Laby 3

October 15, 2019

1 Szyfrowanie blokowe

1.1 Tryby kodowania

Poniewa wiadomoci, jakie chcemy zakodowa, s zwykle znacznie wiksze od rozmiaru bloku, musimy uywa jakiego trybu kodowania. Najbardziej naiwne podzielenie wiadomoci na bloki odpowiednich rozmiarów i zakodowanie osobno kadego z nich (ECB) nie zapewnia nam bezpieczestwa.

1.1.1 ECB

Tryb elektronicznej ksiki kodowej (ang. Electronic CodeBook – ECB) – jeden z najprostszych trybów szyfrowania wiadomoci z wykorzystaniem szyfru blokowego. W trybie tym blok tekstu jawnego jest szyfrowany w blok szyfrogramu. Moliwe jest niezalene szyfrowanie oraz deszyfrowanie bloków wiadomoci, takie zachowanie pozwala, teoretycznie, stworzy ksik kodow tekstu jawnego i odpowiadajcemu mu szyfrogramu, która bdzie zawieraa 2n rónych wpisów (n – dugo bloku w bitach).

Wad tego trybu jest fakt, e kryptoanalitycy, dysponujc kilkoma tekstami jawnymi i odpowiadajcymi im szyfrogramami, mog rozpocz odtwarzanie ksiki kodowej – dla szyfrów z dugimi kluczami cakowite odtworzenie ksiki kodowej jest jednak nierealne. Atakujcy ma take moliwo zmiany wiadomoci bez znajomoci klucza.

1.1.2 CBC

Tryb wizania bloków zaszyfrowanych (z ang. Cipher Block Chaining – CBC) – jeden z trybów pracy szyfrów blokowych wykorzystujcy sprzenie zwrotne, samosynchronizujcy si; w trybie tym blok tekstu jawnego jest sumowany modulo 2 z szyfrogramem poprzedzajcego go bloku w zwizku z czym wynik szyfrowania jest zaleny od poprzednich bloków. Pierwszy blok, przed zaszyfrowaniem, jest sumowany modulo dwa z losowo wygenerowanym wektorem pocztkowym IV (ang. initialization vector), wektor ten nie musi by utrzymywany w tajemnicy.

1.1.3 CFB

Tryb sprzenia zwrotnego szyfrogramu (z ang. Cipher Feedback – CFB) – jeden z trybów dziaania szyfrów blokowych, przeznaczony do szyfrowania strumieni danych. Szyfrowanie nie moe by jednak rozpoczte zanim nie zostanie odebrany peny blok danych do zaszyfrowania.

Szyfr blokowy dziaajcy w trybie sprzenia zwrotnego szyfrogramu dziaa na rejestrze, który jest w stanie pomieci peny blok danych przeznaczonych do szyfrowania. Przed rozpoczciem

procedury szyfrowania rejestr ten wypeniany jest losowymi danymi, które umownie nazwane s wektorem pocztkowym (ang. IV - initialization vector). Zawarto tego rejestru jest szyfrowana a nastpnie n-skrajnych, lewych bitów jest sumowana modulo dwa z n pierwszymi bitami tekstu jawnego – w ten sposób powstaje pierwsze n-bitów szyfrogramu. Zaszyfrowane w ten sposób bity zapisywane s na n-skrajnych, prawych bitach kolejki, jednoczenie pozostae bity kolejki przesuwane s w lewo i procedura szyfrowania jest powtarzana[1].

Liczba n jest zalena od trybu CFB – moliwe jest szyfrowanie bit po bicie (1-bitowy CFB), bajt po bajcie (8-bitowy CFB) lub dowolne inne.

---

ródo: Wikipedia

1.2 Piaskownica

1.2.1 Wczytanie bibliotek

[1]:
```
from Crypto.Cipher import DES,AES
from Crypto.Random import get_random_bytes
from typing import Iterable
from itertools import product
import math
import hashlib
```

1.2.2 Przydatne funkcje

[2]:
```
def padd_data(data : bytes, block_size : int) -> bytes:
    padding = ord('@')
    diff = block_size - len(data) % block_size

    return data + bytes([padding] * diff)

def strToCodes(string : str) -> Iterable[int]:
    return (ord(c) for c in string)

def codesToStr(codes : Iterable[int]) -> str:
    return ''.join((chr(c) for c in codes))
```

