Block Ciphers

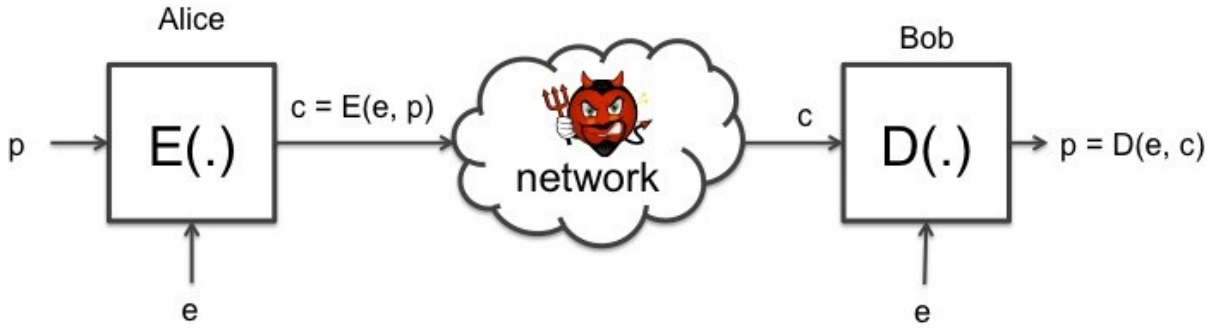
Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
gianluca.dini@unipi.it
Version: 10/03/2025

Block Ciphers

GENERAL CONCEPTS

Block cipher

- Block ciphers break up the plaintext in blocks of fixed length *n* bits and encrypt one block at time



- E_k: $\{0,1\}^n \rightarrow \{0,1\}^n$ D_k: $\{0,1\}^n \rightarrow \{0,1\}^n$
- E is a keyed permutation: $E(k, p) = E_k(p) = Enc_k(p)$
- $E_K(\cdot)$ is a permutation

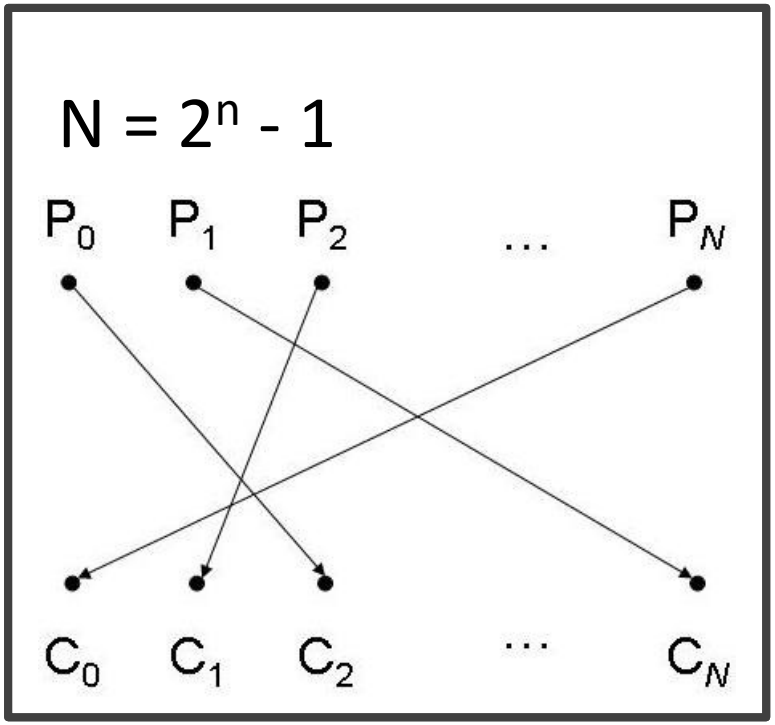
Permutation

- E_k is a permutation
  - E_K is efficiently computable
  - E_k is bijective
    - Surjective (or onto)
    - Injective (or one-to-one)
  - E_k^{-1} is efficiently computable

Examples

- Block ciphers
  - DES n = 64 bits, k = 56 bits
  - 3DES n = 64 bits, k = 168 bits
  - AES n = 128 bits k = 128, 192, 256 bits

Random permutations



A possible random permutation π

- Let $Perm_n$ be the set of all permutations π: $\{0,1\}^n \rightarrow \{0,1\}^n$
- $|Perm_n| = 2^n!$
- A true random cipher
  - implements all the permutations in $Perm_n$
  - uniformly selects a permutation $\pi \in Perm_n$ at random

True Random Cipher

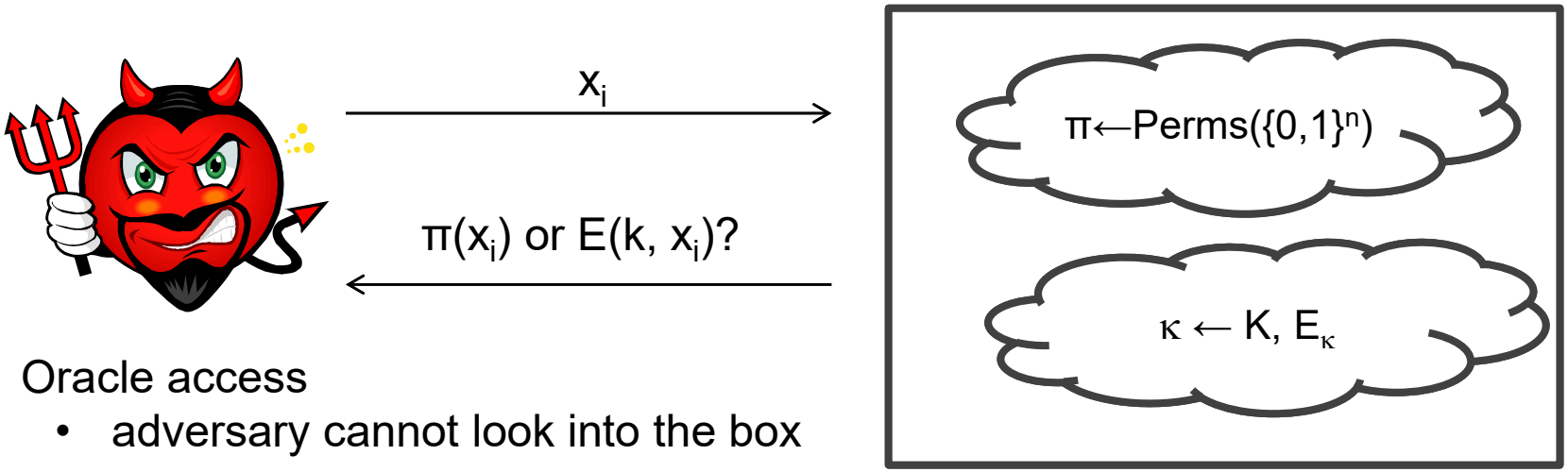
- A True random cipher is perfect
- A true random cipher implements all possible Random permutations ($2^n!$)
  - Need a uniform random key for each permutation (naming)
    - key size := $\log_2 (2^n!) \approx (n - 1.44)\, 2^n$
      - Exponential in the block size!
      - The block size cannot be small to avoid a dictionary attack
- A true random cipher cannot be implemented

Pseudorandom permutations

- Consider a *family of permutations* parametrized by $\kappa \in \mathrm{K} = \{0, 1\}^k$, $E_\kappa: \{0,1\}^n \rightarrow \{0,1\}^n$
- A E_κ is a *pseudorandom permutation* (PRP) if it is indistinguishable from a uniform random permutation by a limited adversary
- $| \{E_\kappa\} | = 2^k << |\mathrm{Perm}_n|$, with $|\kappa| = k$
- A block cipher is a practical instantiation of a PRP

Practical block cipher

- In practice, the encryption function corresponding to a randomly chosen key should appear as a randomly chosen permutation to a limited adversary



- Oracle access
  - adversary cannot look into the box

Exhaustive key search attack

- The attack
  - Given a pair (pt, ct), check whether ct == E_{ki}(pt), i = 0, 1, …, $2^k - 1$
    - Known-plaintext attack
    - Time complexity: $O(2^k)$
- False positives
  - Do you expect that just one key k maps pt into ct?
  - How many keys (false positives) do we expect to map pt into ct?
  - How do you discriminate the good one?

Exhaustive key search

- False positives
  - Do you expect that just one key k maps pt into ct?
  - How many keys (false positives) do we expect to map pt into ct?
  - How do you discriminate the good one?

False positives

- Problem: Given (ct, pt) s.t. ct = E_{k^*}(pt) for a given k*, determine the number of keys that map pt into ct
- Solution.
  - Given a certain key k, P(k) = Pr[E_k(pt) == ct] = $1/2^n$
  - The *expected* number of keys that map pt into ct is $2^k \times 1/2^n = 2^{k-n}$

False positives

- Example 1 – DES with n = 64 and k = 56
  - On average 2^{-8} keys map pt into ct
  - One pair (pt, ct) is sufficient for an exhaustive key search
- Example 2 – Skipjack with n = 64 and k = 80
  - On average 2^{16} keys map pt into ct
  - Two or more plaintext-ciphertext pairs are necessary for an exhaustive key search

False positives

- Consider now t pairs (pt_i, ct_i), i = 1, 2,…, t
  - Given k, $Pr[E_k(pt_i) = ct_i$, for all i = 1, 2,…, t$] = (1/2^n)^t = 1/2^{tn}$
  - Expected number of keys that map pt_i into ct_i, for all i = 1, 2, …, t, is $2^k/2^{tn} = 2^{k-tn}$
- Example 3 – Skypjack with k = 80, n = 64, t = 2
  - The expected number of keys is = $2^{80 - 2\times64} = 2^{-48}$
  - Two pairs are sufficient for an exhaustive key search

False positives

- THEOREM
  - Given a block cipher with a key lenght of *k* bits and a block size of *n* bits, as well as *t* plaintext-ciphertext pairs, (pt_1, ct_1),…, (pt_t, ct_t), the expected number of false keys which encrypt all plaintexts to the corresponding ciphertexts is $2^{k - tn}$
- FACT
  - Two input-output pairs are generally enough for exhaustive key search

Block ciphers

EXERCISES

Exercise 1 - Exhaustive key search

- Exhaustive key search is a known-plaintext attack
- However, the adversary can mount a cyphertext-only attack if (s)he has some knowledge on PT

Exercise 1 – exhaustive key search

- Assume DES is used to encrypt 64-bit blocks of 8 ASCII chars, with one bit per char serving as parity bit
- How many CT blocks the adversary needs to remove false positives with a probability smaller than ε?
- Answer: $2^{-8t} < \varepsilon$, with t number of ct-blocks
  - With DES, t = 10 is sufficient for the most practical uses

Exercise 2 - dictionary attack

- Consider a block cipher with k and n.
- The adversary has collected D pairs (pt_i, ct_i), i = 1,…, D, with D << 2^n (the dictionary)
- Now the adversary reads C newly produced cyphertexts ct^*_j, j = 1,…, C.
- Determine the value of C s.t. the Pr[Exists j, j = 1, 2,… C, s.t. c^*_j is in the dictionary] = P
- Answer: C = $2^n/D$

Exercise 3 - Rekeying

- An adversary can successfully perform an exhaustive key search in a month.
- Our security policy requires that keys are changed every hour.
- What is the probability P that, in a month, the adversary is able to find any key before it is changed?
  - For simplicity assume that every month is composed of 30 days.
- What if we refresh key every minute?
- Answer: P = 0.63.

Symmetric Encryption

MULTIPLE ENCRYPTION AND KEY WHITENING

Increasing the Security of Block Ciphers

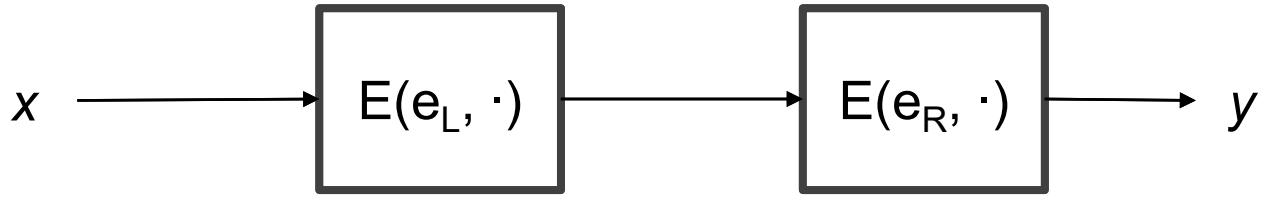
- DES is a secure cipher, no efficient cryptanalys is known
- DES does not define a group
- DES key has become too short
- Can we improve the security of DES?
- Yes, by means of two techniques
  - Multiple encryption
  - Key whitening

DES does not define a group

- If DES were a group then $\forall k_1, k_2 \in \mathcal{K}, \exists k_3 \in \mathcal{K}$ s.t. $\forall x \in \mathcal{M}, E_{k_2}\left(E_{k_1}(x)\right) = E_{k_3}(x)$
- So, double (multiple) encryption would be useless
- Furthermore, DES would be vulnerable to Meet-in-the-Middle attack that runs in 2^{28}

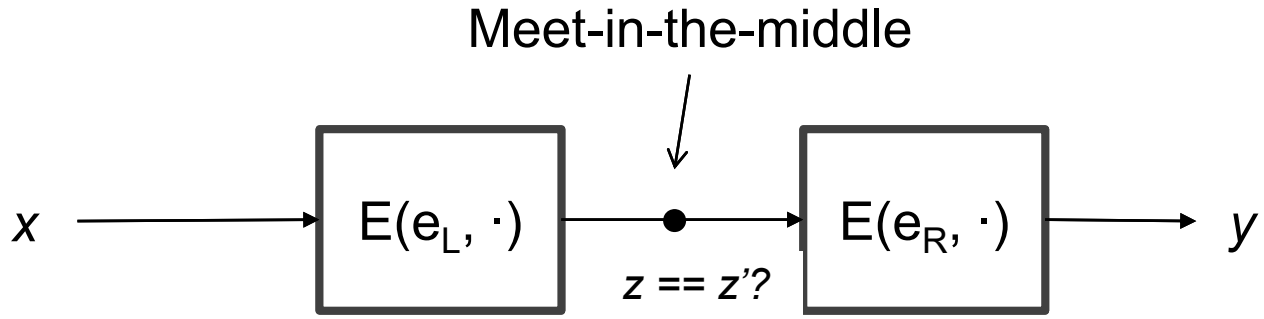
Two-times Encryption (2E)

- $y = 2E((e_L, e_R), m) = E(e_R, E(e_L, x))$
  - key size is 2k bits
  - Brute force attack requires 2^{2k} steps
  - 2E is two times slower than E
- Is it really more secure than single encryption?
- Meet-in-the-middle attack



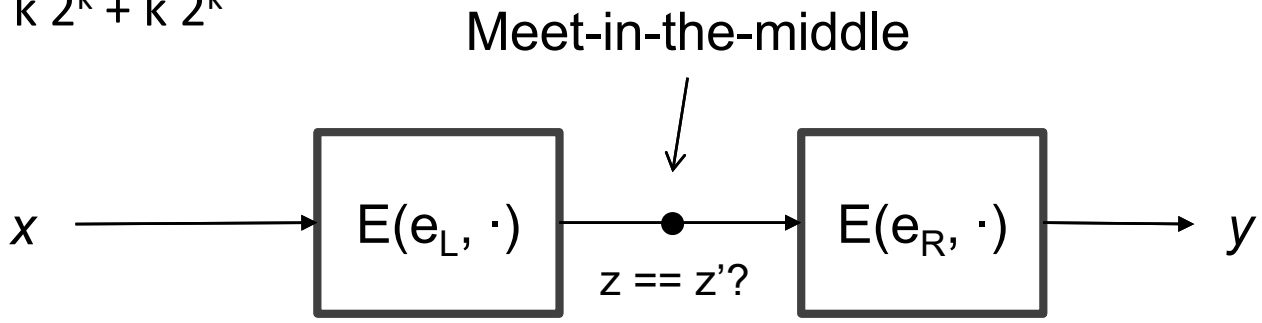
Meet-in-the-middle attack

- Attack Sketch
  1. Build a table T containing $z = E(e_L, x)$ for all possible keys e_L. Keep T sorted according to z.
  2. Check whether $z' = D(e_R, y)$ is contained in the table T, for all possible key e_R.
     1. If z' in contained in T then (e_L, e_R) maps x into y with e_L s.t. $T[e_L] = z'$.



