Politechnika Wrocławska, Informatyka Stosowana

BLOKOWE ALGORYTMY SZYFROWANIA

Cyberbezpieczeństwo, Laboratorium nr.3 - raport

Autor: Aleksander Stepaniuk
Nr. Indeksu: 272644

Zad 1. Przykładowe algorytmy blokowe.

Teksty których użyłem do analizy kolejnych algorytmów:

Tekst 1:

Litera „n” powtórzona 2000 razy.

Tekst 2:

Tekst „i potem jeszcze do biedronki na lody” powtórzony 1000 razy.

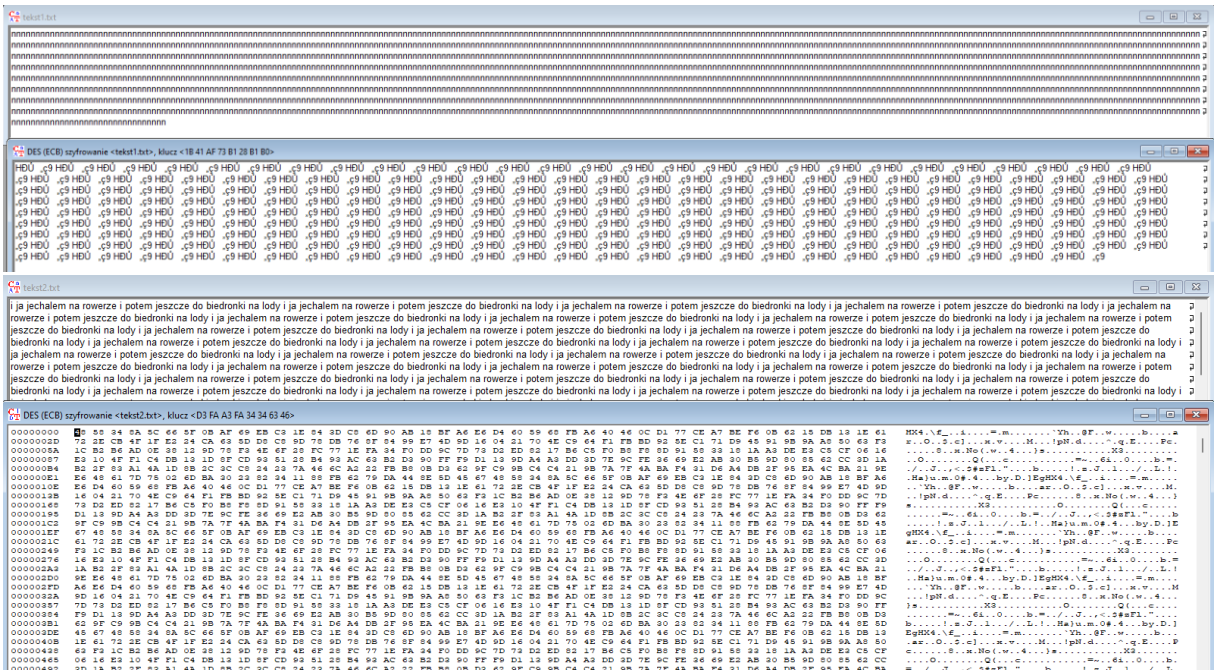
Tekst 3:

5000 pierwszych znaków pana Tadeusza („litwo ojczyzno moja ty jestes jak zdrowie ile cie trzeba cenic ten tylko sie dowie kto cie stracil…”)

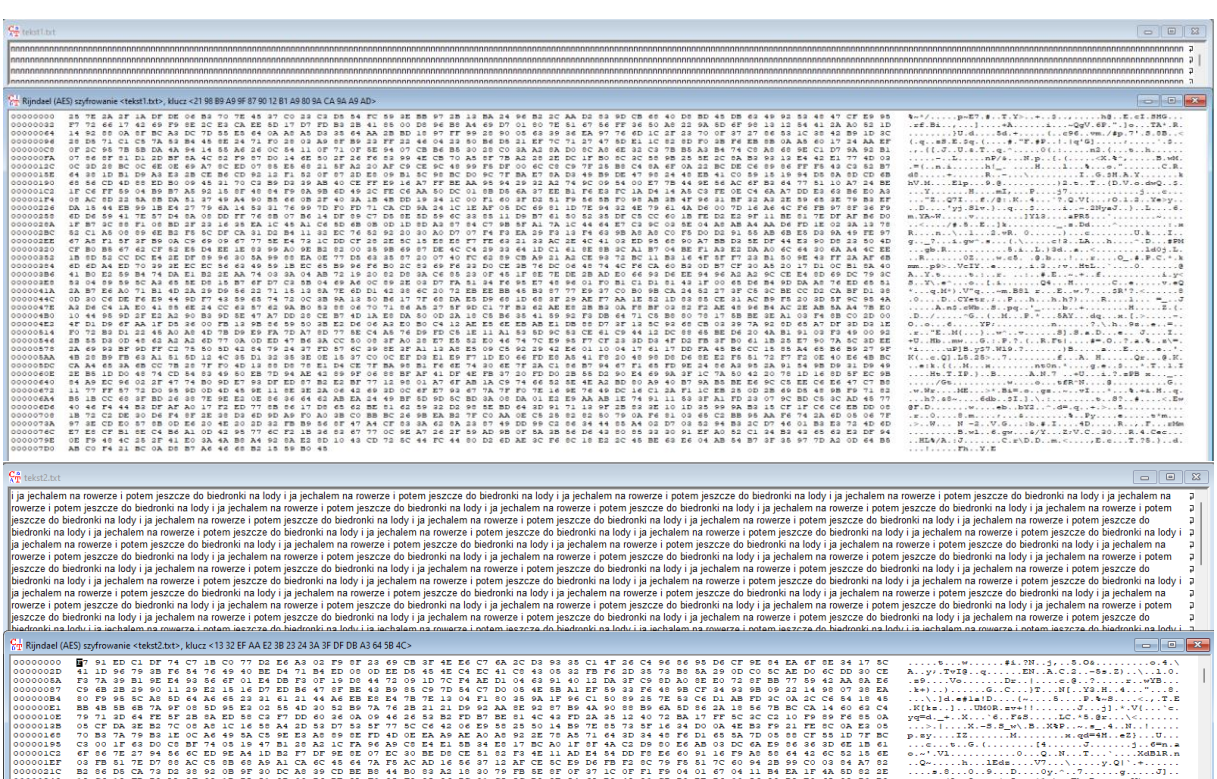
Analizie poddałem następujące algorytmy: IDEA, AES, DES