1.2.3 Szyfr DES

[3]:
```
key = 'key12345'
data = 'secret12'
iv = get_random_bytes(8)
print(data)

des = DES.new(key, DES.MODE_CBC, iv)
cryptogram = des.encrypt(data)
```

```
print(cryptogram)

des = DES.new(key, DES.MODE_CBC, iv)
restored = des.decrypt(cryptogram)
print(restored)
```

```
secret12
b'\xfa\xbaZ\x01\xa6\xad\x00\xa3'
b'secret12'
```

1.2.4 Szyfr AES

[4]:
```
key = 'key4567890123456'
data = 'Ala ma dwa koty.'
iv = get_random_bytes(16)

print(data)

des = AES.new(key, DES.MODE_CBC, iv)
cryptogram = des.encrypt(data)
print(cryptogram)

des = AES.new(key, DES.MODE_CBC, iv)
restored = des.decrypt(cryptogram)
print(restored)
```

```
Ala ma dwa koty.
b'd\xeb\xd4\xdf\xef\x95\x1b\xef\x84DV\xce\xc8\x12\xc5G'
b'Ala ma dwa koty.'
```

1.3 Zadania

Zadania z iSoda

1.3.1 Zadanie 1

Napisz algorytm obliczajcy entropi.
Implementacja na podstawie artykuu.

[5]:
```
def entropy(data : bytes) -> float:
    count = {i : 0 for i in range(256)}
    for b in data: count[b] += 1

    p = lambda b: count[b] / len(data)
    entropy = sum((p(b) * count[b] for b in range(256)))

    return 1 - entropy / len(data)
```

```
def iSod_entropy(data : bytes) -> float:
    count = {i : 0 for i in range(256)}
    for b in data: count[b] += 1
    p = lambda b: count[b] / len(data)

    entropy = 0
    for b in range(256):
        prob = p(b)
        if prob > 0:
            entropy += prob * math.log2(prob)
#     entropy = -sum((p(b) * math.log2(p(b)) for b in range(256)))
    return -entropy
```

1.3.2 Zadanie 2

Porównaj entropie tekstu naturalnego z entropia kryptogramu.

[6]:
```
poem = None
with open('Dziewczyna.txt') as file:
    poem = file.read()

e = entropy(poem.encode())
print(f'Entropia wiersza (pdf) = {e}')

e = iSod_entropy(poem.encode())
print(f'Entropia wiersza (iSod) = {e}')
print()

key = 'key12345'
iv = get_random_bytes(8)
data = padd_data(poem.encode(), 8)

des = DES.new(key, DES.MODE_CBC, iv)
cryptogram = des.encrypt(data)

e = entropy(cryptogram)
print(f'Entropia kryptogramu DES = {e}')

key = 'key4567890123456'
iv = get_random_bytes(16)
data = padd_data(poem.encode(), 16)

des = AES.new(key, DES.MODE_CBC, iv)
cryptogram = des.encrypt(data)

e = entropy(cryptogram)
```

```
print(f'Entropia kryptogramu AES = {e}')
```

```
Entropia wiersza (pdf) = 0.952523347795931
Entropia wiersza (iSod) = 4.9664311123485

Entropia kryptogramu DES = 0.9957451059504504
Entropia kryptogramu AES = 0.9956954089506173
```

Wyglda na to, e entropia liczona metod z iSoda nie jest noramlzowana.

1.3.3 Zadanie 3

Porównaj wynik szyfrowania w trybach ECB i CBC.
Jaka jest entropia kryptogramów?

[7]:
```
poem
with open('Dziewczyna.txt') as file:
    poem = file.read()

print('Syfr DES')
key = 'key12345'
data = padd_data(poem.encode(), 8)

des = DES.new(key, DES.MODE_ECB)
e = entropy(des.encrypt(data))
print(f'Entropia w trybie ECB: {e}')

iv = get_random_bytes(8)
des = DES.new(key, DES.MODE_CBC, iv)
e = entropy(des.encrypt(data))
print(f'Entropia w trybie CBC: {e}')

# des_cbc = DES.new(key, DES.MODE_CBC)
print('\nSyfr AES')
key = 'key4567890123456'
data = padd_data(poem.encode(), 16)

aes = AES.new(key, AES.MODE_ECB)
e = entropy(aes.encrypt(data))
print(f'Entropia w trybie ECB: {e}')

iv = get_random_bytes(16)
aes = AES.new(key, AES.MODE_CBC, iv)
e = entropy(aes.encrypt(data))
print(f'Entropia w trybie CBC: {e}')
```

```
Syfr DES
Entropia w trybie ECB: 0.9957271630418758
```

```
Entropia w trybie CBC: 0.9957785670502246

Syfr AES
Entropia w trybie ECB: 0.9956987847222222
Entropia w trybie CBC: 0.9957065007716049
```