Meet-in-the-middle attack

- Attack complexity
  - Data complexity: negligible.
  - Storage complexity: $O(2^k)$.
    - Storage necessary for table T.
  - Time complexity: $O(k2^k)$.
    - Time complexity for step 1 + Time complexity for step 2 = Time for building and sorting the table + Time for searching in a sorted table = $k\,2^k + k\,2^k$



Two-times DES

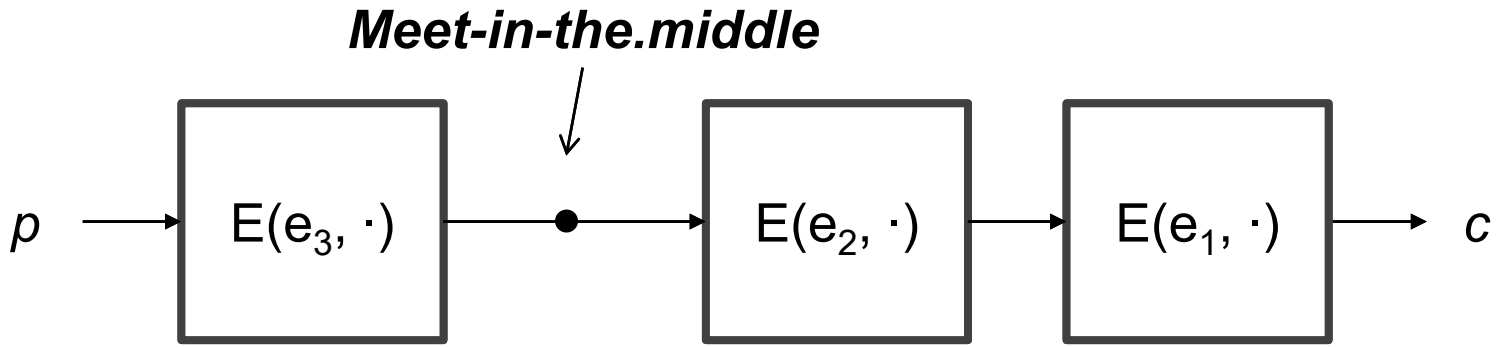
- 2DES
  - Time complexity: 2^{56} (doable nowadays!)
  - Space complexity: 2^{56} (lot of space!)
  - 2DES brings no advantage

Triple DES (3DES)

- EDE scheme
  - Standard ANSI X9.17 and ISO 8732
  - $Y = 3E((e_1, e_2, e_3), x) = E(e_1, D(e_2, E(e_3, x)))$
    - If $e_1 = e_2 = e_3$, 3DES becomes DES
      - backward compatibility
  - Key size = 168-bits
  - 3 times slower than DES
  - Simple attack ≈ 2^{118}

3DES – meet-in-the-middle attack

- Time = 2^{112} (undoable!)
- Space = 2^{56} (lot of space!)



False positives for multiple encryption

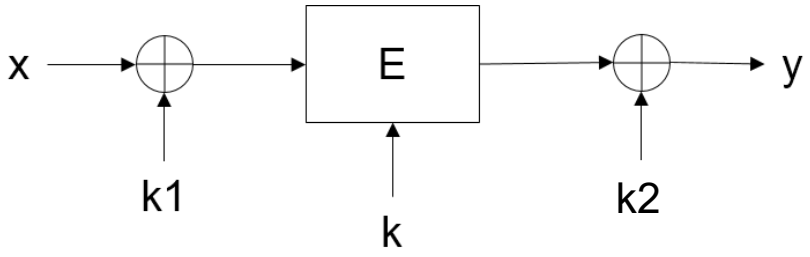
- THEOREM
  - Given there are r subsequent encyptions with a block cipher with a key lenght of k bits and a block size of n bits, as well as t plaintext-ciphertext pairs, $(pt_1, ct_2),\ldots, (pt_t, ct_t)$, the expected number of false keys which encrypt all plaintext to the corresponsig ciphertext is $2^{rk - tn}$

Limitations of 3DES

- 3DES resists brute force but
  - It is not efficient regarding software implementation
  - It has a short block size (n = 64)
    - A drawback if you want to make a hash function from 3DES, for example
  - Key lengths of at least 256-bit are necessary to resist quantum computing attack

Key whitening

- Considerations
  - KW is not a “cure” for weak ciphers
- Applications
  - DESX: a variant of DES
  - AES: uses KW internally
- Performance
  - Negliglible overhead w.r.t. E (Just two XOR’s!)



Definition 5.3.1 Key whitening for block ciphers

Encryption: $y = e_{k,k_1,k_2}(x) = e_k(x \oplus k_1) \oplus k_2$.

Decryption: $x = e^{-1}_{k,k_1,k_2}(x) = e^{-1}_k(y \oplus k_2) \oplus k_1$

Key whitening

- Attacks
  - Brute-force attack
    - Time complexity: 2^{k+2n} encryption ops
  - Meet-in-the-middle:
    - Time complexity 2^{k+n}
    - Storage complexity: 2^n data sets
  - The most efficient attack
    - If the adversary can collect 2^m pt-ct pairs, then time complexity becomes 2^{k+n-m}
      - The adversary cannot control m (rekeying)
    - Example: DES (m = 32)
      - Time complexity 2^{88} encryptions (nowadays, out of reach)
      - Storage complexity 2^{32} pairs = 64 GBytes of data (!!!)

Symmetric Encryption

ENCRYPTION MODES

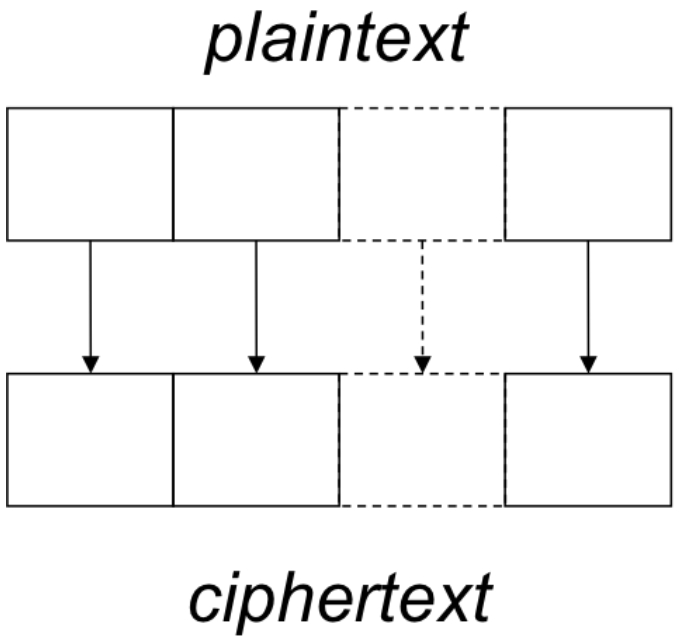
Encryption Modes

- A block cipher encrypts PT in fixed-size *n*-bit blocks
- When the PT len exceeds n bits, there are several modes to use the block cipher
  - Electronic Codebook (ECB)
  - Cipher-block Chaining (CBC)

Other encryption modes

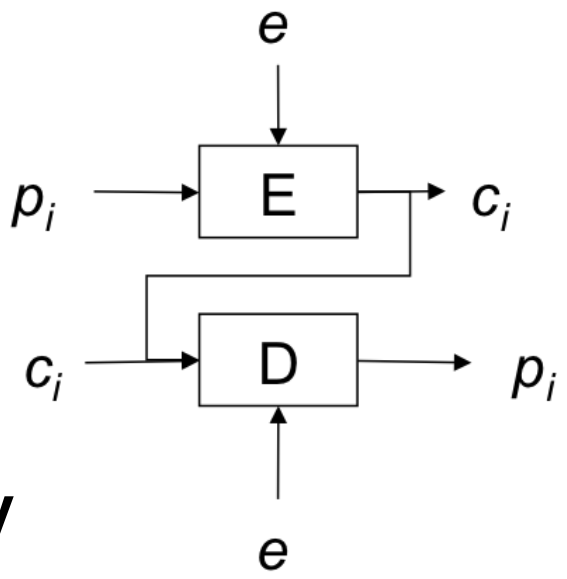
- Other encryption modes
  - To build a stream cipher out of a block cipher
    - Cipher Feedback mode (CFB)
    - Output Feedback mode (OFB)
    - Counter mode (CTR)
  - Authenticated encryption
    - Galois Counter mode (GCM, CCM, …)
  - and many others (e.g., CTS, …)
- Block ciphers are very versatile components

Electronic codebook



$$\forall 1 \le i \le t, c_i \leftarrow E\left(e, p_i\right)$$

$$\forall 1 \le i \le t, p_i \leftarrow D\left(e, c_i\right)$$



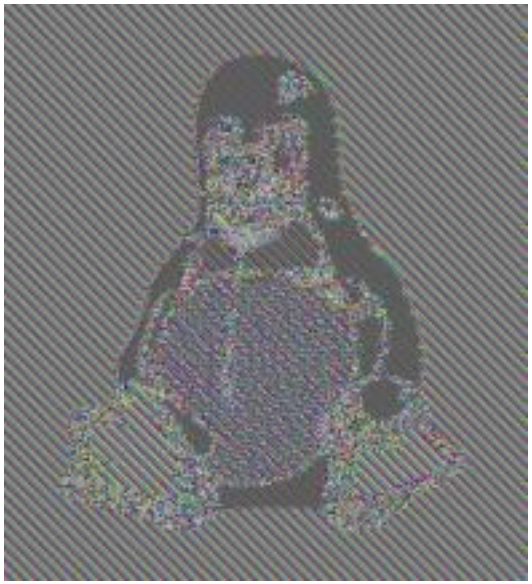
PT blocks are encrypted separately

ECB - properties

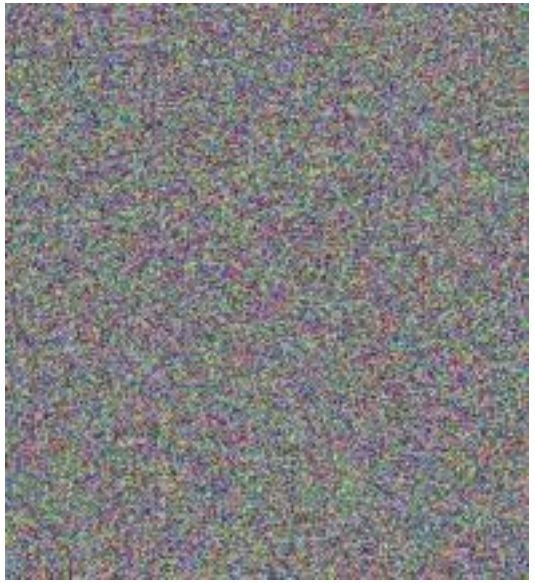
- PROS
  - No error propagation
    - One or more bits in a single CT block affects decryption of that block only
  - Enc & Dec can be parallelized
- CONS (it is insecure)
  - Blocks are encrypted separately
    - Identical PT results in identical CT
      - ECB doesn’t hide data pattern
      - ECB allows traffic analysis
    - ECB allows block re-ordering and substitution

ECB doesn’t hide data patterns

Plaintext



ECB encrypted



Non-ECB encrypted

ECB – block attack

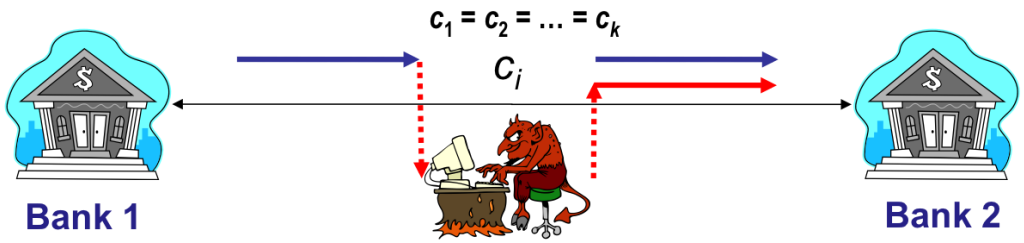
- Bank transaction that transfers a customer C's amount of money D from bank B1 to bank B2
  - Bank B1 debits D to C
  - Bank B1 sends the "credit D to C" message to bank B2
  - Upon receiving the message, Bank B2 credits D to C
- Credit message format
  - Src bank: M (12 byte)
  - Rcv banck: R (12 byte)
  - Customer: C (48 byte)
  - Bank account number: N (16 byte)
  - Amount of money: D (8 byte)
- Cipher: n = 64 bit; ECB mode

ECB – block attack

- Mr. Lou Cipher is a client of the banks and wants to make a fraud
- Attack aim
  - To replay Bank B1's message "credit 100$ to Lou Cipher" many times
- Attack strategy
  - Lou Cipher activates multiple transfers of 100$ so that multiple messages "credit 100$ to Lou Cipher" are sent from B1 to B2
  - The adversary identifies at least one of these messages
  - The adversary replies the message several times

