1. Differential cryptanalysis requires to examine plaintexts with a given differences on a given positions. So if you perform a CPA attack, you can prepare the pairs in this way.

   What to do if the attack is a known plaintext attack (KPA): you get a large number of pairs (plaintext,ciphertext), but you cannot choose them. Is differential cryptanalysis still applicable?

Krótka odpowiedź: to zależy.

Dłuższa odpowiedź: jeśli plaintexty są wybierane z odpowiednim rozkładem (np. jednostajnym) i mamy ich odpowiednio dużo, istnieje szansa, że znajdziemy wystarczająco dużo par, które spełnią wykorzystywane w ataku zależności (jeżeli plaintexty są dobierane tak, aby unikać takich par, nawet ich duża liczba może być niewystarczająca).

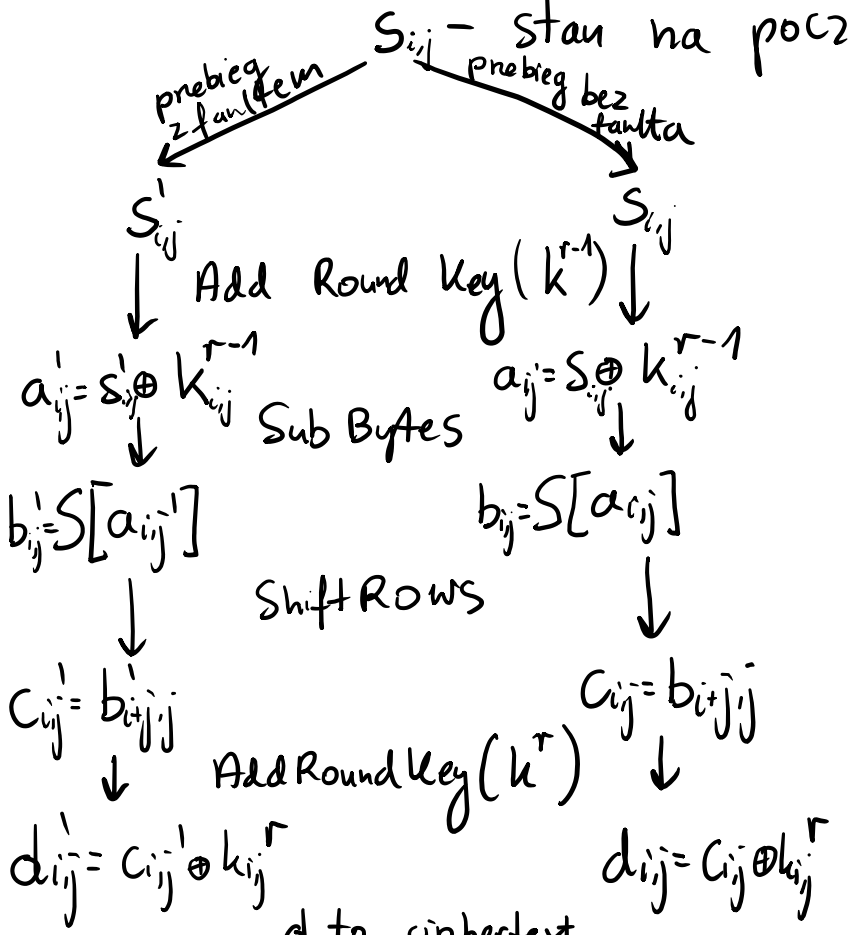
To też zależy od samego schematu i tego, na który jego element jest przeprowadzany atak (i jak często występuje statystycznie wykorzystywana w ataku zależność); jeśli interesuje nas np. różnica w wejściu konkretnego S-boxa, szansa na znalezienie takich par jest duża, a jeśli potrzebujemy np. plaintextów różniących się tylko jednym bitem, szansa jest bardzo mała.

2. Explain in detail how to use differential cryptanalysis together with faults to break AES. By generating fault we mean flipping a bit on a chosen position of a chosen intermediate value of the computation.

a) Can linear cryptanalysis be directly applied together with faults of the same kind? Could linear cryptanalysis be simplified in case that you may set certain bits of intermediate values to 1?