1.3.4 Zadanie 4

Napisz program szyfrujcy pliki przy pomocy algorytmu AES w trybie CBC.

[8]:
```python
from pathlib import Path

def encryptFile(file_name : str, key : str) -> None:
    data = None
    with open(file_name, 'rb') as file: data = file.read()
    data = padd_data(data, 16)

    aes = AES.new(key, AES.MODE_CBC, iv)
    encrypted = aes.encrypt(data)

    new_file = Path(file_name).with_suffix('.aes')
    with open(new_file, 'wb') as output: output.write(encrypted)

encryptFile('Dziewczyna.txt', 'key4567890123456')
```

1.3.5 Zadanie 5

Zaproponuj algorytm tworzenia klucza na podstawie hasa podawanego przez czowieka.
Implementacja bazujca na algorytmie **KDF1**

```
INPUT:
Z, shared secret, a byte string;
Hash, hash function with output hLen bytes;
kLen, intended length of keying material in bytes;
[OtherInfo], optional extra shared material.
OUTPUT: Derived key, K, of length kLen bytes.

Set d = ceiling(kLen/hLen).
Set T = "", the empty string.
for Counter = 0 to d-1 do:
    C = IntegerToString(Counter, 4)
    T = T || Hash(Z || C || [OtherInfo])
Output the first kLen bytes of T as K.
```

---

ródo

[47]:
```python
def create_key_KDF1(secret : bytes, hash_name : str, desired_length : int, salt␣
↪: bytes = b'') -> bytes:
```

```
    h = hashlib.new(hash_name)
    d = math.ceil(desired_length / h.block_size)
    T = b''

    for i in range(d):
        c = i.to_bytes(4, byteorder='big')
        h = hashlib.new(hash_name)
        h.update(secret + c + salt)
        T += h.digest()

    return T[:desired_length]


create_key_KDF1(b'aa', 'sha256', 8, b'02')
```

[47]: `b'\xac\xbfK\\\xbd\xe9\xca\xa4'`

Autorska implementacja
Wielokrotne hashowanie poprzedniego wyniku i soli

[52]:
```
def create_key_simple(secret : bytes, desired_length : int, iterations : int,␣
↪salt : bytes = b'') -> bytes:
    if desired_length > hashlib.sha256().block_size:
        raise ValueError(f'This function can create keys of max {hashlib.
↪sha256().block_size} size. Recived {desired_length} lenght')

    result = secret
    for i in range(iterations):
        h = hashlib.sha256()
        h.update(result)
        h.update(salt)
        result = h.digest()

    return result[:desired_length]


create_key_simple(b'aa', 8, 1000, b'02')
```

[52]: `b'\xc1h\xb3w\x06\x99!\xb1'`

Wykorzystanie funkcji z biblioteki

[53]:
```
hashlib.pbkdf2_hmac('sha256', b'password', b'salt', 100000, 8)
```

[53]: `b'\x03\x94\xa2\xed\xe32\xc9\xa1'`

1.3.6 Zadanie 6

Okrel ile znaków [a-z] naley poda, eby entropia hasa zbliya si do 256-bitowego klucza AES.

[ ]:

1.3.7 Zadanie 7

Napisz program do ataku brutalnej siy na kryptogram przy wykorzystaniu entropii jako uniwersalnego kryterium zakoczenia algorytmu.