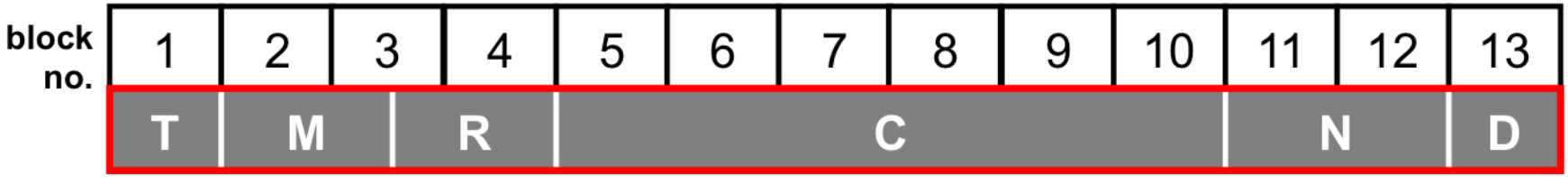
ECB – block attack

- The fraud
  1. Mr. Lou Cipher performs k equal transfers
     - credit 100$ to Lou Cipher → c1
     - credit 100$ to Lou Cipher → c2
     - ...
     - credit 100$ to Lou Cipher → c_k
  2. Then, he searches for “his own” CTs, namely k equal CTs!
  3. Finally he replies one of these cryptograms (many times)



ECB – block attack

- The message lacks any notion of time so it can be easily replied
- An 8-byte timestamp field T (block #1) is added to the message to prevent replay attacks
- A replied message can now be discarded



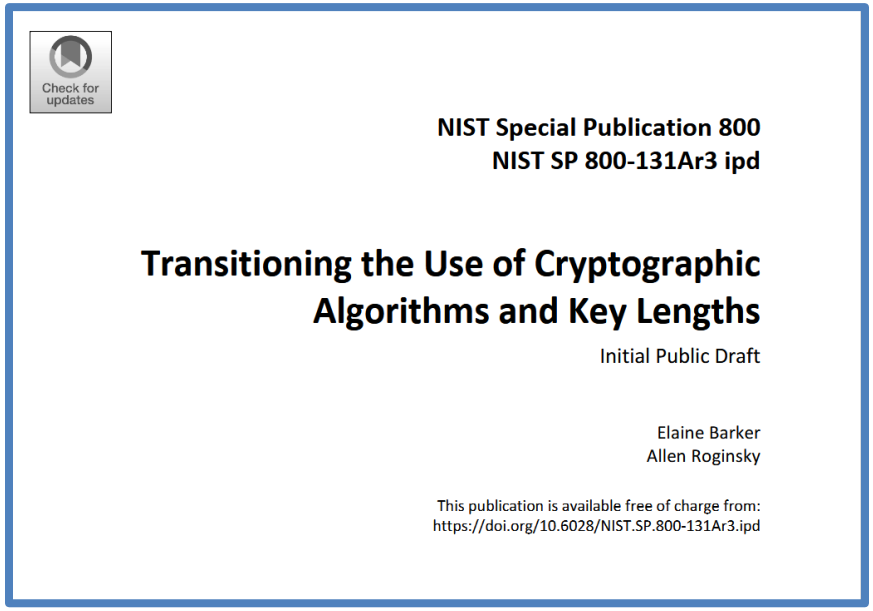
ECB – block attack

- However, Mr Lou Cipher can still perform the attack
  1. Identify “his own” CTs by inspecting blocks #2-#13
  2. Select any his-own-CT
  3. Substitute block #1 of his-own-CT with block #1 of any intercepted “fresh” block
  4. Replay the resulting CT

ECB is disallowed

Table 2. Approval status of the block cipher modes of operation for AES encryption and decryption

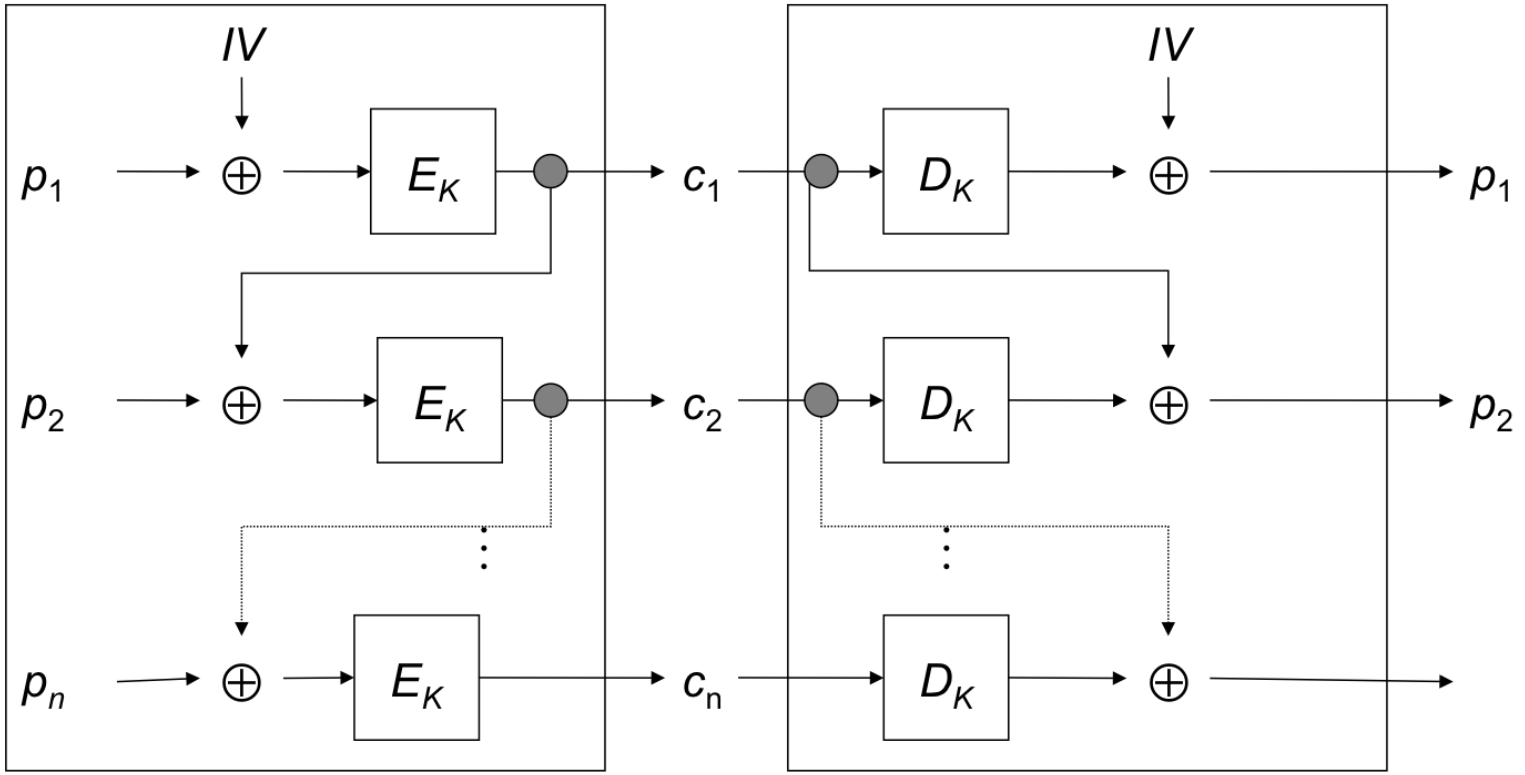
Publication	Mode	Status
SP 800-38A	ECB	Disallowed for data encryption Legacy use for decryption
	CBC	Acceptable
	CFB	Acceptable
	CTR	Acceptable
	OFB	Acceptable



Cipher block chaining (CBC)

Encryption: $c_0 \leftarrow IV. \forall 1 \le i \le t, c_i \leftarrow E_k\left(p_i \oplus c_{i-1}\right)$

Decryption: $c_0 \leftarrow IV. \forall 1 \le i \le t, p_i \leftarrow c_{i-1} \oplus D_k\left(c_i\right)$



CBC – properties (→)

- CBC mode is CPA-secure.
- CBC-Enc is *randomized* by using IV (nonce).
  - Identical ciphertext results from the same PT under the same key and IV.
- Chaining dependencies: c_i depends on p_i and the preceding CT block c_{i-1}
- CT-block reordering affects decryption
- CBC suffers from Error propagation
  - Bit errors in c_i affect p_i and p_{i+1} (*error propagation*)

CBC – properties

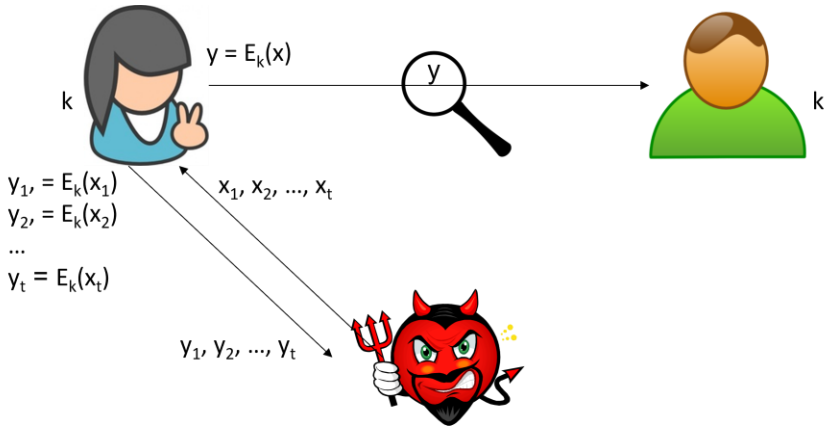
- IV can be sent in the clear but its integrity must be guaranteed
- Cyphertext expansion is just one block (IV)
- Only CBC-dec can be parallelized

CBC – block attack

- If Bank A chooses a random IV for each wire transfer the attack will not work.
- However, if Lou Cipher substitutes blocks #5–10 and #13, bank B would decrypt *account number* and *deposit amount* to random numbers ➔
  - This is highly undesirable!
  - Encryption itself is not sufficient, we need additional mechanisms (MDC, MAC, digsig) to protect integrity

Chosen-Plaintext Attack (Informal)

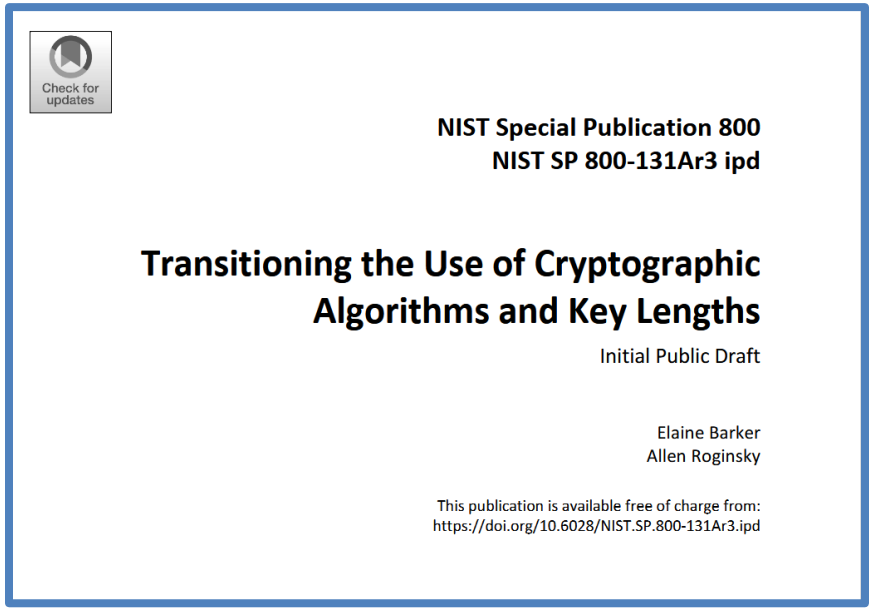
- CPA Attack
  - Attacker *makes* the sender to encrypt $x_1,\ldots, x_t$
    - The attacker may influence or control encryption
  - The sender encrypts and transmits $y_1 = E_k(x_1), \ldots, y_t = E_k(x_t)$
  - Later on, the sender encrypts x and transmits $y = E_k(x)$
- CPA-security guarantees that the adversary cannot learn anything about x
- The encryption scheme must be randomized



CBC is acceptable

Table 2. Approval status of the block cipher modes of operation for AES encryption and decryption

Publication	Mode	Status
SP 800-38A	ECB	Disallowed for data encryption Legacy use for decryption
	CBC	Acceptable
	CFB	Acceptable
	CTR	Acceptable
	OFB	Acceptable



Block Ciphers

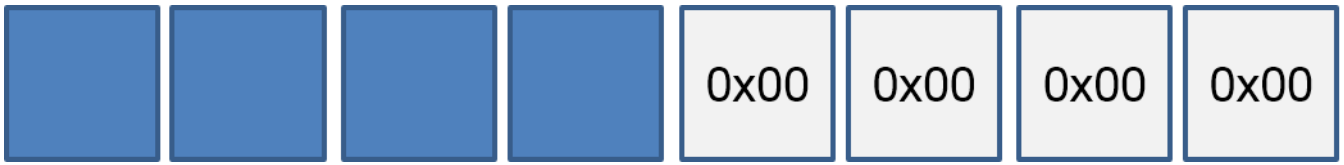
PADDING

Padding

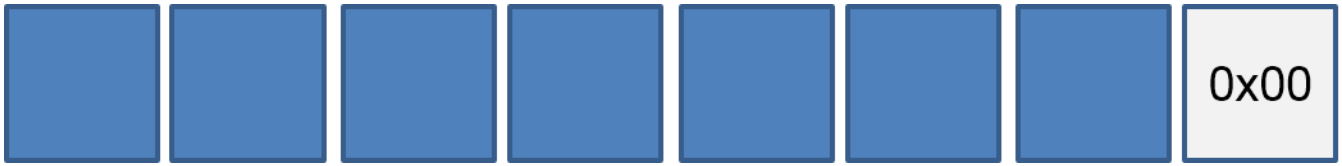
- Padding is necessary when PT len is not an integer multiple of the block

A naïve (wrong) solution

Pad the message with zeroes to the right, without ambiguous boundaries



Problem: What if the message was a NULL-terminated string?



