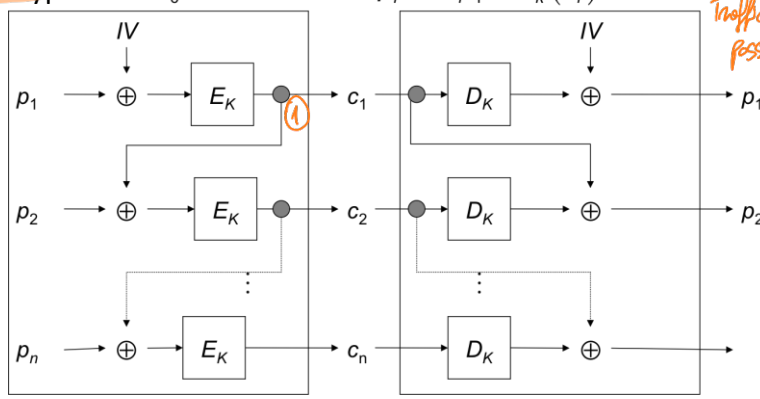
Cipher block chaining (CBC)

① can be seen as random from a limited adversary (like 1TP) If I transmit same message but use different IV, done. Traffic analysis not possible. And I cannot move blocks.

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

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



Mar-25 Block Ciphers 46

46

still under the assumption that message is an n-key multiple of the block size

CBC – properties (→)

- CBC mode is CPA-secure
- CBC-Enc is *randomized* by using IV (nonce) [IV has to change]
  - Identical ciphertext results from the same PT under the same key and IV (IV diff., traffic analysis not possible)
- Chaining dependencies: c_i depends on p_i and the preceding PT block
- CT-block reordering affects decryption (decryption fails) → same for substitution
- Cyphertext expansion is just one block

Mar-25 Block Ciphers 47

47

CBC – properties

- IV can be sent in the clear but its integrity must be guaranteed
- CBC suffers from Error propagation
  - Bit errors in c_i affect p_i and p_{i+1} (*error propagation*)
- Only CBC-dec can be parallelized

C_3' is received instead of C_3. $C_3' \neq C_3$. What is the effect?

$P_i = D_K(C_i) \oplus C_{i-1}$ (CORRUPTED; RECEIVED CORRECTLY)

$P_{i+1} = D_K(C_{i+1}) \oplus C_i$ (CORRUPTED)

It has effects on 2 blocks

48

If cypher is done correctly I have avalanche property: if I commute just one bit, 50% of the output changes in a random way (should look enough like a random variable) (every bit of the output changes with 50% of prob.). But in this case plaintext changes

CBC – block attack

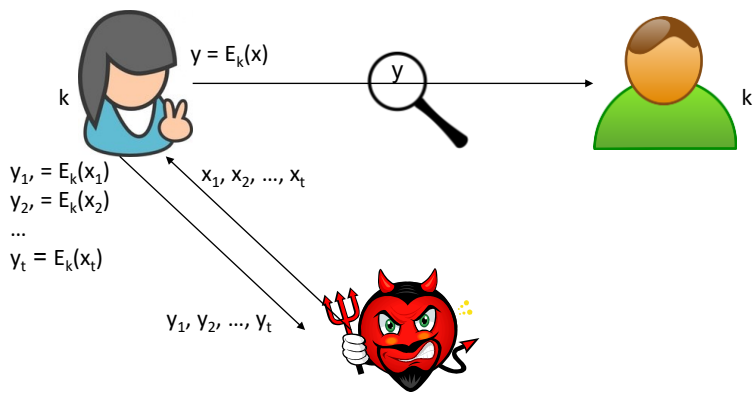
- If Bank A chooses a random IV for each wire transfer the attack will not work
- However, if Lou Cipher substitutes blocks #5–10 and #13, bank B would decrypt *account number* and *deposit amount* to random numbers => this is highly undesirable!
- Encryption itself is not sufficient, we need additional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

49

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 $y = E_k(m)$
- CPA-security guarantees that the adversary cannot learn anything about x (any new message)
- The encryption scheme must be randomized
  - ↳ informally: if I encrypt x twice I get two diff. ct (we get this through IV)

