

Block Ciphers II

Martin Stanek

Department of Computer Science
Comenius University
`stanek@dcs.fmph.uniba.sk`

Cryptology 1 (2022/23)

Content

Confidentiality modes – ECB, CBC, OFB, CFB, CTR

Padding

- Padding oracle attack

- Ciphertext stealing

Authenticated encryption – CCM, GCM

Other constructions

- Format-preserving encryption

- Encryption of block devices

Theoretical security of block ciphers

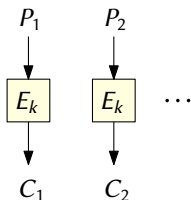
Modes of operation

- ▶ plaintext usually much longer than the block length
- ▶ modes of operation can provide:
 - ▶ confidentiality (and not authenticity) ... “traditional” modes
 - ▶ authenticity (and not confidentiality)
 - ▶ confidentiality & authenticity (authenticated encryption)
 - ▶ confidentiality for block-oriented storage devices (e.g. disks)
 - ▶ key wrapping
 - ▶ format-preserving encryption, ...
- ▶ varying requirements (speed, security properties, ability to parallelize, availability of RNG, etc.) \Rightarrow different modes

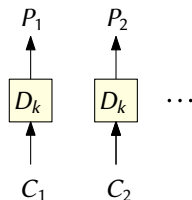
Confidentiality modes

- ▶ the most important confidentiality modes: ECB, CBC, OFB, CFB, CTR
- ▶ e.g. see NIST SP 800-38A: *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*
- ▶ None of these modes provide protection against accidental or adversarial modifications of ciphertext!
- ▶ however, the effect of ciphertext modification on resulting plaintext varies among modes

ECB (Electronic Codebook)



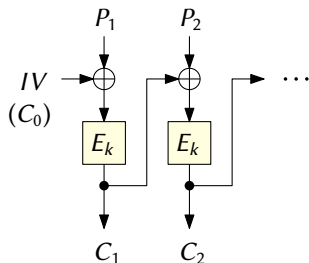
encrypt: $C_i = E_k(P_i)$



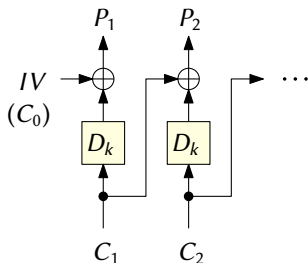
decrypt: $P_i = D_k(C_i)$

- ▶ the simplest mode
- ▶ data leaks: $C_i = C_j \Leftrightarrow P_i = P_j$
- ▶ easy to rearrange the ciphertexts blocks (permute, duplicate, ...)
- ▶ encryption and decryption trivially parallelizable
- ▶ easy to perform a seek (random access)
- ▶ bit changes do not propagate (single block affected)

CBC (Cipher Block Chaining) 1



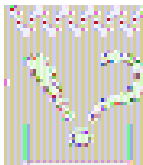
encrypt: $C_i = E_k(P_i \oplus C_{i-1})$



decrypt: $P_i = D_k(C_i) \oplus C_{i-1}$

- ▶ IV (initialization vector) – secrecy not required, usually appended as C_0
- ▶ popular mode (e.g. AES-128 CBC was mandatory in TLS 1.2, RFC 5246)
- ▶ parallelizable decryption but not encryption
- ▶ bit change in plaintext or IV propagates to the rest of the ciphertext
- ▶ bit change in the ciphertext affects only two plaintext blocks

Visual comparison of ECB and CBC (AES-128)



ECB



CBC

CBC 2

- ▶ “self-synchronizing” after losing a ciphertext block
- ▶ similarly to ECB, plaintext should be a multiple of block length
 - ▶ padding, ciphertext stealing
- ▶ IV should be unpredictable (e.g. $IV = E_k(\text{msg}_{\text{seq}})$, random, ...)
 - ▶ otherwise, in CPA scenario, an attacker gets an $E_k(\cdot)$ oracle
 - ▶ since $C_1 = E_k(IV \oplus P_1)$, predictable IV allows him/her to adjust P_1
 - ▶ the attacker with $E_k(\cdot)$ access can test any plaintext block