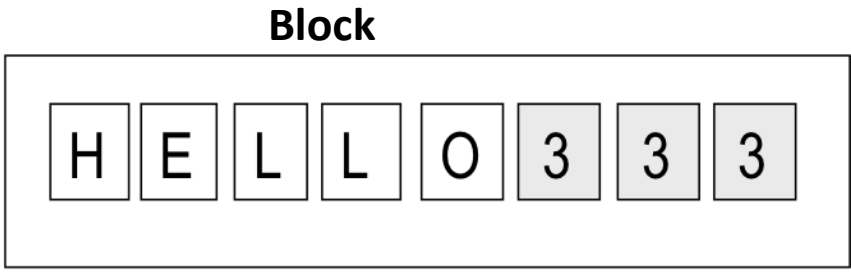
At the receiving side: Was it a NULL-terminated string or a 7-bytes pt?

The PKCS #5 padding scheme

If PT len is NOT a block multiple

- We need *b* padding bytes
- Fill each padding byte by *b*

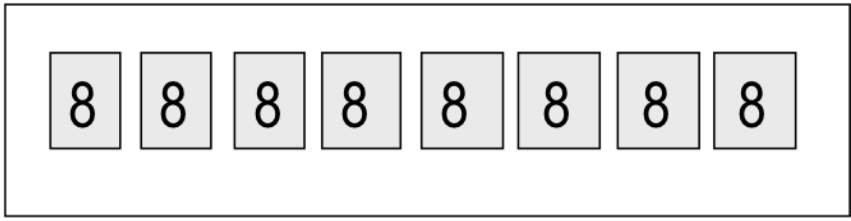
Example: b = 3 then append 0x030303



If PT len is a block multiple

Padding = block

Fill each padding block by 8



Padding causes *ciphertext expansion*

PKCS #5: encryption & decryption →

- Let *L* be the block length (in bytes) of the cipher
- Let *b* be the # of bytes that need to be appended to the plaintext to get its length a multiple of L, $1 \leq b \leq L$
- Before encryption
  - Append *b* (encoded in 1 byte), *b* times
  - Example: if b = 3, append 0x030303

PKCS #5: encryption & decryption

- After decryption, say the final byte has value *b*
  - If b == 0 or b > L, return “error”
  - If the trailing *b* bytes are not all equal to *b*, return “error”
  - Strip off the trailing b bytes and output the left as the message

PKCS #5 vs PKCS #7

- Difference between PKCS#5 and PKCS#7
- PKCS#5: padding is defined for 8-byte block sizes (RFC 2898)
- PKCS#7: padding is defined for block of any size ranging from 1 to 255 bytes (RFC 2315)

Block Ciphers | Padding

PADDING ORACLE ATTACK

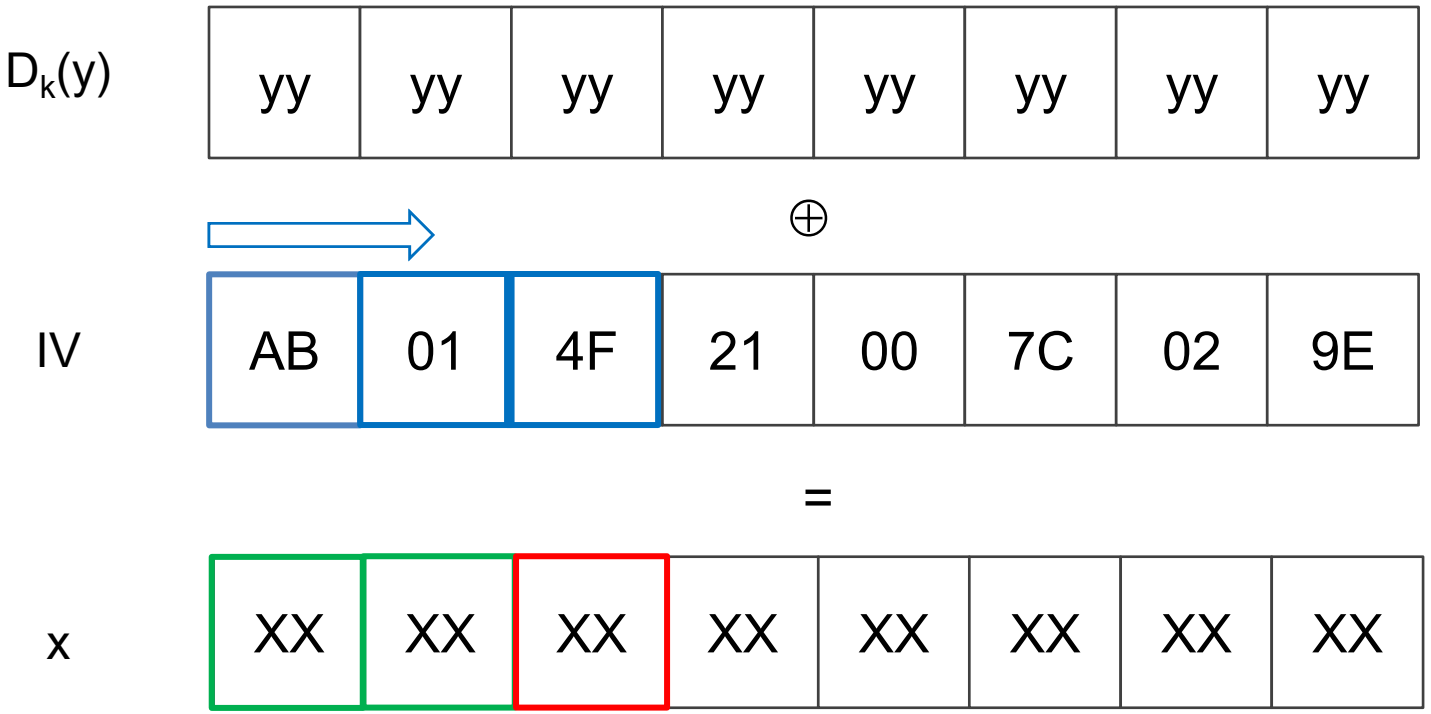
Padding Oracle Attack (CCA)

- The attacker
  - intercepts y and wants to obtain x (*ciphertext-only attack*)
  - modifies y into y’ and submits to the receiver
- The receiver (the padding oracle)
  - Receiver decrypts y’ and returns “error”, if x’ is not properly formatted (padding)
- On padding oracles
  - Frequently present in web applications
  - Error, receiver timing, receiver behaviour,…

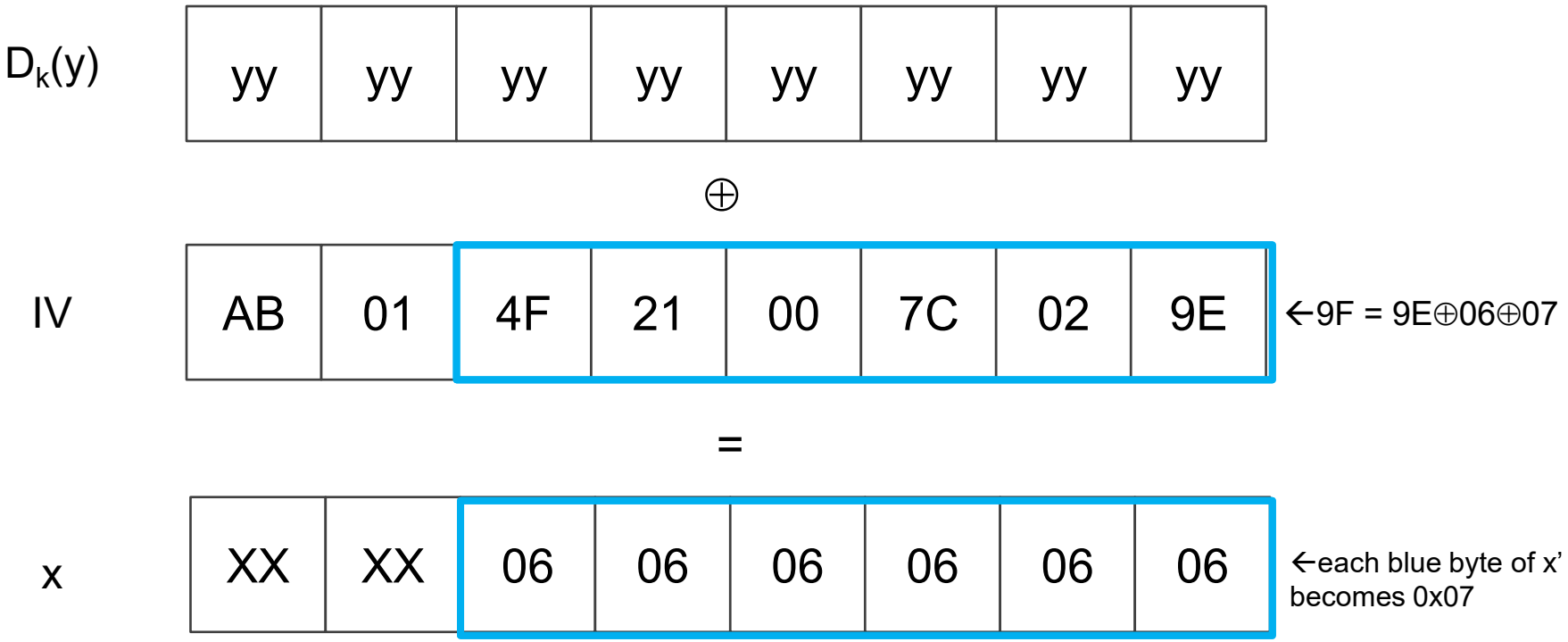
Main idea of the attack

- For simplicity, let CT be a two-block ciphertext (IV, y), with $y = Enc_k(x \oplus IV)$
- At the receiving site: $x = D_k(y) \oplus IV$
- Assume message x is well formatted in terms of padding
- Main intuition of the attack
  - If the attacker changes the i-th byte of IV, this causes a predictable change (only) to the i-th byte of x

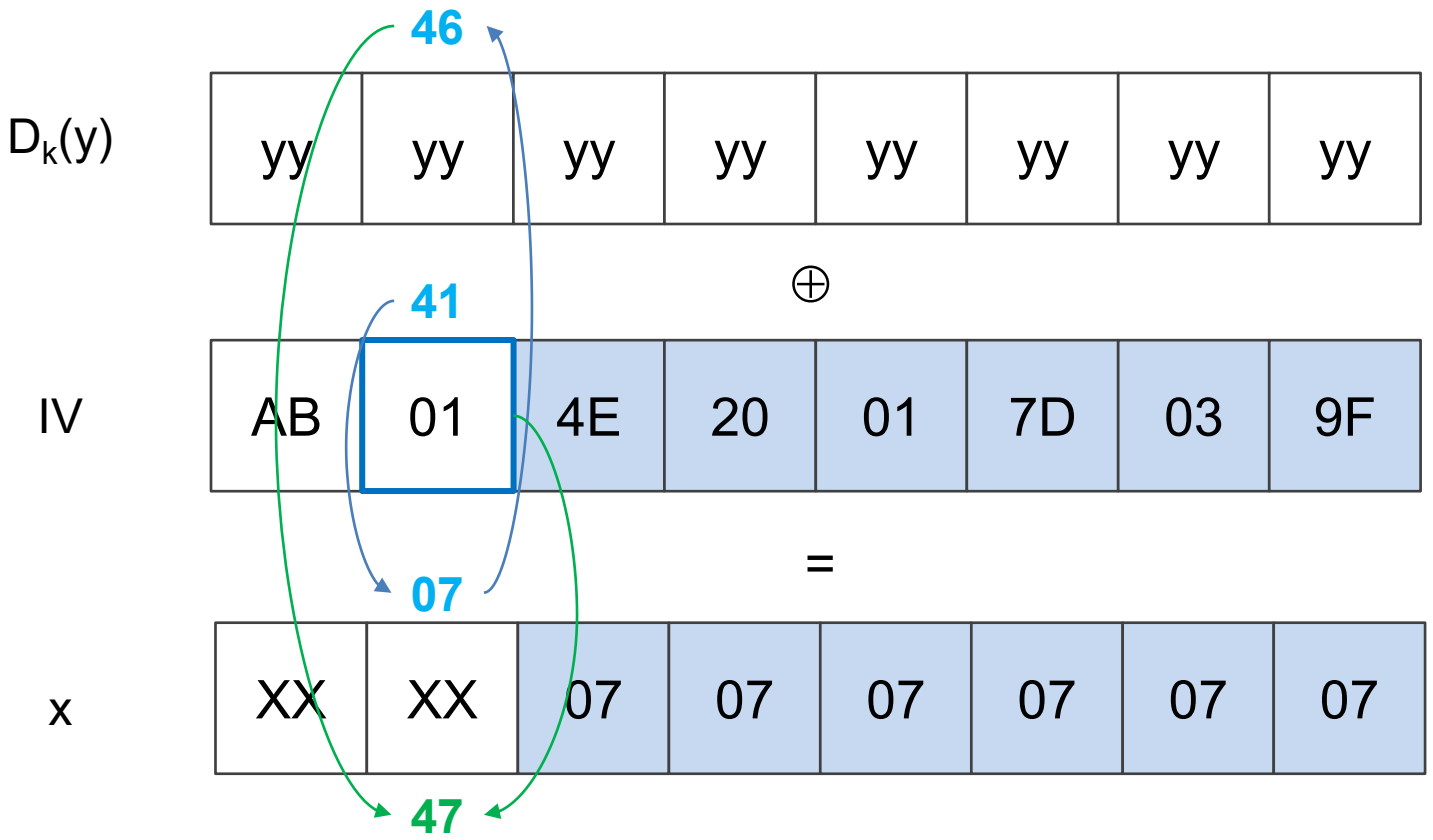
The attack – step 1 – determine padding lenght