(W AES-128 mając którykolwiek klucz rundowy, możemy policzyć klucz główny)

a) Atakujemy ostatnią rundę AES-a (w ostatniej rundzie nie ma MixColumns)



$s' \oplus s = \Delta$ (różnica na jednym bicie)

Możemy zauważyć, że $a' \oplus a = (s' \oplus k^{r-1}) \oplus (s \oplus k^{r-1}) = \Delta$
(więc nie ma znaczenia, czy zrobimy fault po czy przed przedostatnim AddRoundKey).

SubBytes "rozprowadzi" błąd po całym bajcie i zachowa liniowość.
ShiftRows jedynie popreastawia bajty, zatem między d i d' różnica będzie tylko na jednym bajcie. Dla tego bajtu możemy sprawdzić wszystkie 256 możliwości wartości bajtu k^r; dla każdej możliwości "cofamy" obliczenia do stanu $a_{i,j}$ oraz $a'_{i,j}$ i patrzymy, czy $a \oplus a' = \Delta$. Jeśli tak, to testowany bajt może być poprawnym bajtem klucza k^r (można eksperymentalnie sprawdzić, że po takiej operacji zostanie nam max 4 kandydatów).
Atak powtarzamy dla pozostałych bajtów klucza. Mając maksymalnie 4 kandydatów na każdy bajt, znajdujemy klucz metodą brute force (max 4^{16} możliwości).

b) Taki sam fault nie będzie użyteczny w kryptoanalizie liniowej (liniowość będzie zmieniana na proporcjonalną do niej afiniczność), np. z równania

$A \oplus B \oplus C = 0$ po faulcie zrobi nam się $A \oplus B \oplus \underbrace{(C \oplus 1)}_{\text{„}C\text{”}} = 0$, czyli $A \oplus B \oplus C = 1$.

$P[A \oplus B \oplus C = 0] = \frac{1}{2} + \delta$ i $P[A \oplus B \oplus C = 1] = (1 - \frac{1}{2} - \delta) = \frac{1}{2} - \delta$.

W kryptoanalizie liniowej interesuje nas tylko $|\delta|$, zatem taki fault nam nie pomaga.

Natomiast możliwość ustawiania konkretnych bitów w stanie na 1 może nam pomóc, bo zmniejsza liczbę zmiennych w równaniu, np.

$A = 0$ z ppb $\frac{1}{3}$
$B = 0$ z ppb $\frac{2}{3}$
$C = 0$ z ppb $\frac{2}{3}$

$$P[A \oplus B \oplus C = 0] = \underbrace{\frac{1}{3}\cdot\frac{2}{3}\cdot\frac{2}{3}}_{\text{same 0}} + \underbrace{\frac{1}{3}\cdot\frac{1}{3}\cdot\frac{1}{3}}_{\text{tylko } A=0} + \underbrace{\frac{2}{3}\cdot\frac{2}{3}\cdot\frac{1}{3}}_{\text{tylko } B=0} + \underbrace{\frac{2}{3}\cdot\frac{1}{3}\cdot\frac{2}{3}}_{\text{tylko } C=0} = \frac{13}{27} = \frac{1}{2} - \frac{1}{54}$$

Jeśli C zawsze faultujemy na 1, to

$$P[A \oplus B \oplus C = 0] = \underbrace{\frac{1}{3}\cdot\frac{1}{3}\cdot 1}_{\text{tylko } A=0} + \underbrace{\frac{2}{3}\cdot\frac{2}{3}\cdot 1}_{\text{tylko } B=0} = \frac{5}{9} = \frac{1}{2} + \frac{1}{18}$$

$\left|\frac{1}{18}\right| > \left|\frac{1}{54}\right|$, zatem mamy większy bias.

(ta odpowiedź jest generyczna i dotyczy nie tylko AESa)

Na marginesie:

Mogąc ustawiać dowolny bit stanu na 1 przed ostatnim AddRoundKey, w ciphertexcie na tym samym indeksie dostaniemy zanegowany bit klucza (ostatniej rundy).
Jeśli fault zrobimy przed ostatnim ShiftRows, będzie podobnie - jedynie indeks będzie przesunięty.
Jeśli taki fault zrobimy przed SubBytes (8 razy, na każdy bit w bajcie stanu), możemy poznać wartość tego bajtu (jeśli fault zmienia wynik, to bit był zerem, jeśli nie, to był jedynką). Znając wartość tego bajtu, możemy policzyć odpowiadający mu bajt klucza ostatniej rundy.