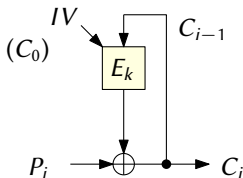
CBC 3

- ▶ data leak (birthday & two-time pad):

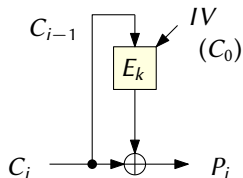
$$\begin{aligned}C_i = C_j &\Rightarrow E_k(P_i \oplus C_{i-1}) = E_k(P_j \oplus C_{j-1}) \\P_i \oplus P_j &= C_{i-1} \oplus C_{j-1}\end{aligned}$$

- ▶ Sweet32 attack (2016): ciphers with block length 64 bits and large amount of data encrypted using the same key (TLS, OpenVPN)
 - ▶ 64 bit block \Rightarrow collision expected after $\approx 2^{32}$ blocks (32 GiB)
- ▶ limit number of blocks encrypted with a single key

CFB (Cipher Feedback)



encrypt: $C_i = P_i \oplus E_k(C_{i-1})$



decrypt: $P_i = C_i \oplus E_k(C_{i-1})$

- ▶ parallelizable decryption but not encryption; D_k not needed
- ▶ bit change in plaintext or IV propagates to the rest of the ciphertext
- ▶ bit change in the ciphertext affects only two plaintext blocks
- ▶ self-synchronizing after full ciphertext block is lost
 - ▶ ciphertext block and its predecessor are needed to decrypt correctly
 - ▶ there is a variant for more granular losses

CFB 2

- ▶ plaintext length does not need to be a multiple of block length
- ▶ IV should be unique for each plaintext
- ▶ repeated IV :
 - ▶ two-time pad for the first blocks:

$$C_1 \oplus C'_1 = E_k(IV) \oplus P_1 \oplus E_k(IV) \oplus P'_1 = P_1 \oplus P'_1$$

- ▶ for constant IV we have an encryption oracle in CPA scenario;
2nd block (C_2):

$$C_2 = E_k(\underbrace{E_k(IV) \oplus P_1}_{C_1}) \oplus P_2$$

choosing $P'_1 = C_1 \oplus P_1 \oplus X$ and arbitrary P'_2 yields

$$C'_2 = E_k(E_k(IV) \oplus C_1 \oplus P_1 \oplus X) \oplus P'_2 = E_k(X) \oplus P'_2$$

thus $E_k(X) = C'_2 \oplus P'_2$

CFB8 variant of CFB mode and Zerologon

- ▶ Zerologon (CVE-2020-1472, Tom Tervoort) compromising domain admin in AD
- ▶ problems with cryptography in Netlogon protocol
- ▶ AES-CFB8
- ▶ CFB8 mode (P_i and C_i denote bytes):

$$C_1 = E_k(IV[0 \dots 15])[0] \oplus P_1$$

$$C_2 = E_k(IV[1 \dots 15]C_1)[0] \oplus P_2$$

$$C_3 = E_k(IV[2 \dots 15]C_1C_2)[0] \oplus P_3$$

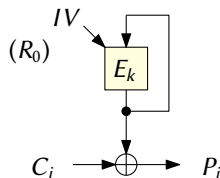
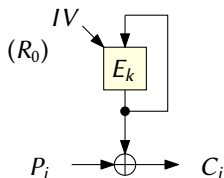
...

$$C_{i+1} = E_k(C[i-15, \dots, i])[0] \oplus P_{i+1}$$

CFB8 variant of CFB mode and Zerologon 2

- ▶ function in Netlogon implementation used all-zero IV (always)
- ▶ consider all-zero plaintext
 - ▶ $1/256$ of all keys lead to all-zero ciphertext
- ▶ client authenticates by encrypting his own challenge with a session key
 - ▶ the attacker chooses all-zero challenge
 - ▶ session-key is unknown
 - ▶ the attacker succeeds with probability $1/256$
 - ▶ if unsuccessful try again (session-key will change since it depends on server challenge as well)
- ▶ the attack requires more than this, but this is the core problem