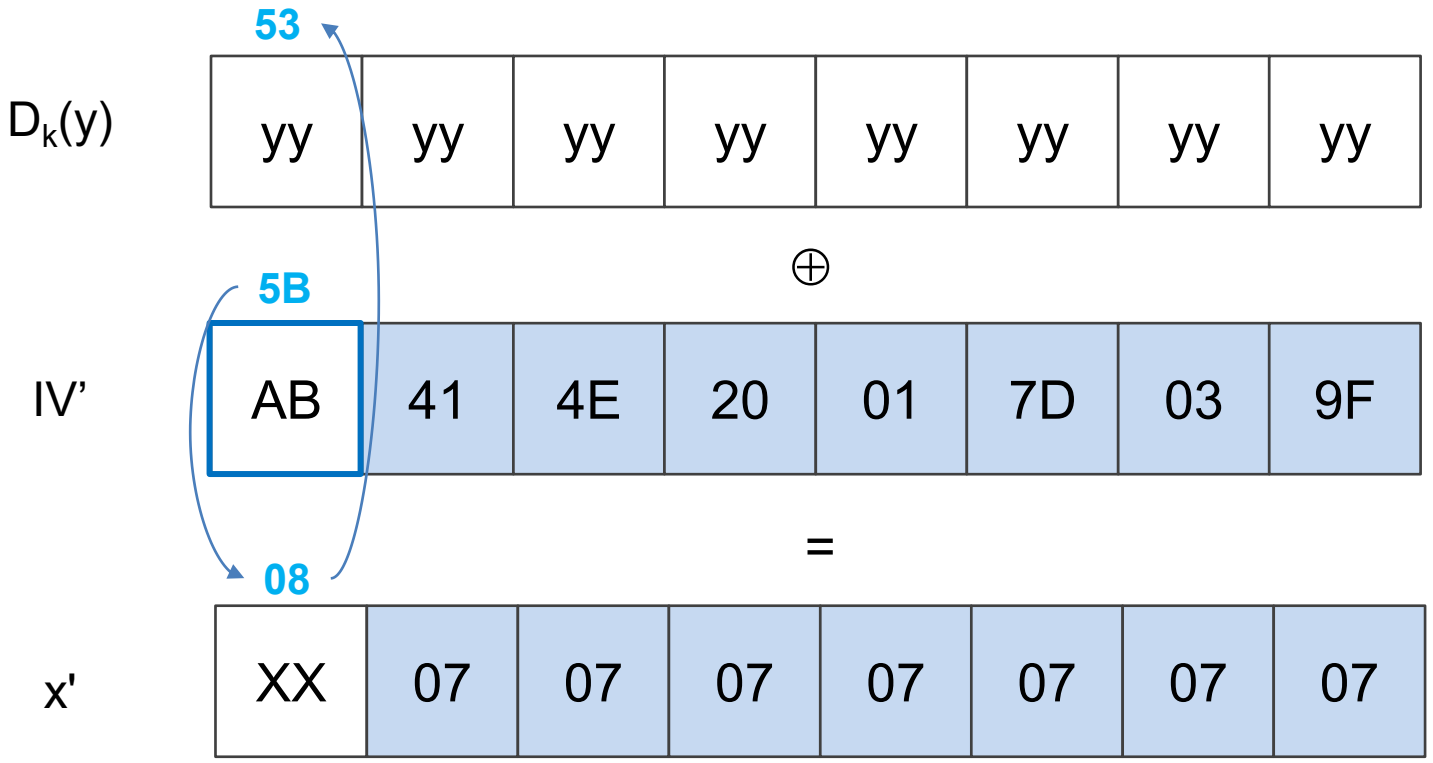
The attack – step 2a – determine pt



The attack – step 2b – determine pt



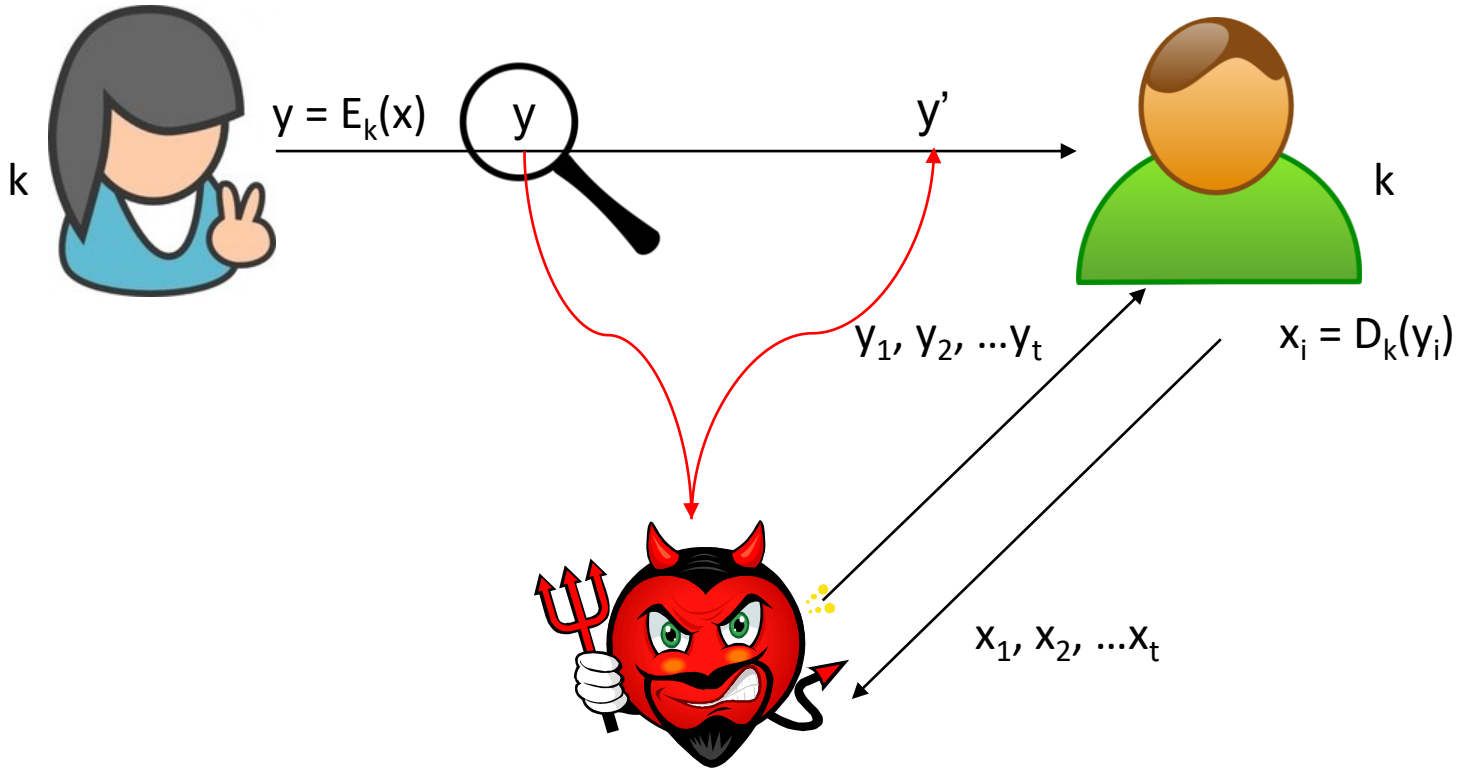
The attack – step 2c – determine pt



Attack complexity

- At most L tries to learn the # of padding bytes
- At most $2^8 = 256$ tries to learn each plaintext byte

CCA model



Chosen-ciphertext attack

- Now the attacker becomes active
- The CCA
  - The attacker intercepts $y = E_k(x)$ and modifies it into y'
  - The receiver decrypts y' and returns (the attacker) either x' or some information about x'
  - The adversary can derive either x or some information about x
- CCA and malleability
  - CCA-security implies non-malleability

CCA-security

- Chosen-ciphertext attacks represent a significant, real-world threat
- Modern encryption schemes are designed to be CCA-secure

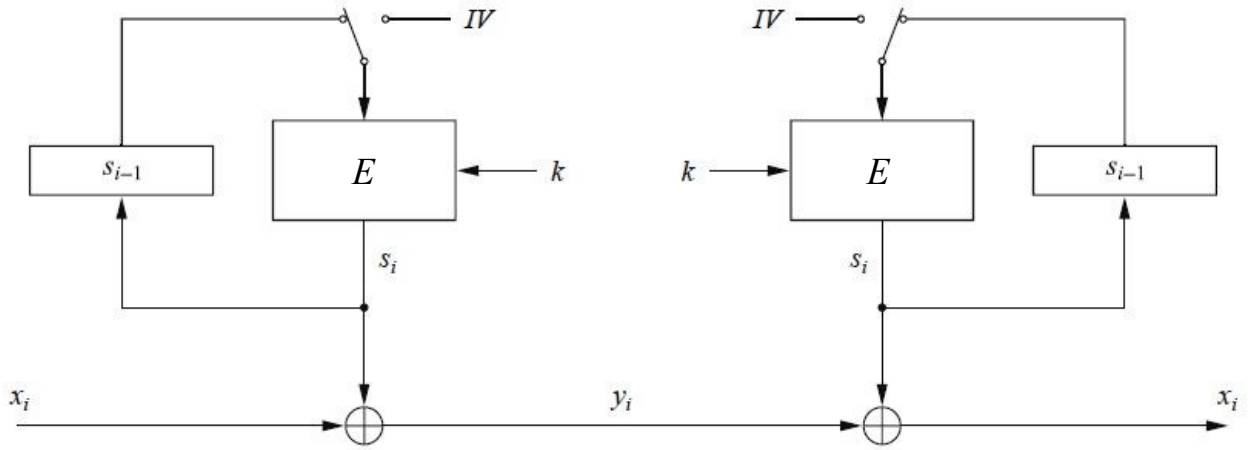
Block Ciphers | Encryption modes

OFB, CFB, CTR

Block cipher vs stream cipher

- Stream ciphers do not require padding
- Stream ciphers can operate in real-time

Output Feedback Mode (OFB) →



Let e() be a block cipher of block size b; let x_i, y_i and s_i be bit strings of length b; and IV be a nonce of length b.

Encryption (first block): $s_1 = e_k(IV)$ *and* $y_1 = s_1 \oplus x_1$

Encryption (general block): $s_i = e_k(s_{i-1})$ *and* $y_i = s_i \oplus x_i, \quad i \geq 2$

Decryption (first block): $s_1 = e_k(IV)$ *and* $x_1 = s_1 \oplus y_1$

Decryption (general block): $s_i = e_k(s_{i-1})$ *and* $x_i = s_i \oplus y_i, \quad i \geq 2$

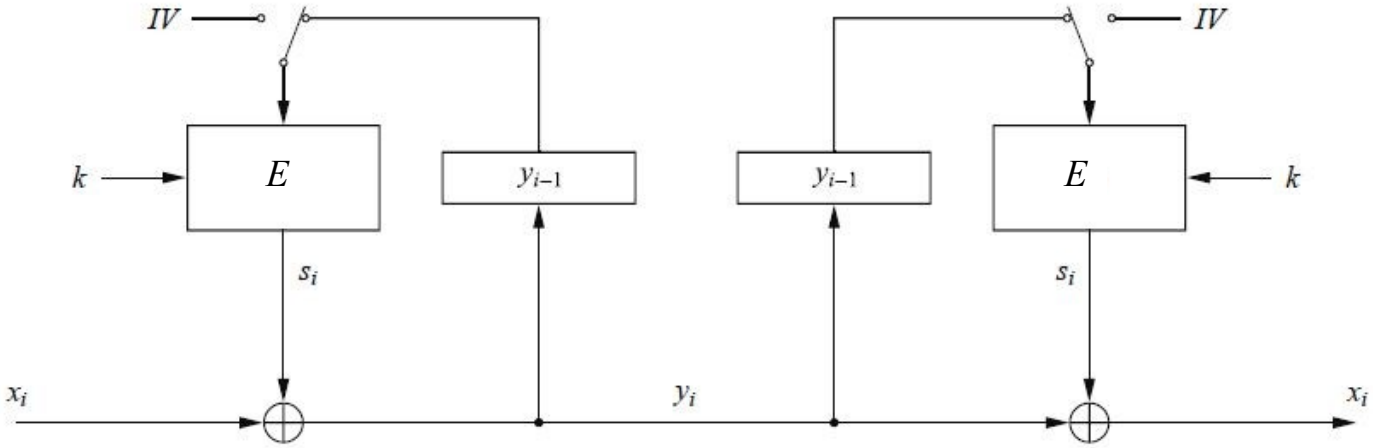
Output Feedback Mode (OFB) →

- OFB builds a stream cipher out of a block cipher.
- The key stream is generated block-wise.
- OFB is a synchronous stream cipher as the key stream is a function of *K* and *IV*, only.
  - → Precomputation of key stream is possible.
- The receiver does not use decryption.
- If |last pt block| < block_size, keystream bits are discarded.

Output Feedback Mode (OFB)

- IV should be a nonce ➔ OFB non-deterministic
- No error propagation
- OFB suffers from malleability

Cipher Feedback Mode (CFB) →



Definition 5.1.4 Cipher feedback mode (CFB)

Let $e()$ be a block cipher of block size b; let x_i and y_i be bit strings of length b; and IV be a nonce of length b.

Encryption (first block): $y_1 = e_k(IV) \oplus x_1$

Encryption (general block): $y_i = e_k(y_{i-1}) \oplus x_i, \quad i \geq 2$

Decryption (first block): $x_1 = e_k(IV) \oplus y_1$

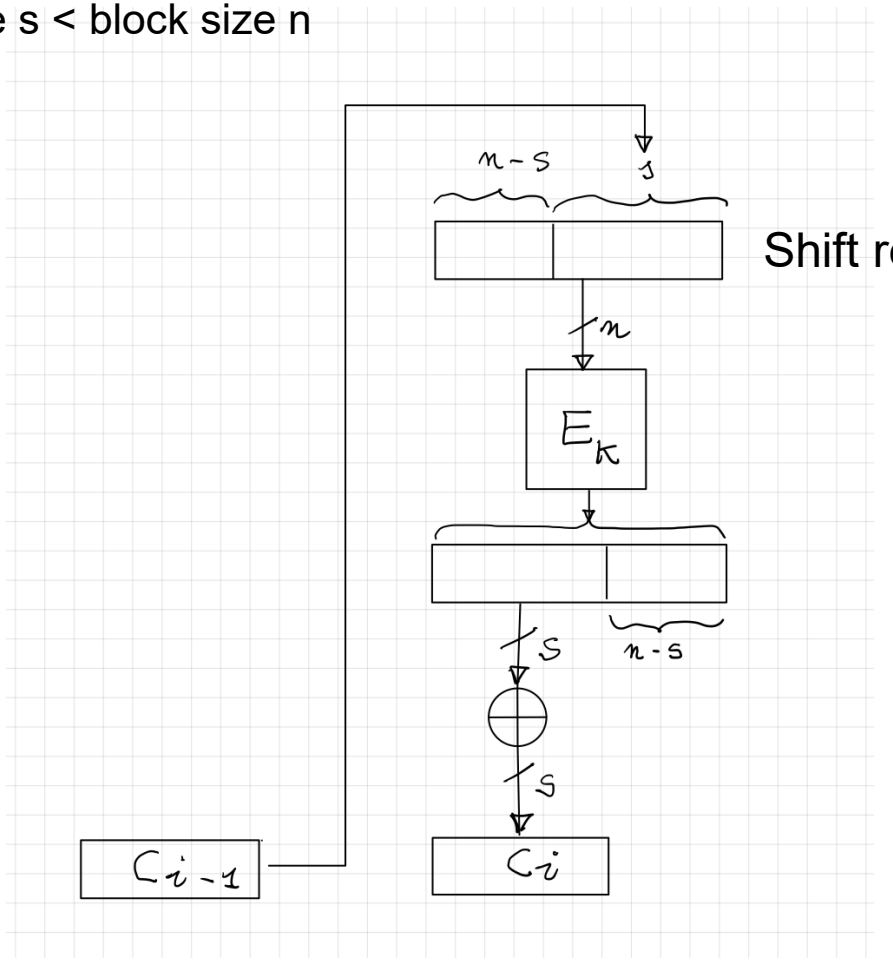
Decryption (general block): $x_i = e_k(y_{i-1}) \oplus y_i, \quad i \geq 2$

