1. One can try to defend against frequency analysis by first compressing the file to be encrypted. Note that a compression algorithm attempts to encode the information so that each byte in the compressed file is (almost) 1 information byte. So it looks promising – there are no "frequent symbols"

   However, it is not recommended to do it in this way. What could be the reason? Maybe you will have to look to some details of the compression schemes.

Problem 1: słowniki, nagłówki metadane...

Przykład: zawartość skompresowanego gzipem pliku (wejściowy plik przed kompresją miał nazwę "uncompressed")

```
00000000: 1f8b 0808 d0d9 2762 0003 756e 636f 6d70  ......'b..uncomp
00000010: 7265 7373 6564 00cb cb4c 4d4f acca 2f4f  ressed...LMO../O
00000020: ccab cc1b 6592 ccac aa2c cf4c 4d1e 0c2e  ....e....,.LM...
00000030: 1912 ccd1 e01a 0d8d 51e6 2873 9439 0498  ........Q.(s.9..
00000040: a3c5 d308 0818 0089 fca3 1900 0a00 00    ..............
```

Problem 2: Co jeśli dla pewnych algorytmów kompresji i szyfrowania da się zdekompresować zaszyfrowany plik?

Problem 3: Kompresja zdradza entropię plaintextu przez długość po kompresji.

Ciąg "AAAAAAAA" (mniejsza entropia) skompresuje się lepiej niż ciąg "XA7FD5HK" (większa entropia).

Przykład (gzip):

```
$ echo 'XA7FD5HK' | gzip | xxd
00000000: 1f8b 0800 0000 0000 0003 8b70 3477 7331  ...........p4ws1
00000010: f5f0 e602 00ac 1226 9909 0000 00         .......&.....
$ echo 'AAAAAAAA' | gzip | xxd
00000000: 1f8b 0800 0000 0000 0003 7374 8400 2e00  ..........st....
00000010: 9158 7bd2 0900 0000                      .X{.....
```

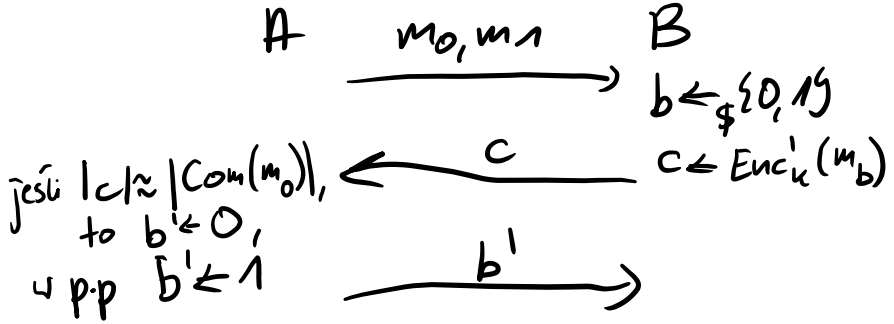
Mając algorytm szyfrowania $Enc_k(\cdot)$ i algorytm kompresji $Com(\cdot)$ zdefiniujmy algorytm szyfrowania $Enc'_k(m)$:

$x \leftarrow Com(m)$

return $Enc_k(x)$.

Gra bezpieczeństwa (semantic security):

Niech entropia m_0 będzie większa niż entropia m_1.



$b = b'$ z dużym prawdopodobieństwem

Ta właściwość została wykorzystana w atakach CRIME i BREACH na protokół SSL.

2. An alternative definition for perfect security for symmetric encryption schemes has been proposed: *perfect security B* means that *each ciphertext is equally probable for a given plaintext and key chosen at random*. Are the notions of *perfect security* and *perfect security B* equivalent?

Perfect security:

$P[c|m_1] = P[m|m_2]$

Perfect security B:

$P[c_1|m] = P[c_2|m]$

Weźmy $OTP_k(\cdot)$

$Enc_k(m)$:

$s = OTP_k(m)$

$b \leftarrow \begin{cases} 0 & z \text{ ppb } \frac{1}{4} \\ 1 & z \text{ ppb } \frac{3}{4} \end{cases}$

return $s \| b$

$P[s\|0 \,|\, m] = \frac{1}{4} \cdot P[s|m]$

$\neq$

$P[s\|1 \,|\, m] = \frac{3}{4} \cdot P[s|m]$

, zatem definicje nie są równoważne.