3. For each of the block encryption modes discussed during the lecture:

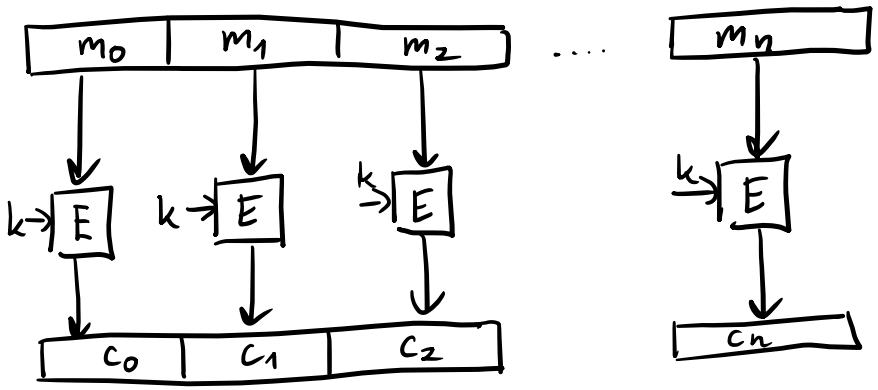
(a) check what happens with the plaintext, if just one bit is flipped in the ciphertext,

(b) check what happens with the ciphertext, if just one bit is flipped in the plaintext,

(c) what would be the consequences of repeating the same initial vector IV (provided that IV is used)?

[1/4]

ECB



a) bitflip w c_i zmienia cały* m_i

b) bitflip w m_i zmienia cały* c_i

c) n/a

* z właściwości efektu lawinowego, zmieni się średnio połowa bitów

3. For each of the block encryption modes discussed during the lecture:

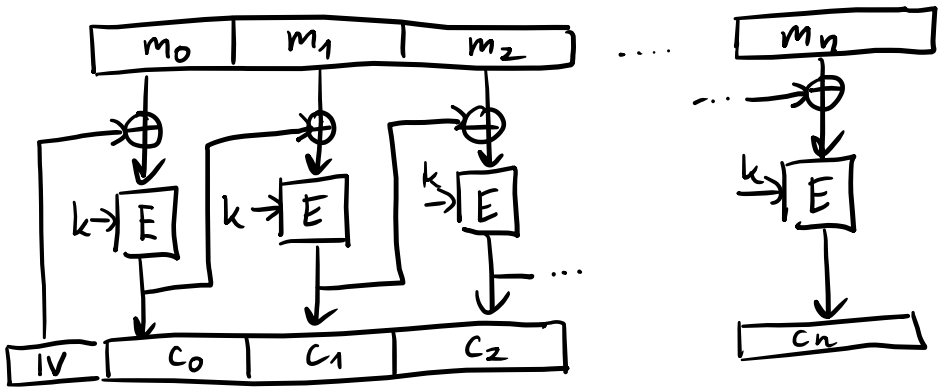
   (a) check what happens with the plaintext, if just one bit is flipped in the ciphertext,

   (b) check what happens with the ciphertext, if just one bit is flipped in the plaintext,

   (c) what would be the consequences of repeating the same initial vector IV (provided that IV is used)?

[2|4]

CBC



a) bitflip w c_i zmienia 1 bit na tym samym indeksie w m_{i+1} oraz całe* m_i.

b) bitflip w m_i zmienia całe* $c_i, c_{i+1}, \dots, c_n$.

c) Tracimy CPA-security; bloki c_i będą takie same do pierwszego różniącego się bloku m_j $(i<j)$.