Cipher Feedback Mode (CFB) →

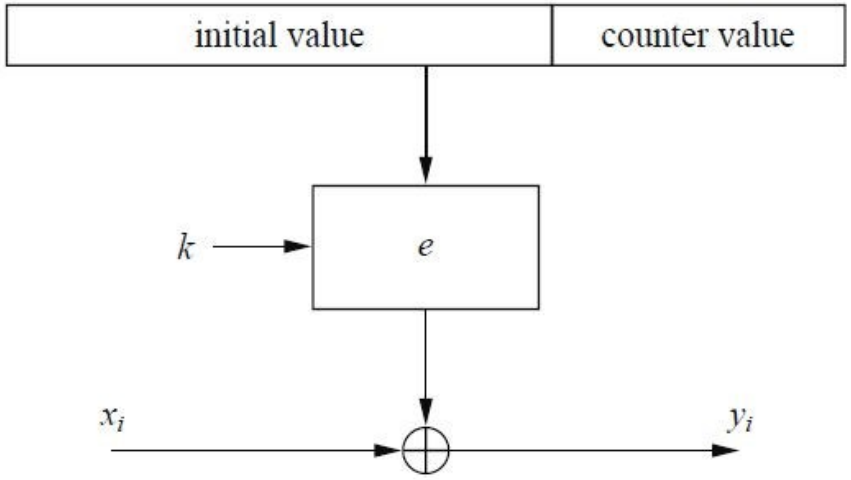
- OFB builds a stream cipher out of a block cipher.
- CFB is an asynchronous stream cipher as the key stream is also a function of the CT.
- Key stream is generated block-wise.
- IV is a nonce and makes CFB non-deterministic.
- Enc is sequential, Dec may be parallelized.
- CFB may operate on pt/ct smaller than a block
  - Sizeof(pt/ct) = $s \leq n$ (cipher block size)

Cipher Feedback Mode (CFB)

Pt/ct character size s < block size n



Counter Mode (CTR)



Definition 5.1.5 Counter mode (CTR)

Let $e()$ be a block cipher of block size b, and let x_i and y_i be bit strings of length b. The concatenation of the initialization value IV and the counter CTR_i is denoted by $(IV||CTR_i)$ and is a bit string of length b.

Encryption: $y_i = e_k(IV||CTR_i) \oplus x_i, \quad i \geq 1$

Decryption: $x_i = e_k(IV||CTR_i) \oplus y_i, \quad i \geq 1$

Counter Mode (CTR) →

- CTR prevents two-time pad (keystream reuse).
- CTR can be parallelized.
- Counter can be a regular counter or a more complex functions, e.g., LFSR.
- Ciphertext expansion is just one block.
  - Output $y_0, y_1, \ldots, y_t$ with $y_0 = (IV|ctr_0)$ being the *expansion block.*
  - $IV|ctr_0$ does not have to be kept secret.
    - Can be transmitted together with ct y_i.

CTR is CPA-secure

- A block cipher is a good approximation of a PRP (PRF), so the sequence $E_k(iv|ctr_0+1)$, …, $E_k(iv|ctr_0+t)$ is pseudorandom.
  - Two-time pad may occur when $(iv|ctr_0+i)$ wraps around ➔ limit to the maximum number of messages you can encrypt
  - Two-time pad may occur when $(iv|ctr_0+i) = (iv'|ctr_0'+ j)$ but the probability of this event is exponentially small.

CTR – 2TP

- 2-time pad
  - Let CT1 = PT1 xor S and CT2 = PT2 xor S.
  - Assume a successful KPA against one pt, e.g., against (CT1, PT1), then
  - The adversary can
    - determine S = PT1 xor CT1 and then
    - decrypt PT2 = CT2 xor S = CT2 xor (PT1 xor CT1)

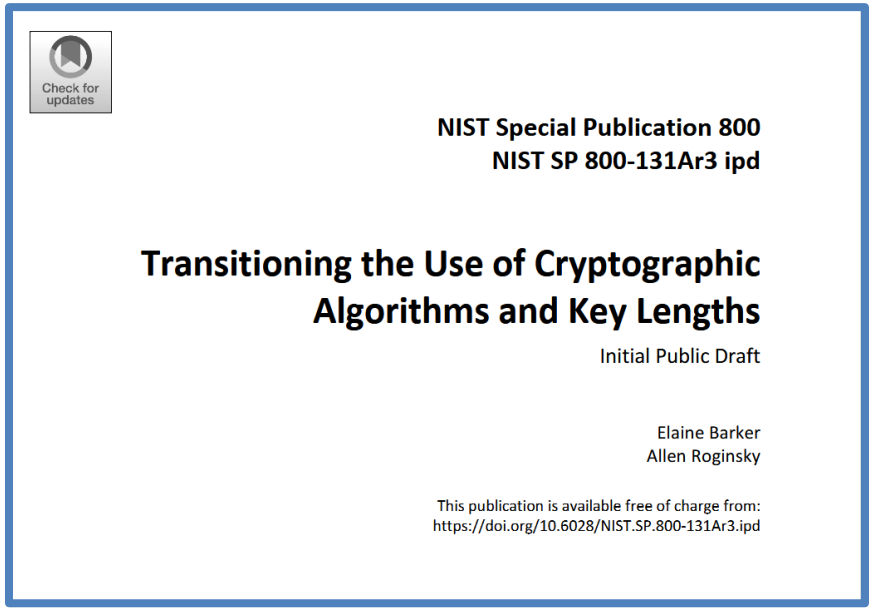
CTR – maximum traffic allowed

- AES, n = 128-bit
  - Assume | IV | = 96-bit and |cnt | = 32-bit
  - AES-CTR can encrypt 2^{32} pt's, each one bein 128-bit (16 bytes)
  - Traffic = $2^{32} \times 2^4$ = 64 Gbytes

CFB, OFB, and CTR are acceptable

Table 2. Approval status of the block cipher modes of operation for AES encryption and decryption

Publication	Mode	Status
SP 800-38A	ECB	Disallowed for data encryption Legacy use for decryption
	CBC	Acceptable
	CFB	Acceptable
	CTR	Acceptable
	OFB	Acceptable



Block Ciphers | Encryption modes

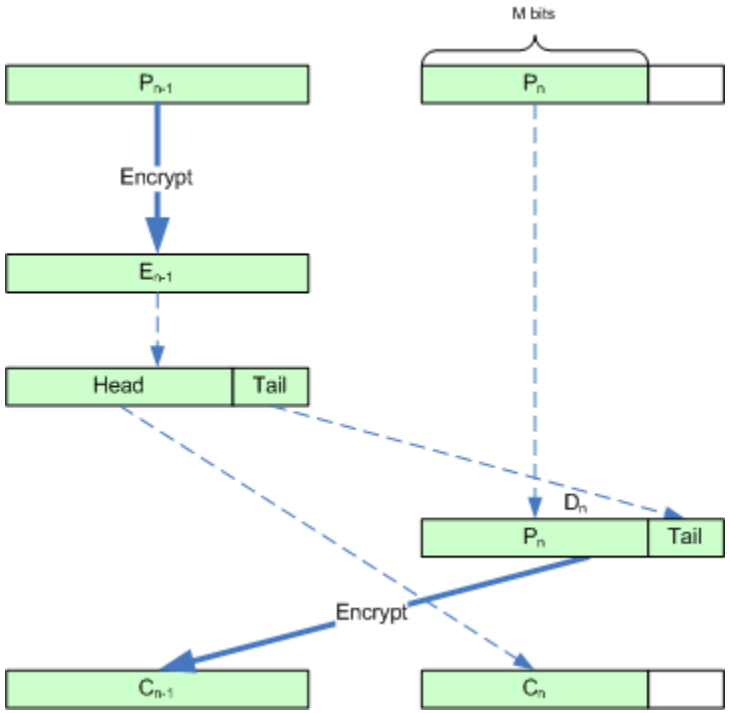
CIPHERTEXT STEALING (CTS)

Ciphertext Stealing (CTS) mode →

- CTS allows encrypting PT that is not evenly divisible into blocks without resulting in any ciphertext expansion
  - sizeof(ciphertext) = sizeof(plaintext)
- CTS operates on the last two blocks
- Intuition: a portion of the 2nd-last CT block *is stolen* to pad the last PT block

Ciphertext stealing (CTS)

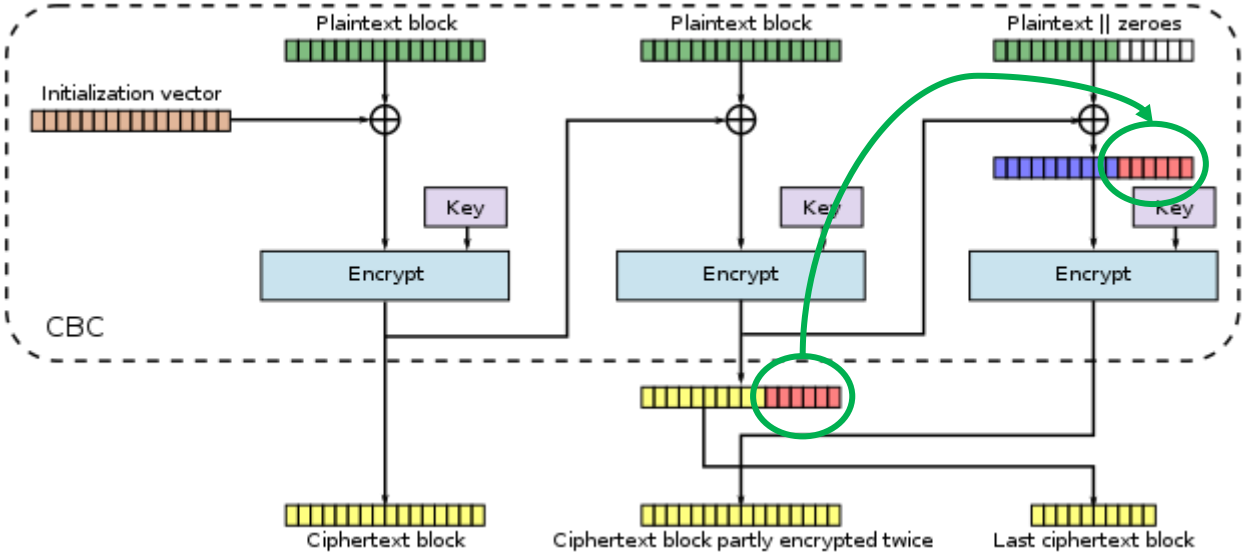
- CTS + ECB mode



Picture taken from Wikipedia

Ciphertext stealing (CTS) →

- CTS + CBC mode



[Picture taken from Wikipedia]

Block ciphers | Encryption modes

XTS-AES MODE

XTS-AES Mode for Block-Oriented Storage Devices

- IEEE Std 1619-2007
  - Standard describes an encryption mode for data stored in sector-based devices where the threat model includes possible access to stored data by the adversary
  - Has received widespread industry support
- Approved as an additional block cipher mode of operation by NIST in 2010

Implementations

- Software Vastly used
  - BestCrypt, dm-crypt, FreeOTFE, TrueCrypt, DiskCryptor, FreeBSD e OpenBSD+
  - Nativo in Mac OS X Lion (nel FileVault)
  - BitLocker di Windows 10
- Hardware
- SPYRUS Hydra PC Digital Attaché
- Kingston DataTraveler 5000

XTS-AES Assumptions

- Hard disk organized in (tracks and) sectors
- A sector is the read/write unit
- Sector size is typically 512 byte
- A sector may be divided up in blocks
- Encryption
  - Use all the space
  - Depends only on a) Cleartext, b) Encryption key, c) Sector number and block number

XTS-AES – Requirements (→)

Now we also worry about "data in use" while we are working on it

- The requirements for encrypting stored data, also referred to as "data at rest", differ somewhat from those for transmitted data (data in transit)

1. The ciphertext is freely available for an attacker Assume attacker can read the HD.
2. The data layout is not changed on the storage medium and in transit. Size of enciphered data must be equal to the size of clear data
3. Data are accessed in fixed sized blocks, independently from each other

Now I need sector 1, then sector 23. Then in sector I need block 1, the 7. Sequentialization is not valuable

XTS-AES – Requirements (→)

- Encryption is performed in 16-byte blocks, independently from each other
- There are no other metadata used, except the location of the data blocks within the whole data set*
- The same plaintext is encrypted to different ciphertexts at different locations, but always to the same ciphertext when written to the same location again:

(No IV, for ex.)