Oznaczenie:

"a || b" to konkatenacja a i b

3. Find an example of an encryption scheme with perfect security that is different from one-time pad.

a) Can it happen that the key length is higher than the length of the ciphertext (we mean schemes where all bits of the key have influence on the encryption outcome)?

b) What is the maximal length of the encryption keys so that no two keys would yield always the same results?

a) Pomysł 1 (Shamir's Secret Sharing):

Klucz: n punktów y_i $(k = \{y_1, y_2 \dots, y_n\})$

Plaintext: m

$Enc_k(m)$: $f(x) :=$ wielomian stopnia n o punktach $(0, m), (1, y_1), (2, y_2), \dots, (n, y_n)$

return $f(n+1)$.

$Dec_k(c)$: $f'(x) :=$ wielomian stopnia n o punktach $(1, y_1), (2, y_2), \dots, (n, y_n), (n+1, c)$

return $f'(0)$

Długość klucza: n wartości

Długość ciphertextu: 1 wartość

Pomysł 2:

Niech $m \in \mathcal{M}$, $c \in \mathcal{C}$, $|m| = |c| = l$

$k = k_1 \| k_2$, gdzie k_2 definiuje losową permutację l-elementową σ_{k_2} i $|k_1| = l$.

$Enc_k(m)$:

$s = OTP_{k_1}(m)$

return $\sigma_{k_2}(s)$

---

b) Załóżmy, że $|\mathcal{M}| = |\mathcal{C}| = 2^l$ (czyli $Enc_k(\cdot) : \{0,1\}^l \to \{0,1\}^l$)

ile jest bijekcji między $\mathcal{M}$ i $\mathcal{C}$?

Wszystkich permutacji jest $2^l!$

Maksymalna długość klucza: $\lceil \log_2(2^l!) \rceil$

Pytanie otwarte: co jeśli $|\mathcal{M}| < |\mathcal{C}|$?

4. Recall the definition of *semantic security*.

a) • show that *perfect security* implies *semantic security*.

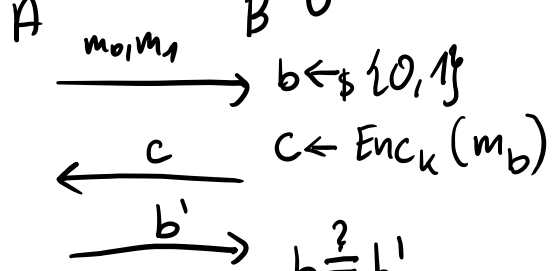
b) • does *semantic security* imply *perfect security*? Find a convincing argument or an example.

c) • Semantic security may be formulated in terms of 4 plaintexts: Alice chooses m_1, m_2, m_3, m_4, encrypts m_b and Bob has to guess b. Is this notion stronger?

Perfect security:

$P[c|m_0] = P[c|m_1]$

Semantic security:



$P[b=b'] = \frac{1}{2} + negl(\cdot)$

a) $P[c|m_0] = P[c|m_1] \Rightarrow P[b=b'] = \frac{1}{2} + 0$ (TAK)

b) (NIE)

Przykład:

$\mathcal{M} = \{0,1\} = \mathcal{C}$

Wtedy $Enc_k(\cdot)$ takie, że

$P[c=0 \mid M=0] = \frac{1}{2} + \varepsilon$

$P[c=0 \mid M=1] = \frac{1}{2} - \varepsilon$ ✗

$P[c=1 \mid M=0] = \frac{1}{2} - \varepsilon$

$P[c=1 \mid M=1] = \frac{1}{2} + \varepsilon$ ✗

gdzie ε - zaniedbywalne, $\varepsilon > 0$.

$Enc_k(\cdot)$ nie jest perfectly secure.

Niech $m_0 = 0$, $m_1 = 1$

Wtedy $M = b$ i $b' = C$

$P[b=b'] = P[M=C] =$

$= P[M=0] \cdot P[C=0|M=0] + P[M=1] \cdot P[C=1|M=1]$

$= \frac{1}{2}(\frac{1}{2}+\varepsilon) + \frac{1}{2}(\frac{1}{2}+\varepsilon) = \frac{1}{2} + \varepsilon$,

więc $Enc_k(\cdot)$ jest semantically secure.

c) A m_0, m_1, m_2, m_3 B

$b \leftarrow_\$ \{0,1,2,3\}$

$c \leftarrow Enc_k(m_b)$

$m_0, m_1, m_2, m_3, c \rightarrow$

$\leftarrow b'$

(gra I)

sprawdzamy $b = b'$

$P[b = b'] = \frac{1}{4} + negl(\cdot)$

To jest słabsze od:

A B

$m_0, m_1, m_2, m_3 \rightarrow$

$b \leftarrow_\$ \{0,1,2,3\}$

$c \leftarrow Enc_k(m_b)$ (gra II)

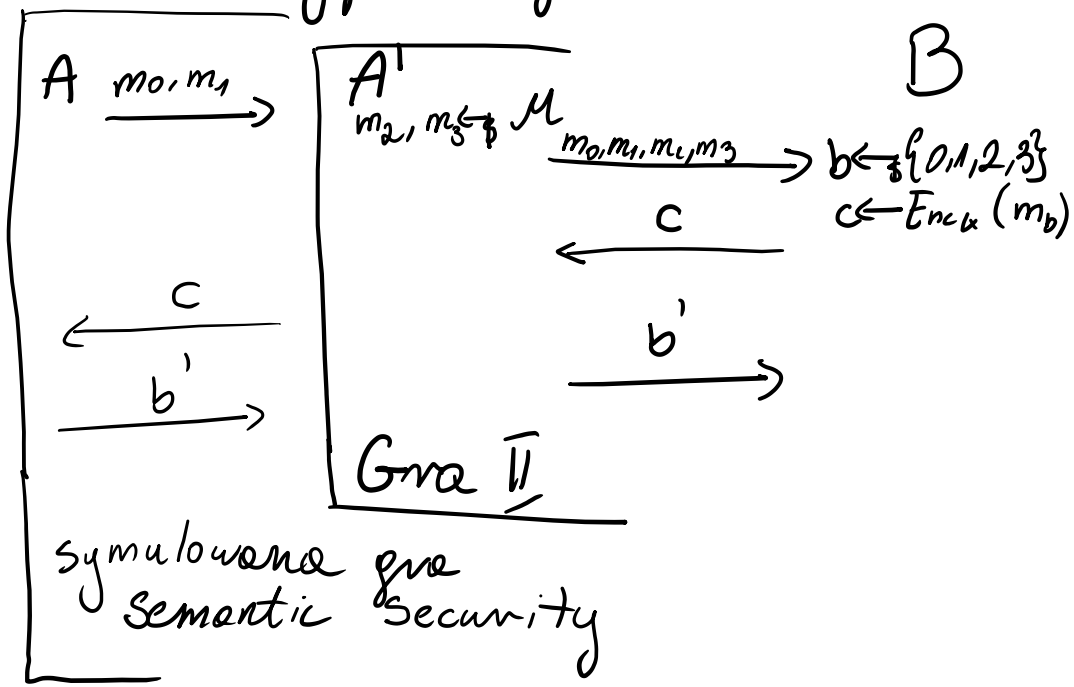
$\leftarrow c$

$b' \rightarrow$

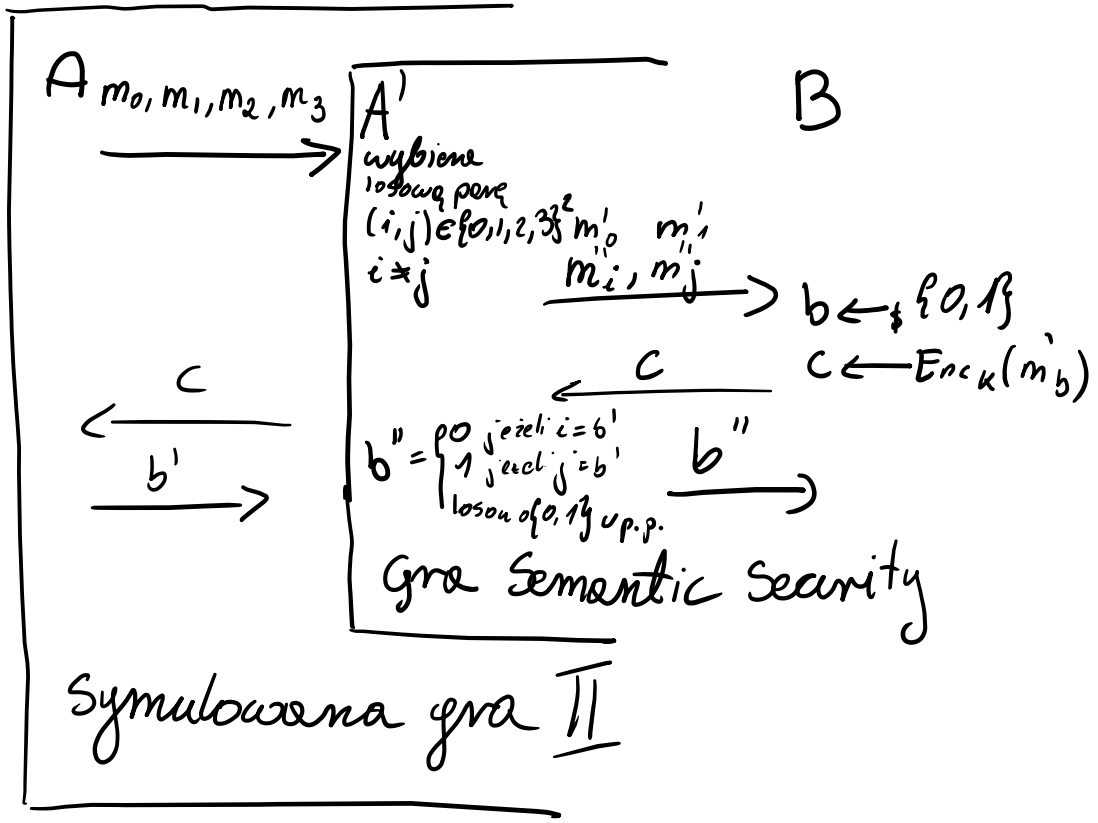
sprawdzamy $b = b'$

Adwersarz, który sam może wybierać m_i ma większą siłę niż ten, który nie może, por. zadanie 1

Czy mając adwersarza 2 gry semantic security można "wygrać" grę II (i na odwrót)?



Z ppb. 1/2 $b \in \{0,1\}$, więc $P[b = b'] = \frac{1}{2} \cdot \left(\frac{1}{2} + \varepsilon\right)$, gdzie ε to przewaga adwersarza 2 gry semantic security



Intuicja: dla Adwersarza z gry II ta symulowana sytuacja jest identyczna do prawdziwej gry II, więc jego przewaga pozwala wprost rozwiązać problem z gry Semantic Security.

Pytanie otwarte: jaka jest przewaga ε' w $P[b'' = b] = \frac{1}{2} + \varepsilon'$, gdy $P[b' = \tilde{b}] = \frac{1}{4} + \varepsilon$

dla $\tilde{b} = \begin{cases} i & \text{gdy } b = 0 \\ j & \text{gdy } b = 1 \end{cases}$

6. How to play Monopoly using only email for communication? (Of course, everybody can cheat)

Mamy n graczy. i oznacza numer gracza.

Rzut kością (6-ścienną)

1. Każdy generuje odpowiednio dużą* i losową liczbę X_i.
2. Każdy liczy $y_i = F(x_i)$, gdzie F jest funkcją jednokierunkową.