CPA model



Block Ciphers

PADDING

52

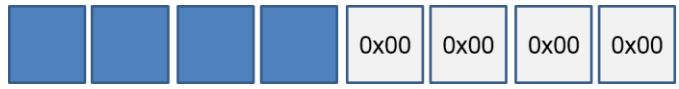
Padding

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

53

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

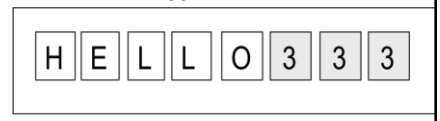
Consider an 8 bytes block

If PT len is NOT a block multiple

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

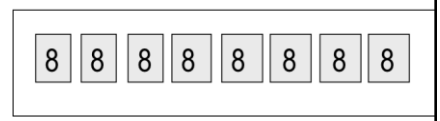
Example: b = 3 then append 0x030303

Block



If PT len is a block multiple
Padding = block
Fill each padding block by 8

Entire block of padding



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
    - i.e., if b = 3, append 0x030303

PKCS #5: decryption

- After decryption, say the final byte has value *b*
  - If b = 0 or b > L, return "error" b should be $\in [1, L]$
  - 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 #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)

→ generalization, but algorithm is the same

Block Ciphers | Padding

PADDING ORACLE ATTACK

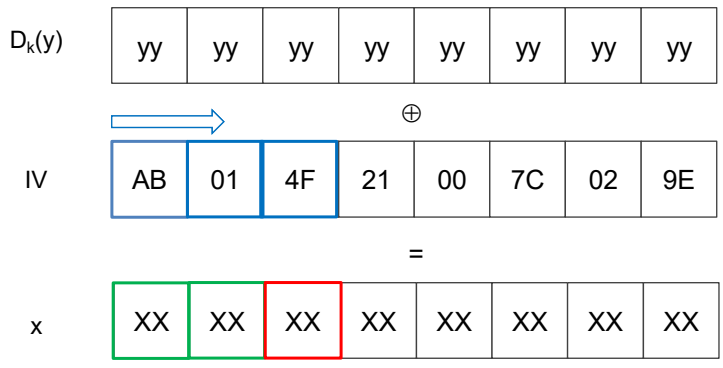
Padding Oracle Attack (CCA)

- 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

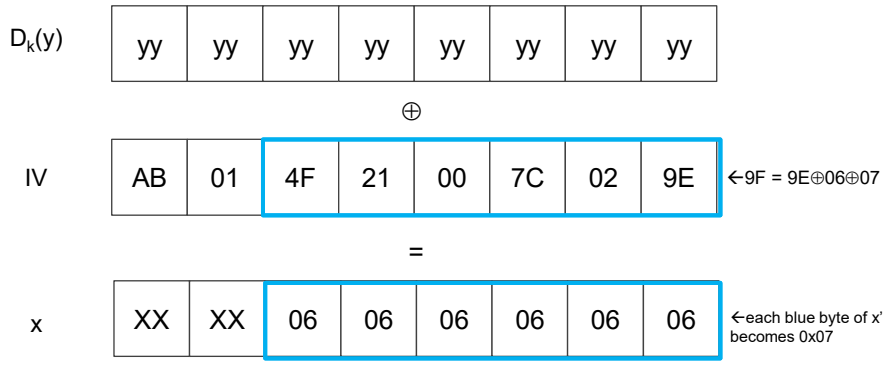
The attack – step 1 – determine padding lenght