OFB (Output Feedback)



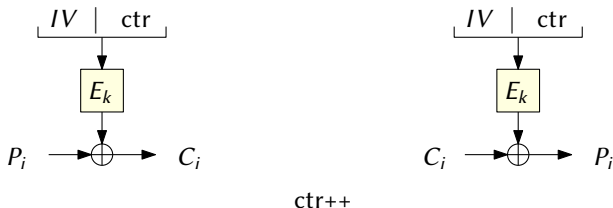
$$R_i = E_k(R_{i-1})$$

$$\text{encrypt: } C_i = P_i \oplus R_i$$

$$\text{decrypt: } P_i = C_i \oplus R_i$$

- ▶ synchronous stream cipher; D_k not needed
- ▶ IV should be unique for each plaintext
- ▶ neither encryption nor decryption can be parallelized
- ▶ single bit change in plaintext/ciphertext causes single bit change in ciphertext/plaintext (easy to flip plaintext bits)

CTR (Counter)



encrypt: $C_i = P_i \oplus E_k(\text{IV} \parallel \text{ctr})$

decrypt: $P_i = C_i \oplus E_k(\text{IV} \parallel \text{ctr})$

- ▶ inputs to E_k should not overlap (otherwise ...two-time pad)
- ▶ similar to OFB (synchronous stream cipher)
- ▶ similar properties of changing ciphertext bits
- ▶ easy to perform a seek (random access)
- ▶ easy to encrypt and decrypt in parallel

Padding

- ▶ ECB and CBC assume: $n \mid |\text{plaintext}|$
(i.e. n divides the length of plaintext)
- ▶ padding required (various paddings used):
 - ▶ bit padding – append 1 (always) and necessary number of 0's:
 $\text{msg} \parallel 1000\dots 0$
 - ▶ byte padding (PKCS #7, CMS (RFC 5652)):
 $\text{msg} \parallel 01$ if $n \mid |\text{msg}| + 1$
 $\text{msg} \parallel 03\ 03\ 03$ if $n \mid |\text{msg}| + 3$
 $\text{msg} \parallel 10\ 10\ \dots\ 10$ if $n \mid |\text{msg}|$ (for $n = 128$)
 - ▶ similarly for TLS 1.2 (RFC 5246): 00; 02 02 02; 0F 0F ... 0F
- ▶ padding $\Rightarrow |\text{ciphertext}| > |\text{plaintext}|$
- ▶ padding should be verified after decryption
- ▶ “stream” modes like OFB, CTR or CFB do not need padding
 - ▶ $|\text{ciphertext}| = |\text{plaintext}|$

Padding oracle attack 1

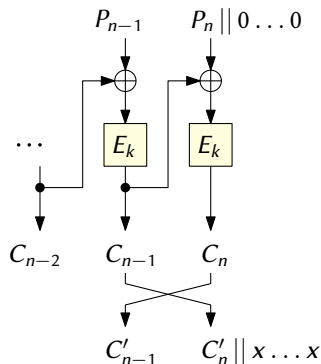
- ▶ implementation issue
- ▶ our assumptions:
 - ▶ CBC mode
 - ▶ we can recognize correct/incorrect padding, e.g. a server behaves differently (observable error, timing differences, ...)
- ▶ goal: decrypt ciphertext block C , i.e. compute $Y = D_k(C)$
- ▶ the attack:
 - ▶ try ciphertexts (assume 16-byte block, X is a random 15-byte value):
 $(X \parallel 00) \parallel C, (X \parallel 01) \parallel C, \dots, (X \parallel \text{7A}) \parallel C, \dots, (X \parallel \text{FF}) \parallel C$,
until we find a ciphertext with valid padding
 - ▶ the highest probability: the corresponding plaintext ends with byte 01
(and not with bytes 02 02 or even longer padding)
 - ▶ there is always a candidate with 01 padding, we can also alter the penultimate byte of X to distinguish it
 - ▶ finally, we can compute Y_{15} , e.g. $(7A \oplus Y_{15}) = 01 \Rightarrow Y_{15} = 7B$