3. Wysyłamy y_i do wszystkich graczy. (commitment)
4. Czekamy, aż odbierzemy wszystkie y_j.
5. Wysyłamy x_i do wszystkich graczy. (opening)
6. Sprawdzamy, czy $\forall_j \; y_j = F(x_j)$.
7. Liczymy $X = \sum_i x_i$, wynik rzutu kością to $X \bmod 6 + 1$.

* taką, żeby nie dało się odnaleźć przeciwobrazu $F(x_i)$ metodą brute force.

5. [homework] We have to find a key K that has been used to obtain a ciphertext C from a plaintext T. We assume that there exists exactly one such a key and that each key consists of k bits. Assume that encryption rate is 10^6 ciphertexts/second. Estimate the time effort required for finding key K by a brute force attack, that is, checking the possible keys one by one. Answer this question for $k = 40, 56, 90, 128, 256$.

   Check encryption speed of AES e.g. on your laptop, estimate the energy cost (assume you have to pay 1PLN/kWh) according to the nominal (real, if you can) power consumption.

Kilka zdań raportu + wyniki (tekstowo)

Benchmark AES-128 1 blok = 16 bajtów

Nie trzeba implementować AESa samodzielnie, można wziąć gotowe narzędzia lub biblioteki (np. OpenSSL)

Można założyć, że czas sprawdzenia jednego klucza to w przybliżeniu czas zaszyfrowania jednego bloku.

Zużycie energii oszacować, np. z dokumentacji CPU / z mocy zasilacza