If PT written in two diff. sectors, two CT must be diff. - No traffic analysis

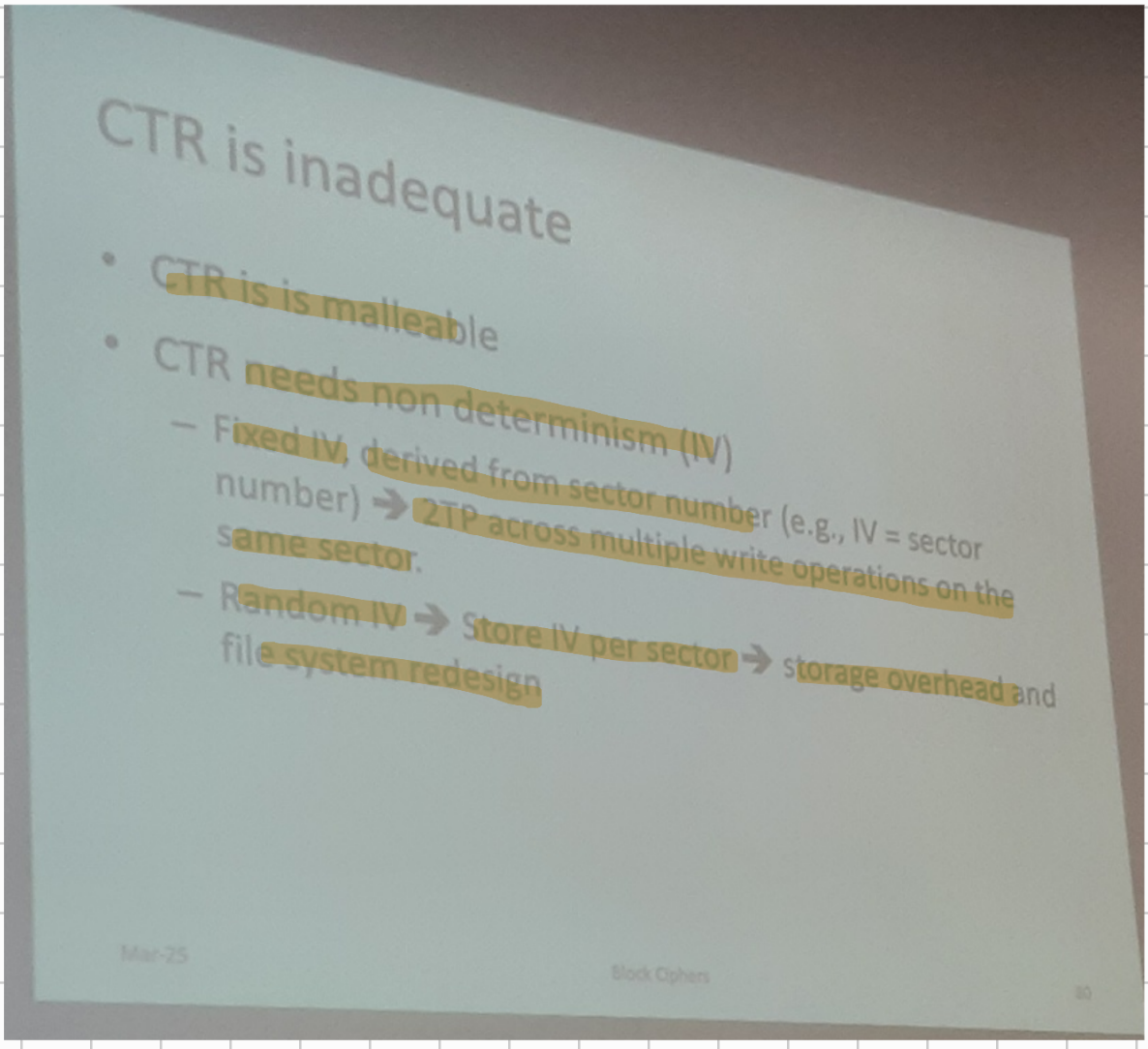
* You don't want to modify the FS on top: ex. no inserting IV in file descriptor

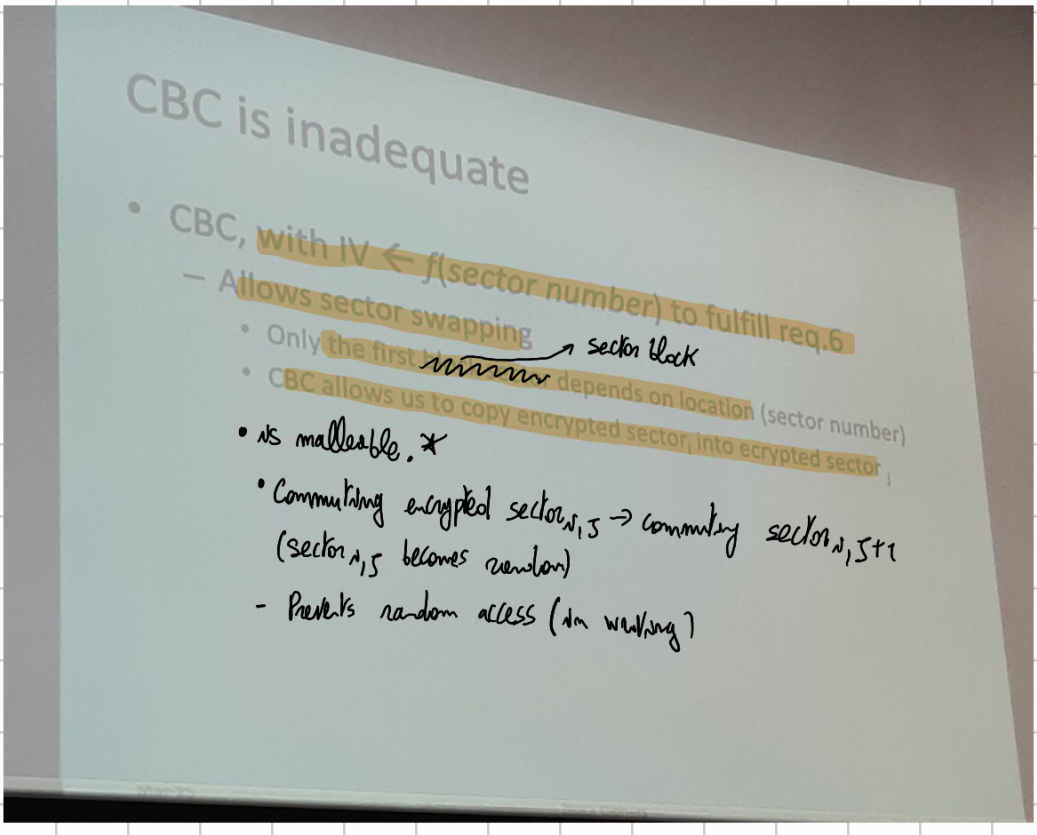
XTS-AES – Requirements (↓)

- A standard conformant device can be constructed for decryption of data encrypted by another standard conformant device : Solution must be standard for portability

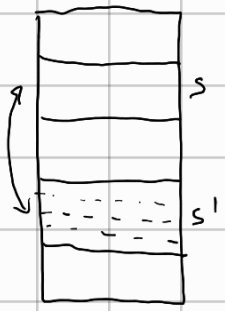
CTR and CB are inadequate

- ECB
  - Pattern analysis
- CBC (with IV = location)
  - Only the first block depends on location
  - Malleable
- CTR XTS does not guarantee integrity, but CTR is malleable in an easy way
  - It is malleable
  - It needs a separate initial state for each sector ➔ waste of space


CTR is inadequate
• CTR is is malleable
• CTR needs non determinism (IV)
– Fixed IV, derived from sector number (e.g., IV = sector number) → 2TP across multiple write operations on the same sector.
– Random IV → Store IV per sector → storage overhead and file system redesign
Mar-25
Block Ciphers


CBC is inadequate
• CBC, with IV ← f(sector number) to fulfill req.6
– Allows sector swapping
• Only the first block depends on location (sector number)
• CBC allows us to copy encrypted sector, into ecrypted sector
• is malleable. *
• Commuting encrypted sector$_{n,5}$ → commuting sector$_{n,5+1}$ (sector$_{n,5}$ becomes random)
- Prevents random access (dm warking)

Suppose IV depends on sector. Sector swapping means I can take sector S and swap it with S':



Sector is composed of several blocks.

$C_j = E_K(P_j \oplus C_{j-1})$, $C_0 = IV$

So the very first block: $C_1 = E_K(P_1 \oplus IV)$

So the very first block depends on the sector, but from C_2 on $C_j = E_K(P_j \oplus C_{j-1})$

So if I obtain P_1, $P_1 = Dec_K(C_1) \oplus IV$

then $P_2 = Dec_K(C_2) \oplus C_1$

⋮

So if I copy blocks up, I will use different init. vector, so I get $P_1' \neq P_1$. But since P_2 will be able to decrypt correctly. Because P_2 depends only on cyphertexts.

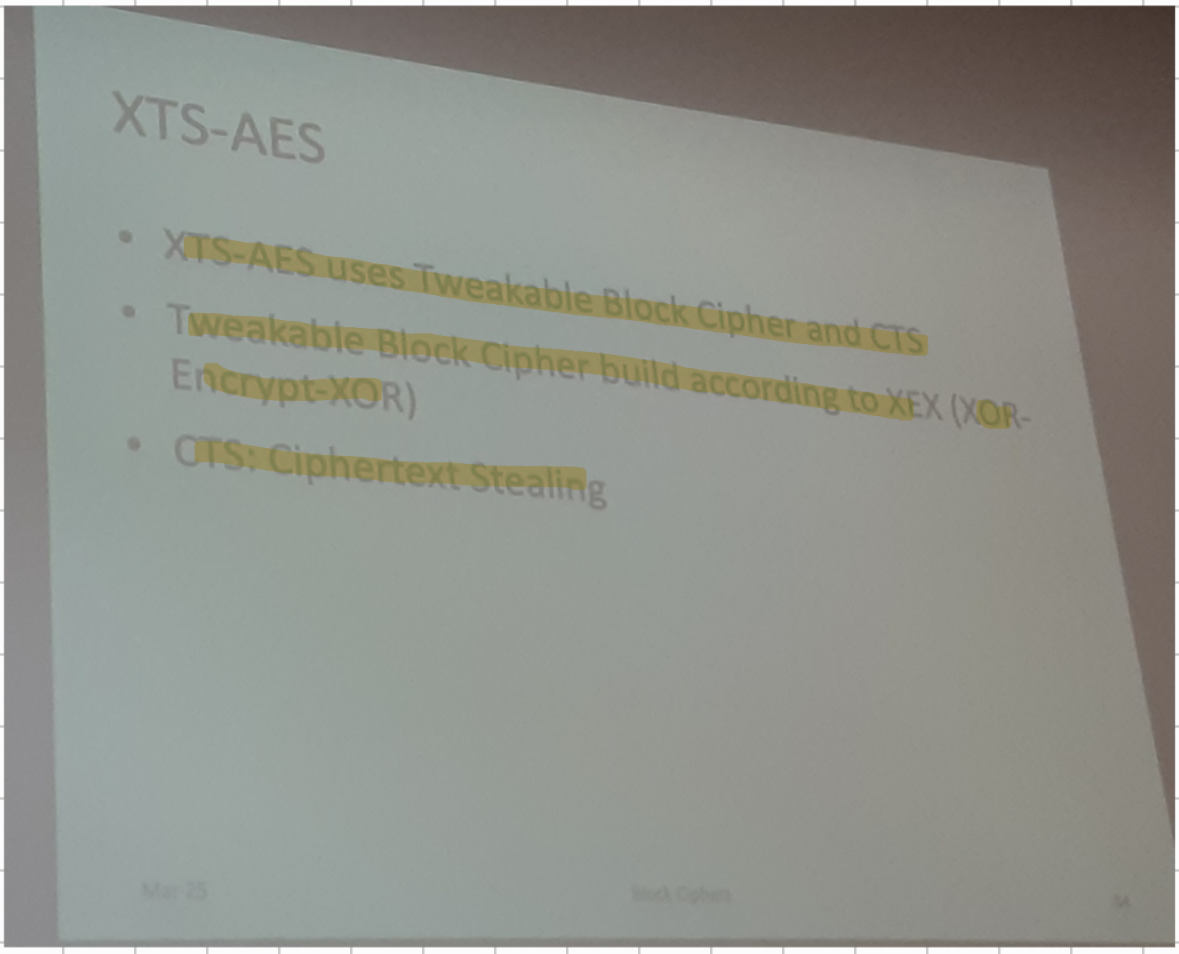
* For any reason, I know that a specific block contains a very important env. variable that I want to change in a block. Block 5 for ex.

If you change bits in preceding CT:

$P_5 = D_K(C_5) \oplus C_{5-1}$

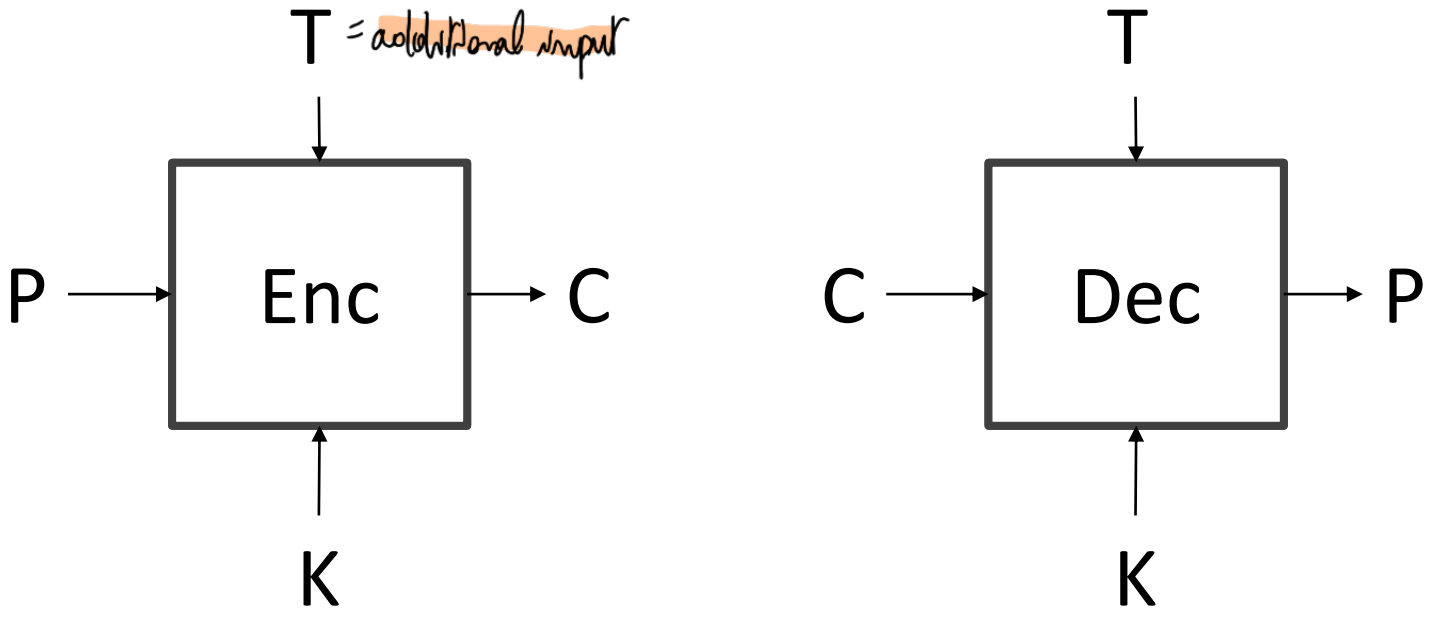
↳ Those have been changed

- ECB leaks patterns
- GCM adds authentication but expands data (keys and nonce), unsuitable for in place encryption