Mar-25 Block Ciphers 62

62

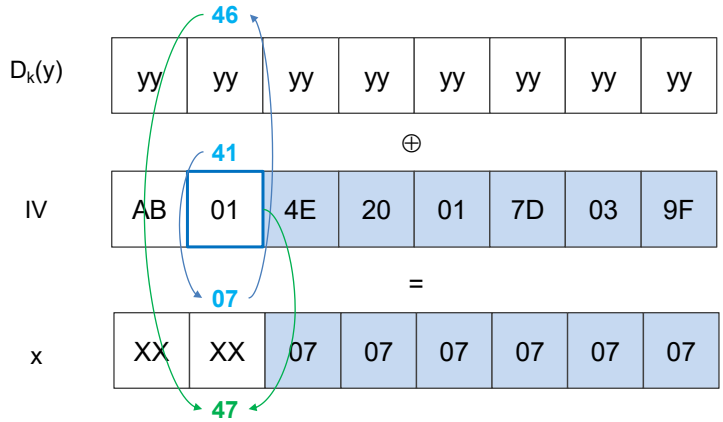
The attack – step 2a – determine pt



Mar-25 Block Ciphers 63

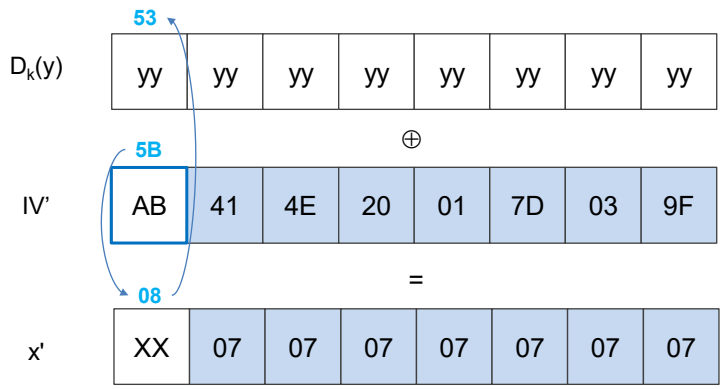
63

The attack – step 2b – determine pt



64

The attack – step 2c – determine pt



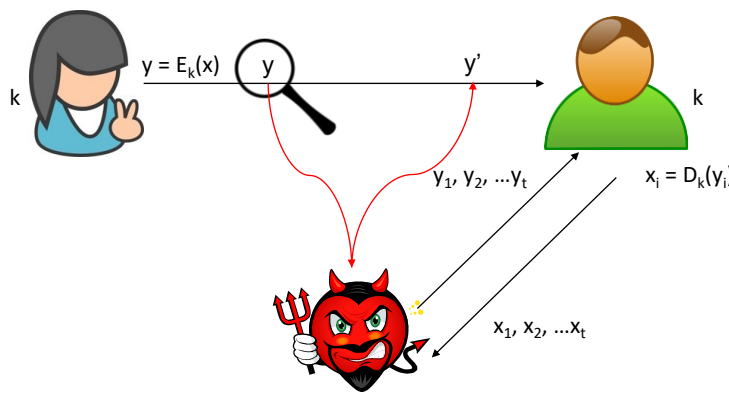
65

Attack complexity

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

66

CCA model



67

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

68

CCA-security

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

69

Enc. modes are modes of using block cyphers. The first two were ways to use BC when you have message longer than block

Block Ciphers: How to transform a block cipher into a stream cipher

MORE ENCRYPTION MODES: OFB, CFB, CTR

Those are to implement stream cyphers and for cryptographically secure RandoR. but yes.

70

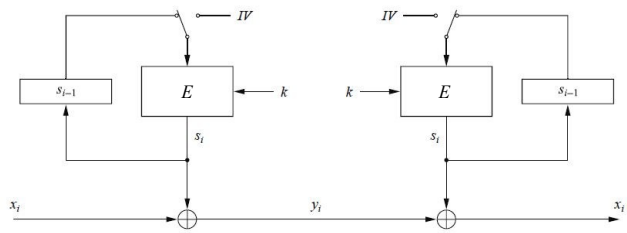
Block cipher vs stream cipher

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

71

Output Feedback Mode (OFB) →

PSEUDO RANDOM GEN BUILT OUT OF BLOCK CYPHER
Secret key to be shared



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* $y_i = s_i \oplus x_i$, $i \geq 2$

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* $x_i = s_i \oplus y_i$, $i \geq 2$

First block switch is on IV!

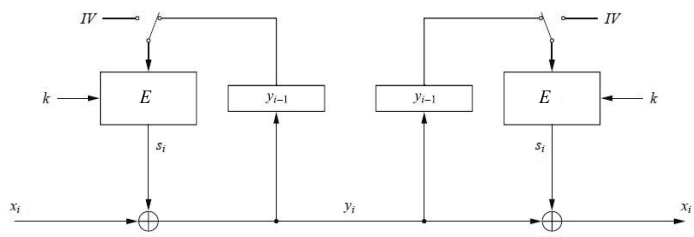
Output Feedback Mode (OFB) →

- OFB builds a stream cipher out of a block cipher
- The key stream is generated block-wise
- OFB is a synchronous stream cipher, i.e., key stream is a function of *K* and *IV*, only
  - → precomputation of key stream is possible (u can precompute a prefix up to max len)
- The receiver does not use decryption
- If |last pt block| < block, 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 nondeterministic
- Enc is sequential, Dec may be parallelized
- CFB may operate on pt/ct smaller than a block
  - Sizeof(pt/ct) = s ≤ n (cipher block size)