* z właściwości efektu lawinowego, zmieni się średnio połowa bitów

3. For each of the block encryption modes discussed during the lecture:

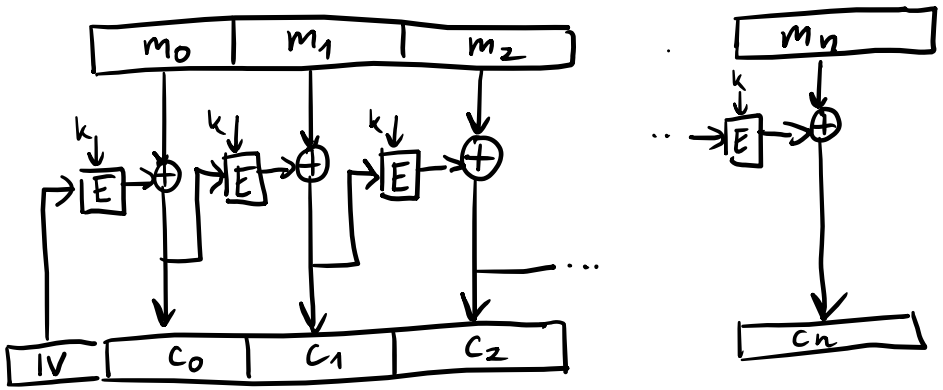
   (a) check what happens with the plaintext, if just one bit is flipped in the ciphertext,

   (b) check what happens with the ciphertext, if just one bit is flipped in the plaintext,

   (c) what would be the consequences of repeating the same initial vector IV (provided that IV is used)?

[3/4]

CFB



a) bitflip w c_i zmienia 1 bit na tym samym indeksie w m_i oraz całe* m_{i+1}

b) bitflip w m_i zmienia 1 bit na tym samym indeksie w c_i oraz całe* $c_{i+1}, c_{i+2}, \dots, c_n$.

c) Tracimy CPA-security; bloki c_i będą takie same do pierwszego różniącego się bloku m_j $(i<j)$; $\Delta m_j = \Delta c_j$.

* z właściwości efektu lawinowego, zmieni się średnio połowa bitów

3. For each of the block encryption modes discussed during the lecture:

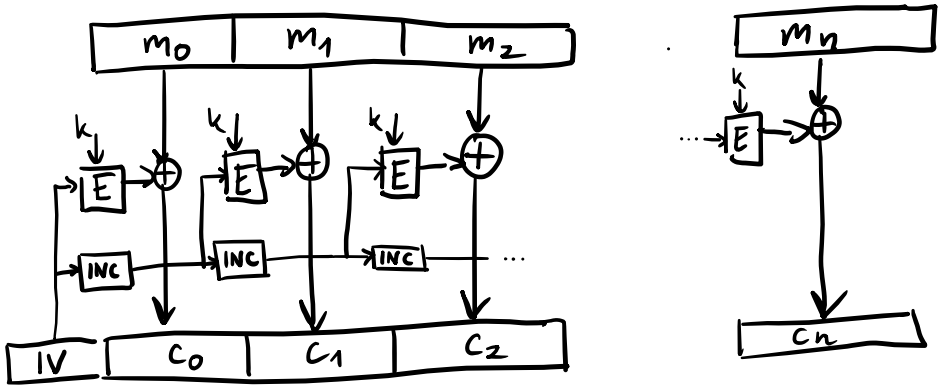
   (a) check what happens with the plaintext, if just one bit is flipped in the ciphertext,

   (b) check what happens with the ciphertext, if just one bit is flipped in the plaintext,

   (c) what would be the consequences of repeating the same initial vector IV (provided that IV is used)?

[4/4]

CTR



a) bitflip w c_i zmienia 1 bit na tym samym indeksie w m_i

b) bitflip w m_i zmienia 1 bit na tym samym indeksie w c_i

c) problemy bezpieczeństwa identyczne z wielkrotnie użytym OTP

($\Delta m = \Delta c$)

4. During the lecture we have shortly discussed why ECB encryption mode is a wrong choice for encrypting data concerning bank operations to be executed by the recipient server.

So is CBC a good choice in this case? You do not know the exact format of the transmission, size of the records, etc. So is CBC a good choice regardless what data format we use?

CBC również nie będzie dobrym wyborem.
Na wykładzie zostały wspomniane dwa problemy ECB w tym kontekście:

1) $c_i = c_j$, jeśli $m_i = m_j$ – ten problem akurat nie wystąpi w CBC, więc tu jest OK.