Padding oracle attack 2

- ▶ the attack (cont.):
 - ▶ set the last byte of the first block to get 02 as the final byte of the plaintext: $b \oplus Y_{15} = 02$, i.e. $b = 79$
 - ▶ try ciphertexts (X is a random 14-byte value):
 $(X \parallel 00 \parallel 79) \parallel C, (X \parallel 01 \parallel 79) \parallel C, \dots,$
 $(X \parallel \text{B2} \parallel 79) \parallel C, \dots, (X \parallel FF \parallel 79) \parallel C,$
until we find a ciphertext with valid padding (this time: 02 02)
 - ▶ we can compute Y_{14} , e.g. $(B2 \oplus Y_{14}) = 02 \Rightarrow Y_{14} = B0$
...similarly for other bytes
- ▶ a variant used against SSL/TLS implementations (Lucky Thirteen, 2013)

Ciphertext stealing 1

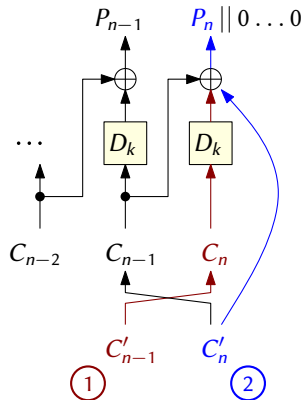
- ▶ method of avoiding padding for CBC or ECB modes
- ▶ ciphertext stealing for CBC mode encryption
 - ▶ example: Kerberos, AES256-CTS



plaintext: $\dots P_{n-2}, P_{n-1}, P_n$
ciphertext: $\dots C_{n-2}, C'_{n-1}, C'_n$

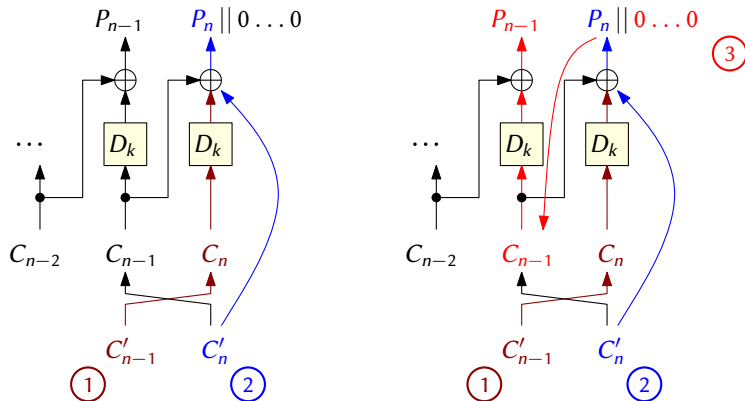
Ciphertext stealing 2

- ▶ decrypting CBC ciphertext stealing:



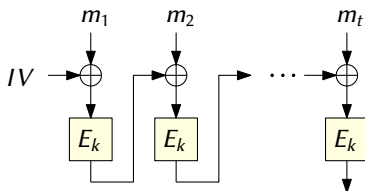
Ciphertext stealing 2

- ▶ decrypting CBC ciphertext stealing:



Authenticated encryption

- ▶ modes providing confidentiality & authenticity of data
- ▶ e.g. CCM (Counter with CBC-MAC), GCM (Galois/Counter Mode)
- ▶ CCM (idea):
 - ▶ plaintext encrypted using CTR mode
 - ▶ authentication tag computed as CBC-MAC
 - ▶ authenticate-then-encrypt (single key is used)
 - ▶ two-pass scheme (two E transforms for each input block)