Here no precomputation

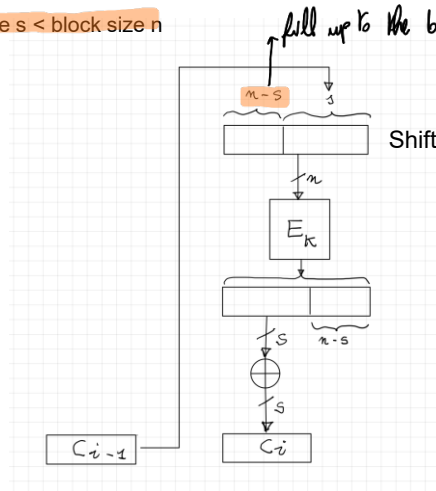
↳ in previous schems you have to work block wise

76

Cipher Feedback Mode (CFB)

Pt/ct character size s < block size n

fill up to the block and encrypt



77

Understanding the Shift Register in CFB Mode

In **CFB mode**, the encryption process does not directly encrypt the plaintext. Instead, the cipher operates on a **shift register**, which is an n -bit value (usually the block size of the cipher, e.g., 128 bits for AES). The shift register **evolves over time** as encryption progresses.

Step 1: Initialization

- The shift register is initially set to the **Initialization Vector (IV)**.
- The IV is encrypted using the block cipher to produce an **output block**.

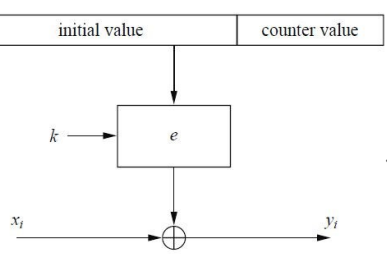
Step 2: Encryption Process

For each segment P_i of length s, the following happens:

1. **Encrypt the shift register** (using the block cipher).
    - Take the current shift register (an n-bit value).
    - Encrypt it using the block cipher to get an **intermediate output block** of size n.
    - This step **does not involve plaintext yet**; it's just encryption of the shift register.
2. **Extract the first s bits** of the encrypted shift register.
    - Take the **most significant s bits** of the encrypted output.
3. **XOR with plaintext to generate ciphertext**.
    - Compute $C_i = P_i \oplus \text{MSB}_s(\text{Enc}(\text{Shift Register}))$.
    - This produces an encrypted segment of length s.
4. **Update the shift register**.
    - The **shift register is updated** by:
        - **Dropping the leftmost s bits** (the oldest part).
        - **Appending the ciphertext segment C_i to the rightmost s bits.**
    - This new value becomes the shift register for the next iteration.

Counter Mode (CTR)

Of course we assume block ciphers are secure



Ad hoc CSPRBG

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

78

Counter Mode (CTR) →

IV

- CTR prevents two-time pad (keystream reuse)
- CTR can be parallelized and precomputed
- Counter can be a regular counter or a more complex functions, e.g., LFSR
- Ciphertext expansion is just one block → To communicate, the receiver must know IV|ctr0
  - 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

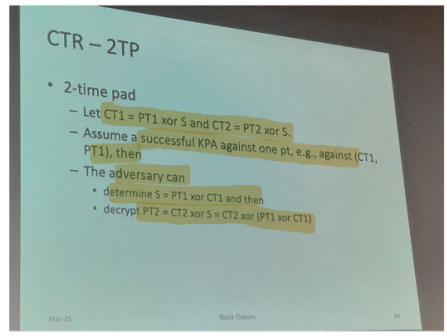
79

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

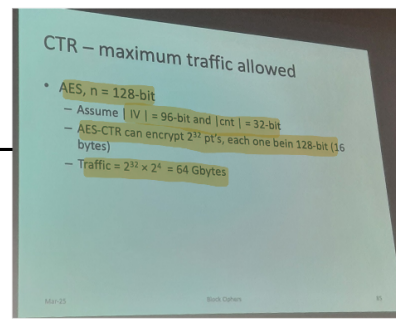
Mar-25 Block Ciphers 80

80



Block Ciphers: How to avoid ciphertext expansion

MORE ENCRYPTION MODES: CTS



Mar-25 81

81

When you ran out of IV, distribute a new Key.