[10]:
```
def bruteforceDES(cryptogram : bytes, keyAlphabet = range(256)) -> None:
    entropyThreshold = 0.965

    possibleKeys = product(keyAlphabet, repeat=8)
    for k in possibleKeys:
        k = codesToStr(k)

        des = DES.new(k, DES.MODE_ECB)
        recovered = des.decrypt(cryptogram)

        e = entropy(recovered)
        if e < entropyThreshold:
            print('\nZnaleziono rozwizanie!\n')
            print(f'Klucz: {k}')
            print(f'Wiadomo: {recovered}')
            return

    print(f'Nie znaleziono rozwizania')
```

[11]:
```
key = 'abcabdbb'
data = padd_data(poem.encode(), 8)

des = DES.new(key, DES.MODE_ECB)
cryptogram = des.encrypt(data)

keyAlphabet = list(range(ord('a'), ord('e') + 1))
bruteforceDES(cryptogram, keyAlphabet)
```

```
Znaleziono rozwizanie!

Klucz: abbabdbb
Wiadomo: b'Boles\xc5\x82aw Le\xc5\x9bmian\n\nNap\xc3\xb3j
cienisty\nDziewczyna\n\nW\xc5\x82adys\xc5\x82awowi Jaroszewiczowi, Jego
entuzjastycznym zapa\xc5\x82om dla dzie\xc5\x82 tw\xc3\xb3rczych i szczerym
wyczuciom czar\xc3\xb3w poetyckich\n\nDwunastu braci, wierz\xc4\x85c w sny,
zbada\xc5\x82o mur od marze\xc5\x84 strony,\nA poza murem p\xc5\x82aka\xc5\x82
g\xc5\x82os, dziewcz\xc4\x99cy g\xc5\x82os zaprzepaszczony.\n\nI pokochali
g\xc5\x82osu d\xc5\xbawi\xc4\x99k i ch\xc4\x99tny domys\xc5\x82 o
Dziewczynie,\nI zgadywali kszta\xc5\x82ty ust po tym, jak \xc5\x9bpiew od
\xc5\xbcalu ginie\xe2\x80\xa6\n\nM\xc3\xb3wili o niej: \xe2\x80\x9e\xc5\x81ka,
wi\xc4\x99c jest!\xe2\x80\x9d \xe2\x80\x94 I nic innego nie m\xc3\xb3wili,\nI
przeprze\xc5\xbcegnali ca\xc5\x82y \xc5\x9bwiat, \xe2\x80\x94 i \xc5\x9bwiat
```