2) możliwość podmiany całych bloków ciphertextu na coś innego (co np. odszyfruje się do wyższych kwot przelewu) – CBC również ma podobny problem; z zadania 3. wiemy, że zmiana 1 bitu w c_i zmieni nam 1 bit w m_i i całe m_{i+1}. Zatem, jeśli wiemy, że kwota przelewu jest w bloku m_{i+1} możemy zmienić 1 lub więcej bitów w c_i i uzyskać podobny efekt, co w przypadku ECB (zmiana w m_i może być np. uznana za literówkę). Co więcej, zmieniając bity w c_i, wywołujemy identyczną zmianę w m_i (kosztem całkowitego popsucia m_{i+1}). Taki atak może być użyteczny, jeśli wiemy, czego spodziewać się w m_i i popsucie m_{i+1} nie grozi odrzuceniem transakcji (lub gdy m_i jest ostatnim blokiem).

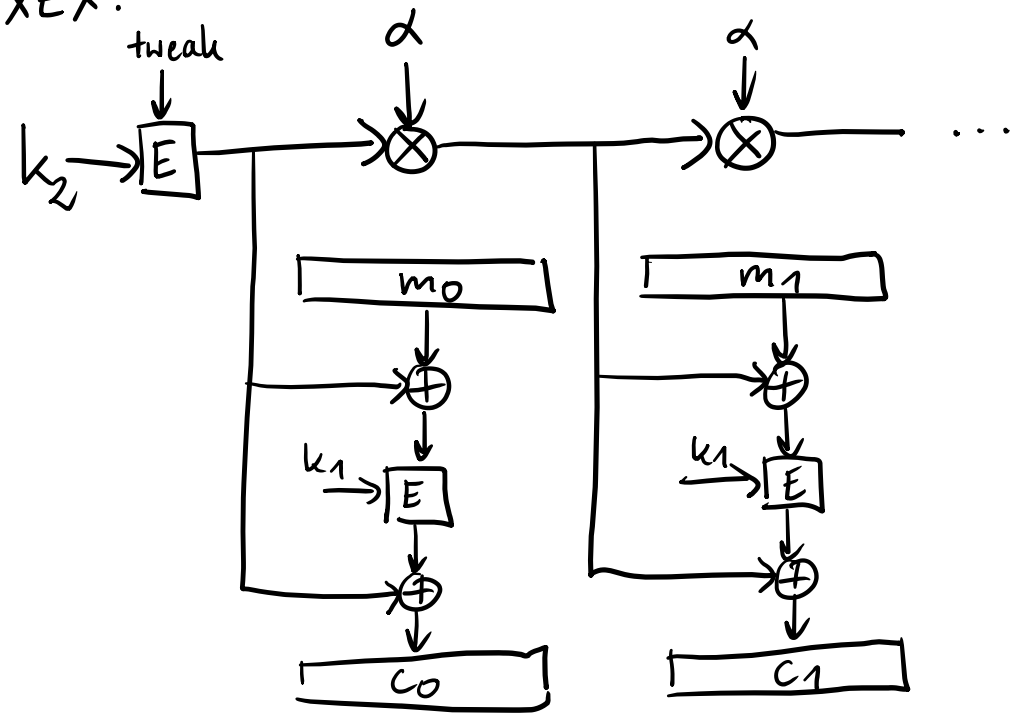
Aby uniknąć tego typu problemów, należałoby użyć któregoś z trybów AEAD lub Encrypt&MAC.

6. For disk encryption none of the encryption modes discussed during the lecture is used in a pure form. Find in the literature how disk encryption is organized. Deduce why it has been designed in this way.

- dysk jest dzielony na szyfrowane osobno sektory.
- IV wylicza się za pomocą numeru sektoru (przy generowanym losowo zajmowałoby nadmiarowe miejsce na dysku).
  Tryby z IV są potrzebne, aby dwa takie same sektory nie zaszyfrowały się do tego samego.
- CTR i inne podobne tryby nie są dobrym pomysłem - zmiana danych odpowiadałaby identycznej zmianie ciphertextu.
- zmiana danych w jednym miejscu nie powinna skutkować dużą zmianą na zaszyfrowanym dysku.

Rozwiązania: XEX, XTS (np. TrueCrypt, BitLocker, wolfCrypt ...)

XEX:



⊕ - XOR

⊗ - mnożenie $GF(2^{128})$
(modulo wielomian $x^{128}+x^7+x^2+x+1$)

k_1, k_2 - niezależne klucze

tweak - np. numer sektora

α - x w $GF(2^{128})$

XTS to XEX z ciphertext stealingiem (patrz: zadanie 1 z listy 4).