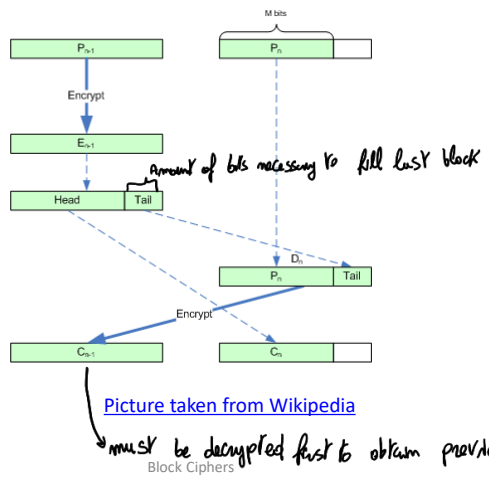
We have ct expansion because of padding. In some cases for embedded applications, in which maybe one component is on batteries and need to save energy, thus might be bad

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

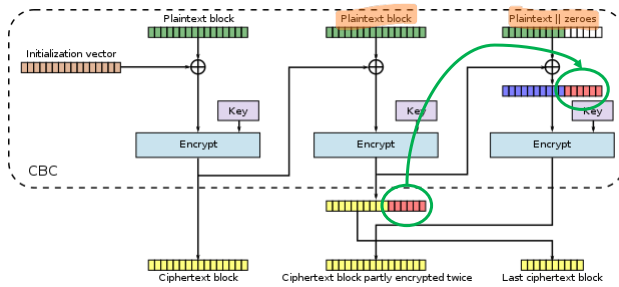


Picture taken from Wikipedia

must be decrypted first to obtain previous block

Ciphertext stealing (CTS) →

- CTS + CBC mode



Picture taken from Wikipedia

Error propagation will be present of course. And things will be a bit more complicated in implementation, especially HW.

84

XTS-AES MODE

85

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

86

Implementations

- Software
  - 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

87

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

88

XTS-AES – Requirements (→)

- The requirements for encrypting stored data, also referred to as “data at rest”, differ somewhat from those for transmitted data
- The ciphertext is freely available for an attacker
- The data layout is not changed on the storage medium and in transit
- Data are accessed in fixed sized blocks, independently from each other

89

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

90

XTS-AES – Requirements (↓)

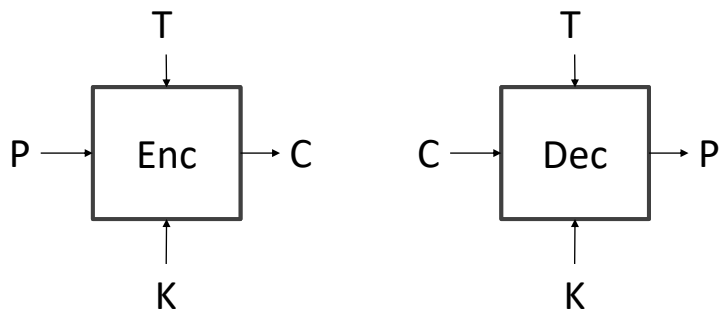
- A standard conformant device can be constructed for decryption of data encrypted by another standard conformant device

91

CTR and CB are inadequate

- ECB
  - Pattern analysis
- CBC (with IV = location)
  - Only the first block depends on location
  - Malleable
- CTR
  - It is malleable
  - It needs a separate initial state for each sector ➔ waste of space