zaduma\xc5\x82 si\xc4\x99 w tej chwili\xe2\x80\xa6\n\nPorwali m\xc5\x82oty w
tward\xc4\x85 d\xc5\x82o\xc5\x84 i j\xc4\x99li w mury t\xc5\x82uc z
\xc5\x82oskotem!\nI nie wiedzia\xc5\x82a \xc5\x9blepa noc, kto jest
cz\xc5\x82owiekiem, a kto m\xc5\x82otem?\n\n\xe2\x80\x9eO, pr\xc4\x99dzej
skruszmy zimny g\xc5\x82az, nim \xc5\x9bmier\xc4\x87 Dziewczyn\xc4\x99
rdz\xc4\x85 powlecze!\xe2\x80\x9d \xe2\x80\x94\nTak, wal\xc4\x85c w mur,
dwunasty brat do jedenastu innych rzecze.\n\nAle daremny by\xc5\x82 ich trud,
daremny ramion sprz\xc4\x99g i usi\xc5\x82!\nOddali cia\xc5\x82a swe na strwon
owemu snowi, co ich kusi\xc5\x82!\n\n\xc5\x81ami\xc4\x85 si\xc4\x99 piersi,
trzeszczy ko\xc5\x9b\xc4\x87, pr\xc3\xb3chniej\xc4\x85 d\xc5\x82onie, twarze
bledn\xc4\x85\xe2\x80\xa6\nI wszyscy w jednym zmarli dniu i noc wieczyst\xc4\x85
mieli jedn\xc4\x85!\n\nLecz cienie zmar\xc5\x82ych \xe2\x80\x94 Bo\xc5\xbce
m\xc3\xb3j! \xe2\x80\x94 nie wypu\xc5\x9bci\xc5\x82y m\xc5\x82ot\xc3\xb3w z
d\xc5\x82oni!\nI tylko inny p\xc5\x82ynie czas \xe2\x80\x94 i tylko m\xc5\x82ot
inaczej dzwoni\xe2\x80\xa6\n\nI dzwoni wprz\xc3\xb3d! I dzwoni wspak! I
wzwy\xc5\xbc za ka\xc5\xbcdym grzmi nawrotem!\nI nie wiedzia\xc5\x82a
\xc5\x9blepa noc, kto tu jest cieniem, a kto m\xc5\x82otem?\n\n\xe2\x80\x9eO,
pr\xc4\x99dzej skruszmy zimny g\xc5\x82az, nim \xc5\x9bmier\xc4\x87
Dziewczyn\xc4\x99 rdz\xc4\x85 powlecze!\xe2\x80\x9d\nTak, wal\xc4\x85c w mur,
dwunasty cie\xc5\x84 do jedenastu innych rzecze.\n\nLecz cieniom zbrak\xc5\x82o
nagle si\xc5\x82, a cie\xc5\x84 si\xc4\x99 mrokom nie opiera!\nI
powymar\xc5\x82y jeszcze raz, bo nigdy do\xc5\x9b\xc4\x87 si\xc4\x99 nie
umiera\xe2\x80\xa6\n\nI nigdy do\xc5\x9b\xc4\x87, i nigdy tak, jak tego pragnie
\xc3\xb3w, co kona!\xe2\x80\xa6\nI znik\xc5\x82a tre\xc5\x9b\xc4\x87
\xe2\x80\x94 i zgin\xc4\x85\xc5\x82 \xc5\x9blad \xe2\x80\x94 i
powie\xc5\x9b\xc4\x87 o nich ju\xc5\xbc sko\xc5\x84czona!\n\nLecz dzielne
m\xc5\x82oty \xe2\x80\x94 Bo\xc5\xbce m\xc3\xb3j! \xe2\x80\x94 md\xc5\x82ej nie
podda\xc5\x82y si\xc4\x99 \xc5\xbca\xc5\x82obie!\nI same przez si\xc4\x99
bi\xc5\x82y w mur, hucza\xc5\x82y spi\xc5\xbcem same w sobie!\n\nHucza\xc5\x82y
w mrok, hucza\xc5\x82y w blask i ocieka\xc5\x82y ludzkim potem!\nI nie
wiedzia\xc5\x82a \xc5\x9blepa noc, czym bywa m\xc5\x82ot, gdy nie jest
m\xc5\x82otem?\n\n\xe2\x80\x9eO, pr\xc4\x99dzej skruszmy zimny g\xc5\x82az, nim
\xc5\x9bmier\xc4\x87 Dziewczyn\xc4\x99 rdz\xc4\x85 powlecze!\xe2\x80\x9d
\xe2\x80\x94\nTak, wal\xc4\x85c w mur, dwunasty m\xc5\x82ot do jedenastu innych
rzecze.\n\nI run\xc4\x85\xc5\x82 mur, tysi\xc4\x85cem ech
wstrz\xc4\x85saj\xc4\x85c wzg\xc3\xb3rza i doliny!\nLecz poza murem \xe2\x80\x94
nic i nic! Ni \xc5\xbcywej duszy, ni Dziewczyny!\n\nNiczyich oczu, ani ust! I
niczyjego w kwiatach losu!\nBo to by\xc5\x82 g\xc5\x82os i tylko \xe2\x80\x94
g\xc5\x82os, i nic nie by\xc5\x82o, opr\xc3\xb3cz g\xc5\x82osu!\n\nNic
\xe2\x80\x94 tylko p\xc5\x82acz i \xc5\xbcal i mrok i niewiadomo\xc5\x9b\xc4\x87
i zatrata!\nTaki\xc5\xbc to \xc5\x9bwiat! Niedobry \xc5\x9bwiat! Czemu\xc5\xbc
innego nie ma \xc5\x9bwiata?\n\nWobec k\xc5\x82amliwych jawnie sn\xc3\xb3w,
wobec zmarnia\xc5\x82ych w nico\xc5\x9b\xc4\x87
cud\xc3\xb3w.\nPot\xc4\x99\xc5\xbcne m\xc5\x82oty leg\xc5\x82y w rz\xc4\x85d na
znak spe\xc5\x82nionych godnie trud\xc3\xb3w.\n\nI by\xc5\x82a zgroza
nag\xc5\x82ych cisz! I by\xc5\x82a pr\xc3\xb3\xc5\xbcnia w ca\xc5\x82ym
niebie!\nA ty z tej pr\xc3\xb3\xc5\xbcni czemu drwisz, kiedy ta
pr\xc3\xb3\xc5\xbcnia nie drwi z ciebie?@'

[ ]:

[ ]: