IK2206 – Internet Security and Privacy
Chapter 4 – Modes of Operation

4.1 Introduction (READ)

4.2 Encrypting a Large Message (READ)

4.2.1 Electronic Code Book (ECB) (READ)

- Can you explain how ECB works when encrypting/decrypting a large message?

  The message is divided into 64-bit blocks and encrypts each block with the secret key.

- What weaknesses of ECB can be used by a "passive" attacker? By an "active attacker"?

  Corresponding two blocks of ciphertext will be identical.

  A passive attacker can figure out which sets of employees have salaries in the same range if a salary list is transmitted.

  If the attacker is an employer then can modify his salary by coping the ciphertext block of another employee to his own entry.

- Do you recommend the use of ECB?

  No, I can't recommend ECB because seeing the ciphertext can acquire information from repeated blocks, and it is also possible to rearrange blocks to his own advantage.

- When considering the monoalphabetic and polyalphabetic ciphers described earlier in the book, which has most similarities with ECB?

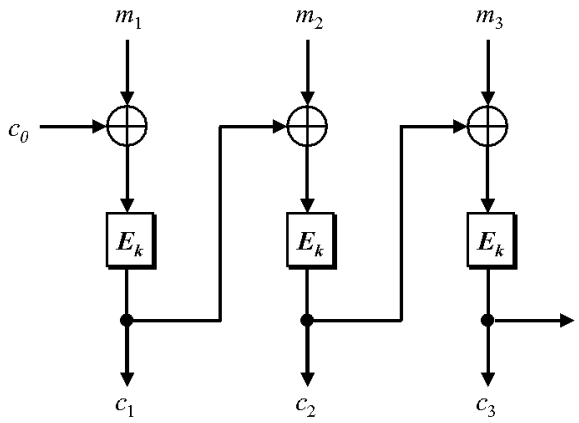
  It is similar with monoalphabetic ciphers because there it is possible to gain the information which letter is used to encrypt a specific letter. Fixed substitutions are used for the whole message. For instance A->N; B->G; C->D

4.2.2 Cipher Block Chaining (CBC) (READ)

- Can you describe how CBC works? Are you able to draw and explain figure such as Fig 4-5?

  CBC generates his own "random numbers". At the beginning an initially random number is generated. The random number is XOR with the m1 and afterwards encrypted with the secret key. The output c_1 *is used as new random number for* m_2.

  The first random number must also be transmitted to the receiver.



- What is an "initialization vector" (IV)?

  The initial random number is known as an initialization vector.

- Is the IV a secret value? If Alice sends a message to Bob using CBC how will Bob know which IV Alice used?

  The IV is also transferred to Bob.

- How should the IV be chosen? (Reading the book is ok, but Wikipedia may give more info on this.)

  The IV is a random number/string. A basic requirement is uniqueness, which means that no IV may be reused under the same key. (Equal IV and equal plaintext result in equal ciphertext). Furthermore, an IV must not be predictable.

- Are there cases when the IV could be omitted? Would that still be better than ECB?

  In many cases the security of CBC would not be adversely affected by omitting the IV. With CBC it is possible to detect the first differ from previous message if a message is sent in a defined interval (weekly, daily etc.). For instance transferring weekly a salary list, it would be easy to detect salary changes compared to the previous week.

  CBC without IV is still better than ECB because in CBC it is not possible to identify identical parts in a message.

- What benefit(s) are there of using the ciphertext of one block a input (XOR) to the encryption of the next block (as done in Fig 4-5) as opposed to pass a separate random number for each data block (as done in Fig 4-4)? What about efficiency?

  It is only necessary to pass the Initially vector(random number) to the receiver. Otherwise we must pass all the random numbers (for each block) to the receiver. The ciphertext why have to pass anyway therefore it is efficient to use it as random number.

- What active attack(s) is the scheme in Fig 4-4 more vulnerable to?

An attacker can rearrange the blocks and have a predictable effect on the resulting plaintext. For instance: if r2/c2 is swapped with r7/c7, then m2 and m7 would be swapped in the result. -> This is not possible if we use the output from the previous message as random number for the next round as it works in CBC.

- What would happen if ciphertext blocks arrive out of order?

  We would have to wait for the missing part of the ciphertext. In CBC, CFB, and OFB modes, encryption can't really be parallelized because the ciphertext for a certain block is necessary to create the ciphertext for the next block; thus, we can't compute ciphertext out of order.

  - What if a ciphertext packet is lost? How many plaintext block will be affected? (One, two, or more? All?)

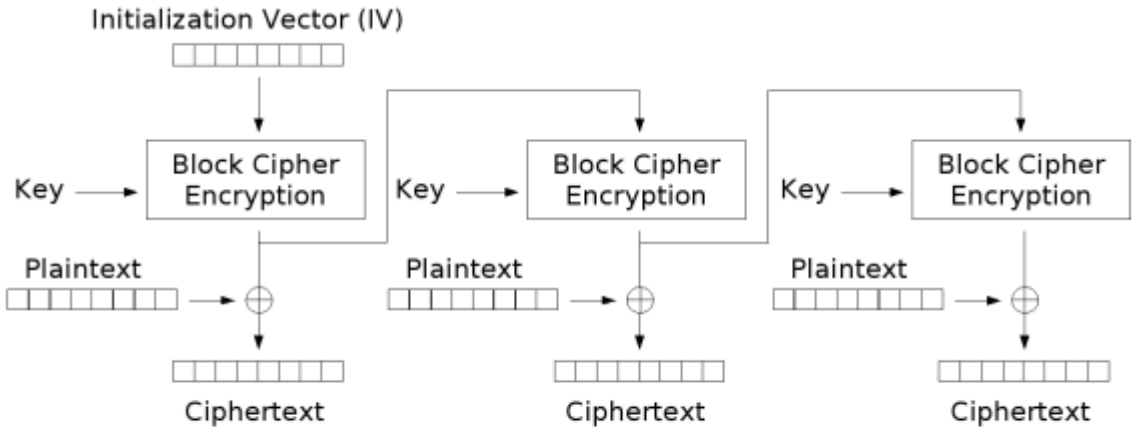
    I depends which block is lost. As each block depends on the previous block, all blocks after the lost block are affected.

4.2.2.1 CBC Threat 1 - Modifying Ciphertext Blocks (READ BRIEFLY)

4.2.2.2 CBC Threat 2 - Rearranging Ciphertext Blocks (READ BRIEFLY)

4.2.3 Output Feedback Mode (OFB) (READ BRIEFLY)

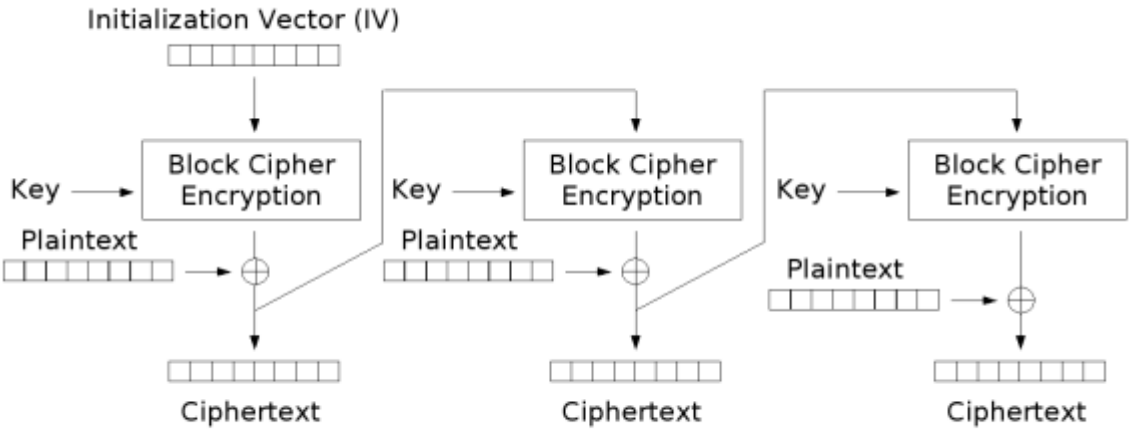
- Are you able to describe how block ciphers can be run in "stream cipher mode", i.e., to create one-time pads which can be XORed with the message? (That's what OFB does.)



Output Feedback (OFB) mode encryption

In that mode we use also an initialization vector. The output of the block cipher encryption is XOR'd with the plaintext. In addition the output of the block cipher encryption is used as key for the next block, so that there a continuous stream of encrypted blocks arises. The bit stream is pseudo randomly because it depends on the block encryption, key and initialization vector.

4.2.4 Cipher Feeback Mode (CFB) (READ BRIEFLY)



Cipher Feedback (CFB) mode encryption

Cipher Feedback is very similar to OFB. The difference is that the key for the next block is the ciphertext of the previous block.
With OFB and CBC, if characters are lost in transmission, or extra characters are added to the ciphertext stream, the entire rest of the transmission is garbled. It means that if a byte is lost in transmission, one byte of plaintext will be lost and the next 8 bytes will be garbled.

- Is CFB a "stream cipher"?

  Yes, a self-synchronizing stream cipher is a block cipher in cipher feedback mode.

4.2.5 Counter Mode (CTR) (READ BRIEFLY)

4.3 Generating MACs (READ)

- What is a MAC/MIC?

  MAC: message authentication code
  MIC: synonym for MAC, Message Integrity Code

  MAC/MIC is a cryptographic checksum. While CBC, CFB, OFB and CTR offer a good protection against attacker deciphering a message, none offers a good protection for attacker who already knows the content and just want to modify messages undetected.

- What is a CBC residue?

  CBC residue is the last block (output) when running CBC.
  A way for protecting against undetected modifications is to compute the CBC and transfer the CBC residue (last block) with the plaintext message. -> Does only ensure integrity and not privacy.

4.3.1 Ensuring Privacy and Integrity Together (SKIP)

4.3.2 CBC with a Weak Cryptographic Checksum (SKIP)

4.3.3 CBC Encryption and CBC Residue with Related Keys (SKIP)

4.3.4 CBC with a Cryptographic Hash (SKIP)

4.3.5 Offset Codebook Mode (OCB) (SKIP)

4.4 Multiple Encryption DES (READ)

- What is the reason for run DES multiple times instead of only one when encrypting the data? Are there some drawbacks with doing it "too many" times?

  The more times the block is encrypted the more secure it is.

  The drawback of doing it to many times is that it is expensive to do an encryption.

- What does it mean to run 3DES in "EDE"-mode? Was there any reason for designing 3DES with "EDE"-mode rather than "EEE"-mode?

  Encrypt with key 1, Decrypt with key 2, Decrypt with key 1

  EEE is only a factor of three for the attacker is not considered much added security, especially since the good gays also have more work to do.

  For encrypting twice with two key a known attack (not really practical) exists.

  Against Triple Encryption with only two keys no attacks other than the straightforward brute-force search is known -> therefore EDE is used.

- How many 64 bit keys (or 56 bit keys to be precise) are commonly used for 3DES?

  Usually two keys are used. Some systems do implement 3DES with three independent keys, but this is not the standard.

(You may need to look at the following subsections to find some answers.)

4.4.1 How Many Encryptions? (READ BRIEFLY)

4.4.2 CBC Outside vs. Inside (SKIP)