Authenticated encryption – GCM

- ▶ faster than CCM
- ▶ simplified GCM:
 - ▶ single key is used
 - ▶ plaintext encrypted using CTR mode (starting with $\text{inc}(J_0)$, an incremented “pre-counter block”, derived from IV, e.g. $J_0 = \text{IV} \parallel 0^{31} \parallel 1$)
 - ▶ authentication tag computed as follows:
$$\text{CTR}_K(\text{GHASH}_H(A, C)) = E_K(J_0) \oplus \text{GHASH}_H(A, C)$$
 - ▶ A – additional authenticated data, C – ciphertext
 - ▶ $H = E_K(0)$
 - ▶ $A \parallel C \mapsto X_1, \dots, X_{n-1}, \underbrace{\text{len}(A) \parallel \text{len}(C)}_{X_n}$

$$\begin{array}{l} Y_0 = 0, Y_i = (Y_{i-1} \oplus X_i) \bullet H \\ \text{GHASH}_H(A, C) \leftarrow Y_n \end{array}$$

- ▶ \bullet is multiplication in $\text{GF}(2^{128})$ (generated by $x^{128} + x^7 + x^2 + x + 1$)
- ▶ IV must be unique for a given key (otherwise “forbidden attack”)

Format-preserving encryption

- ▶ goal: encrypt fixed-length string over some alphabet while preserving the length and the format of data
 - ▶ examples: credit card numbers, social security numbers, ...
- ▶ various proposals; the most important are FFX schemes
- ▶ based on Feistel structure
- ▶ see NIST SP 800-38G for recommended modes (and details)
- ▶ we present a very simplified (illustration) variant of FFX
- ▶ notation:
 - ▶ r – radix, the size of input alphabet
 - ▶ input plaintext X (string in base r) of length n (even)
 - ▶ AES-128 – underlying block cipher; no tweak
 - ▶ PRF – pseudorandom function (based on AES)

Simplified FFX variant

1. $A || B \leftarrow X$ (left and right halves)
 2. $P = \text{constants} || r || n$
 3. for i from 0 to 9: (10 rounds of Feistel structure)
 - 3.1 $R = \text{PRF}_k(P || i || B)$
 - 3.2 $S = R || \text{AES}_k(R \oplus 1) || \text{AES}_k(R \oplus 2) \dots$ (sufficiently long)
 - 3.3 $C = (A + S) \bmod r^{n/2}$
 - 3.4 $A = B$
 - 3.5 $B = C$
 4. return $A || B$
- omitted – all conversions from strings to bytes (and back)

XTS – encryption of block devices

- ▶ XEX-based Tweaked CodeBook mode (TCB) with CipherText Stealing (CTS)
- ▶ encryption of disks (block storage devices), e.g. File Vault 2, VeraCrypt, Bitlocker, etc.
- ▶ IEEE standard; approved by NIST

$$C_i = E_{k_1}(P_i \oplus T_n) \oplus T_n; \quad T_n = E_{k_2}(n) \bullet a^i$$

where

- ▶ k_1, k_2 are two independent keys
- ▶ n – data unit number (e.g. 512B in VeraCrypt)
- ▶ i – block number in particular data unit
- ▶ \bullet – multiplication in $GF(2^{128})$ (defined by $x^{128} + x^7 + x^2 + x + 1$)
- ▶ a – a primitive element in $GF(2^{128})$ (corresponds to polynomial x)

Theoretical security of block ciphers

- ▶ just an idea is presented
- ▶ E is an n -bit cipher
- ▶ attacker A (distinguisher)
- ▶ two different scenarios — A with oracle access to
 - ▶ $E_k(\cdot)$ and $E_k^{-1}(\cdot)$ for fixed, randomly chosen k
 - ▶ π and π^{-1} for randomly chosen permutation on $\{0, 1\}^n$
- ▶ block cipher is strong pseudorandom permutation if for any (efficient) attacker A the probability of distinguishing these scenarios is negligible
- ▶ “standard model” (other models of security exist, e.g. “ideal cipher” model – random independent permutation for every key)