)

The Mode is similar to CFB, except that ISR is fed back with OSR instead of the ctx

Encryption

$$\begin{cases} \texttt{ISR}_0 \leftarrow \text{IV} \\ \texttt{OSR}_i \leftarrow \mathbb{E}_k(\texttt{ISR}_{i-1}), \text{ for } i \geq 1 \\ c_i \leftarrow m_i \oplus j\text{-leftmost bits of } \texttt{OSR}_i \\ \texttt{ISR}_i \leftarrow (\texttt{ISR}_{i-1}\texttt{<<}j) \oplus j\text{-leftmost bits of } \texttt{OSR}_i \end{cases}$$

Decryption

$$\begin{cases} \texttt{ISR}_0 \leftarrow \text{IV} \\ \texttt{OSR}_i \leftarrow \mathbb{E}_k(\texttt{ISR}_{i-1}), \text{ for } i \geq 1 \\ m_i \leftarrow c_i \oplus j\text{-leftmost bits of } \texttt{OSR}_i \\ \texttt{ISR}_i \leftarrow (\texttt{ISR}_{i-1}\texttt{<<}j) \oplus j\text{-leftmost bits of } \texttt{OSR}_i \end{cases}$$

Remarks on OFB

Strengths:

- The transmitted information comes in the form of arbitrarily size data
- The encryption process can be partially parallelized as the values of $\mathtt{OSR}$ can be pre-computed
- In OFB, the bit error(s) in the decrypted ctx block (or segment) occur in the same bit position(s) as in the ctx block (or segment); the other bit positions are not affected

Weaknesses:

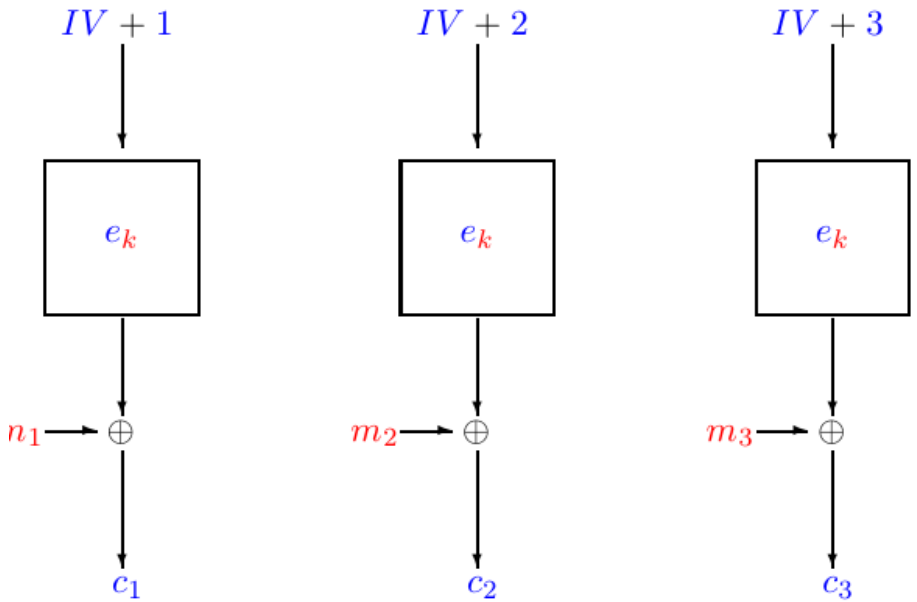
- bit errors in the IV affect the decryption of every ctx block
- It is subject to malicious bit insertion/deletion into the ctx, thus spoiling the synchronization of the block (or segment) boundaries;
- Cannot recover from lost ctx segments; if a ctx segment is lost, all following segments will be decrypted incorrectly (if the receiver is not aware of the segment loss)

CounTeR mode (CTR)

The plaintext message $m \in \mathcal{M}$ is broken into blocks of equal size $m = \langle m_1, m_2, \ldots, \rangle$ The encryption proceeds for the i-th block, by encrypting the value of $\mathrm{IV}+i$ and then xor'ing this with the message block:

$$\begin{cases} \mathtt{ctr}_i \leftarrow \mathrm{IV} + i, \ i \geq 0 \\ t_i \leftarrow \mathbb{E}_k(\mathtt{ctr}_i) \\ c_i \leftarrow m_i \oplus t_i \end{cases}$$

The *Initialization Vector* (IV) must be a random number to make sure that two encryptions of the same ptx produce different ctxs



Remarks on CTR

Strengths:

- Only the encryption primitive is needed
- Fast encryption/decryption; blocks can be processed (encrypted or decrypted) in parallel; good for high speed links
- Random access to encrypted data blocks
- Bit errors in a ctx block cause errors only in the same bit position(s) of the decrypted block

Weaknesses:

- IV should be unpredictable, and must not be reused
- an error in a certain ctx block affects the whole decrypted ctx block; however, no other block is affected

Currently, to simplify the detection of errors (or intentional alterations) in the transmitted data the so-called authenticated modes of operations e.g., GCM (Galois Counter Mode) are used.