Tweakable Block Ciphers



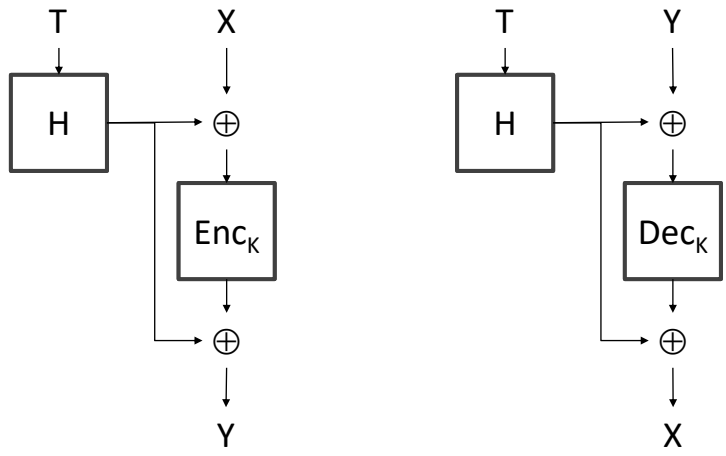
- T is public
- K provides *security* while T provides *variability*

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

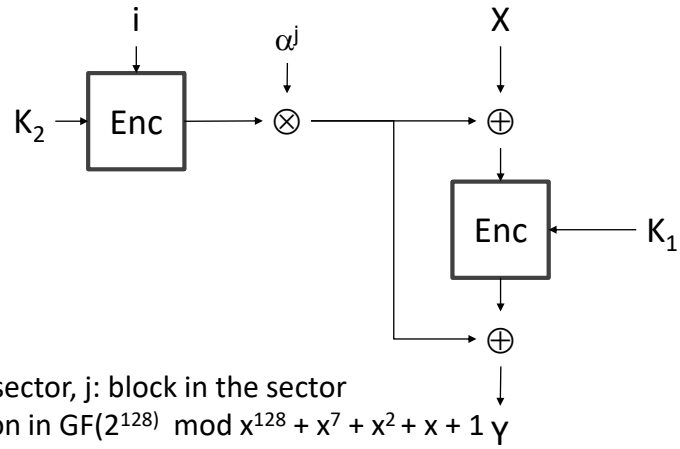
94

Tweakable Block Ciphers



95

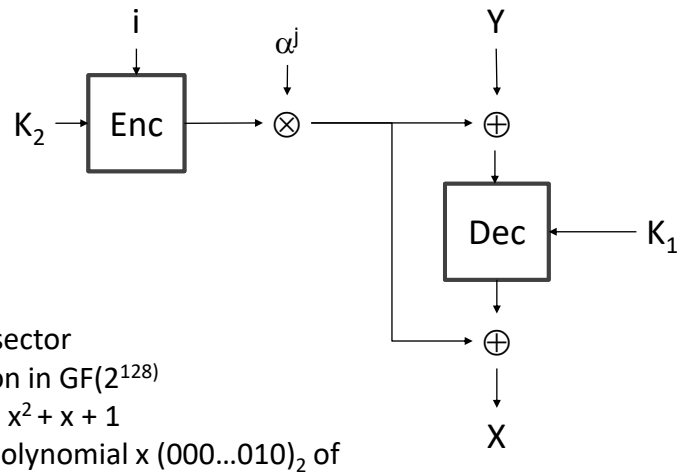
XTS-AES: block encryption



- 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

96

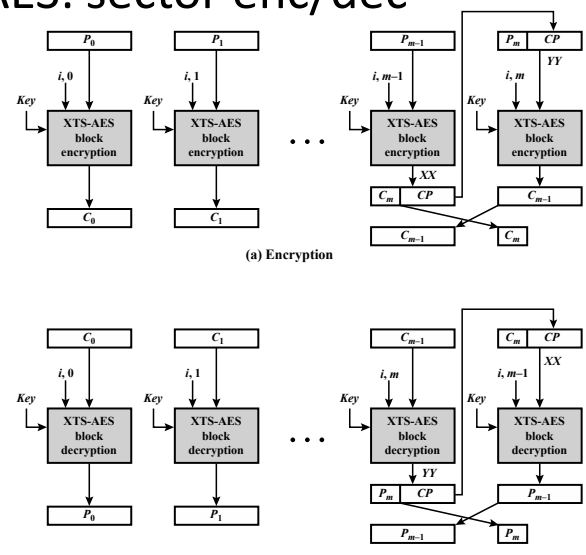
XTS-AES: block decryption



- 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 $GF(2^{128})$

97

XTS-AES: sector enc/dec



98

99