You can change the tweak block by block if you like

Tweakable Block Ciphers



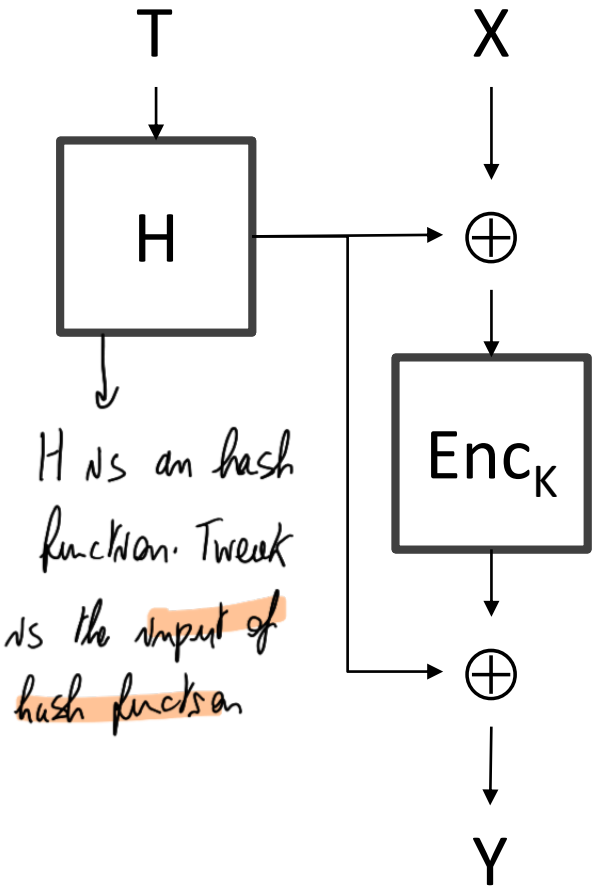
- T is public = tweak
- K provides *security* while T provides *variability*

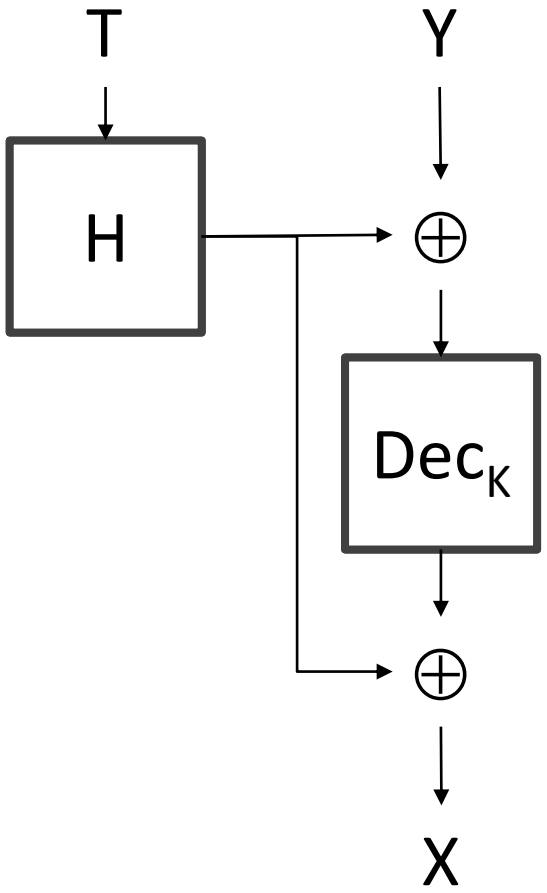
Block cipher implements a family of permutations. K selects one particular permutation. Here K and T both do same.

Tweakable Block Ciphers

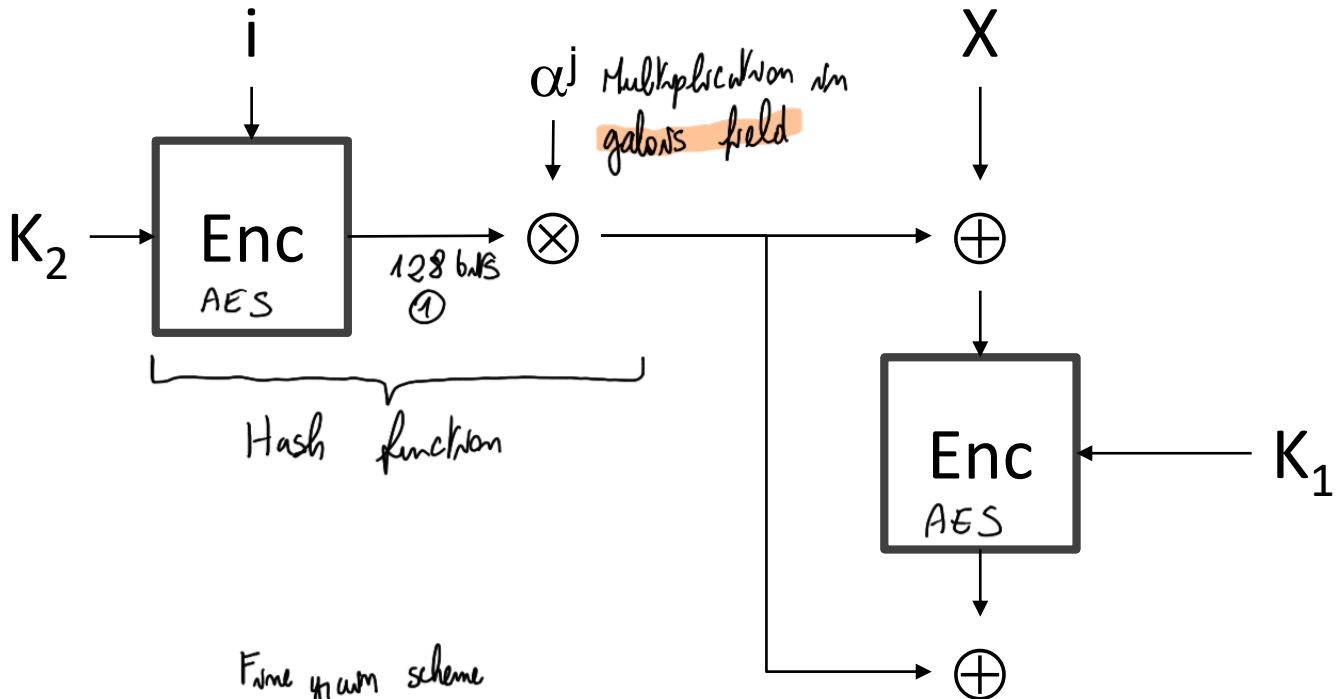
- Intuition: use a different key for each sector
  - $y_t = Enc_{k_t}(x_t)$
- Tweakable cipher: Enc(k, t, x)
- Trivial example
  - $k_t = Enc_K(t)$, with *K* master key
  - Ciphertext: $y_t = Enc_{k_t}(x_t)$
  - Encryption of n blocks sector requires 2n encryptions

Tweakable Block Ciphers





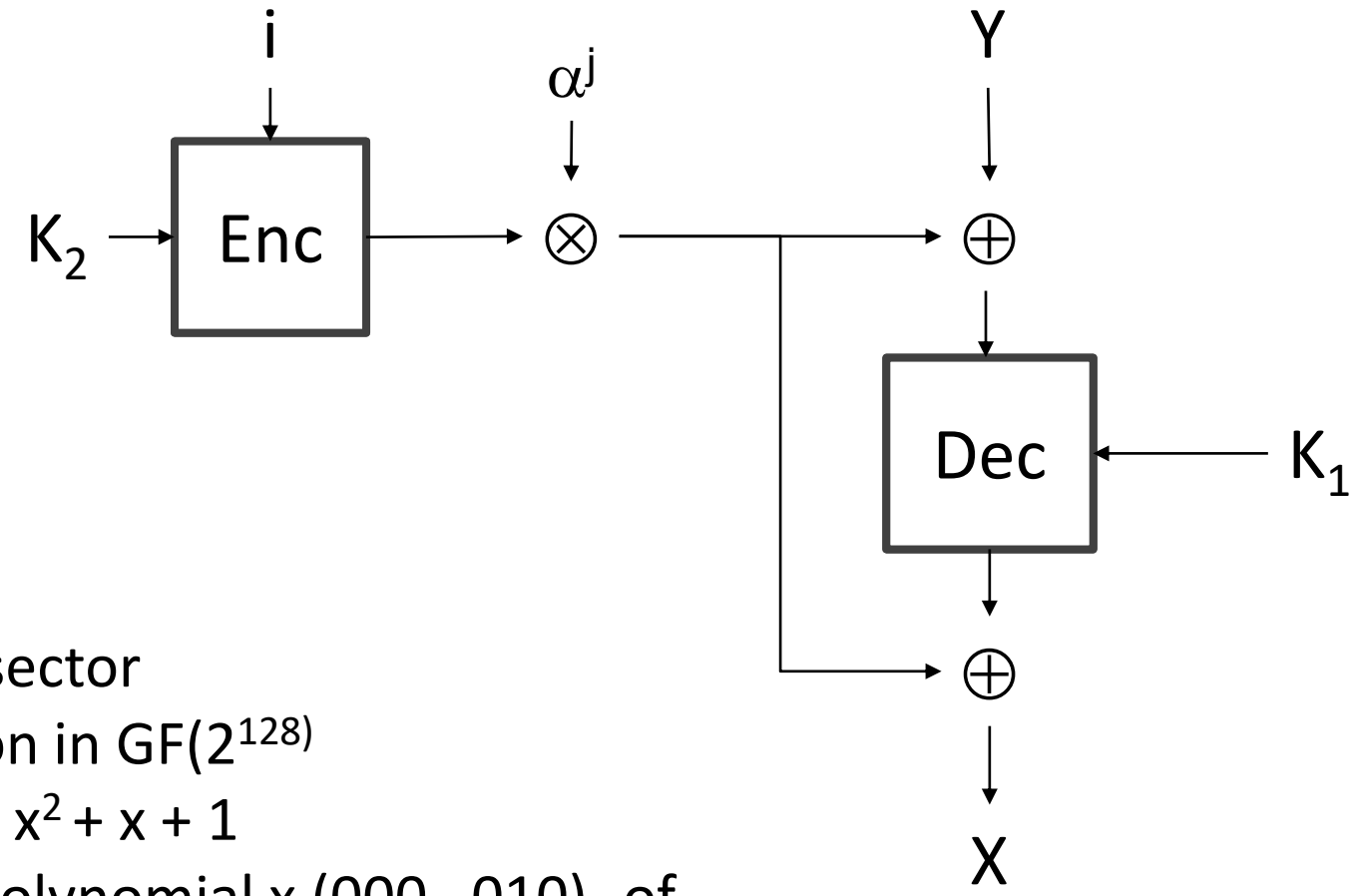
XTS-AES: block encryption



- Enc/Dec = AES
- Tweak = i, j; i: sector, j: block in the sector
- $\otimes$ multiplication in $GF(2^{128})$ mod $x^{128} + x^7 + x^2 + x + 1$
- α = primitive polynomial x $(000\ldots010)_2$ of $GF(2^{128})$
- An n blocks sector requires n+1 encryptions

I am using cypher in ECB mode, but different blocks and sections will give diff. results.

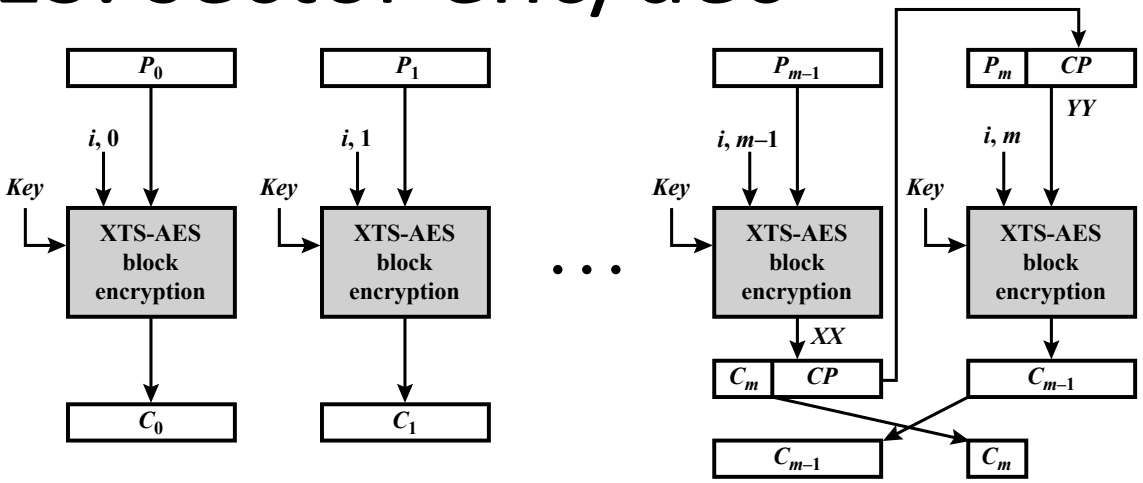
XTS-AES: block decryption



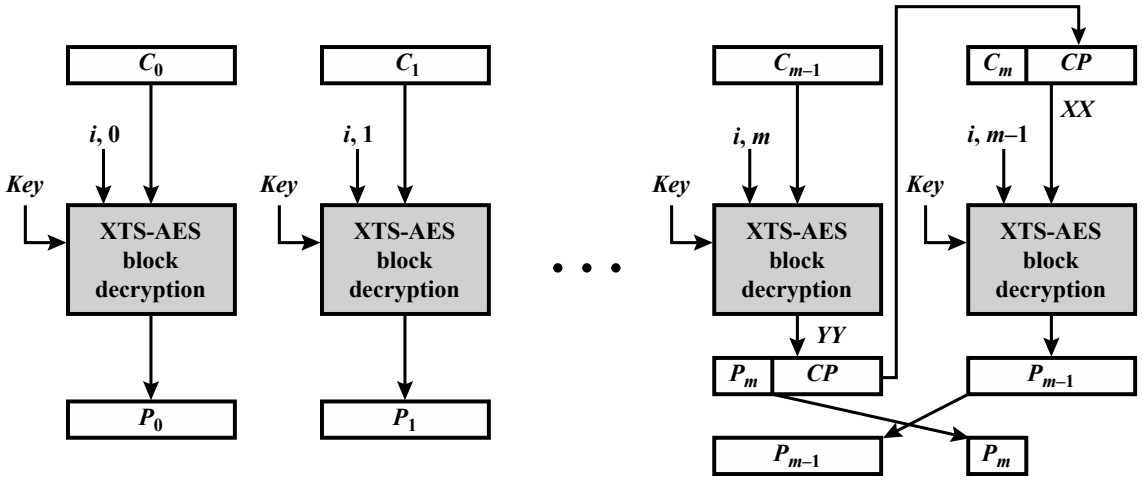
- Enc/Dec = AES
- Tweak = i, j
- i: sector
- j: block in the sector
- ⊗ multiplication in GF(2^{128}) mod $x^{128} + x^7 + x^2 + x + 1$
- α = primitive polynomial x $(000\ldots010)_2$ of GF(2^{128})

XTS-AES: sector enc/dec

Output processed reverse
to CTS mode to
avoid CT expansion



(a) Encryption



(b) Decryption