IDEA:

DES (z ECB):



AES (z CBC): (128 bitów klucz)

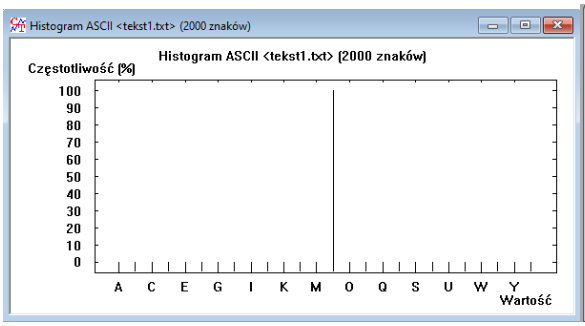


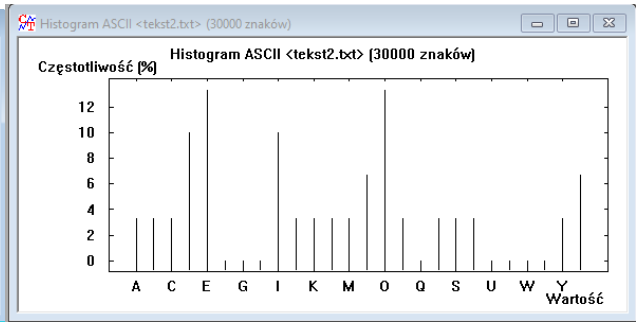
Zadanie 1.1;

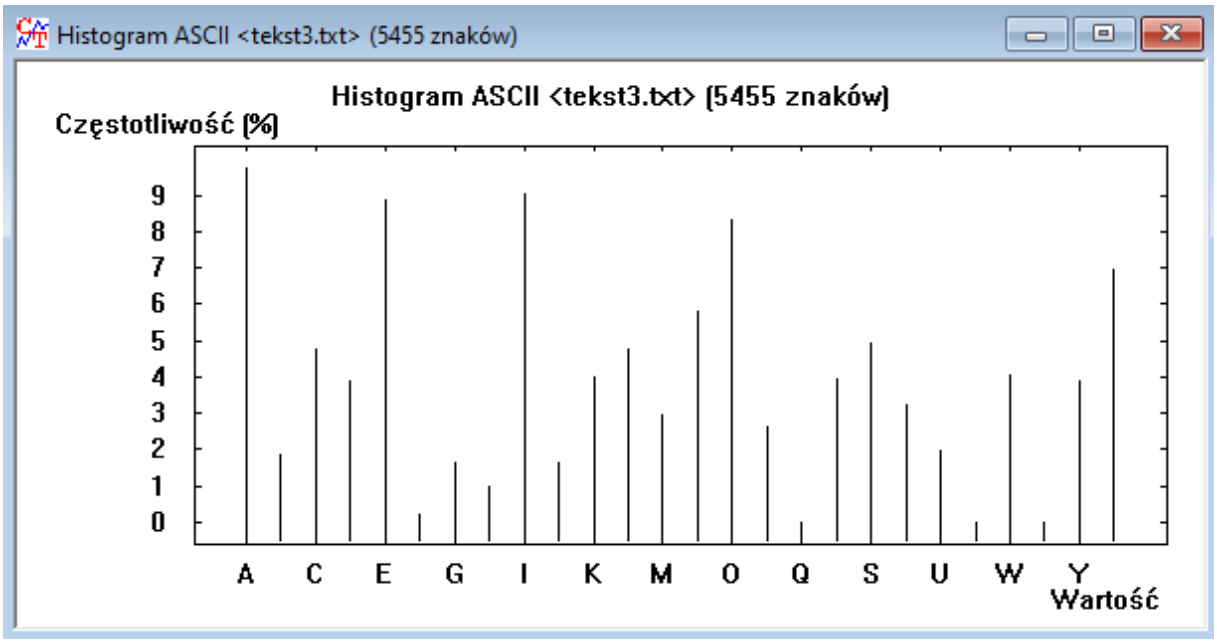
W tabeli poniżej znormalizowano entropię wszystkich tekstów w taki sposób, aby maksymalna możliwa wartość wynosiła 4,70.

Entropia tekstów zależnie od algorytmu		
Algorytm	Tekst jawny	Tekst zaszyfrowany
IDEA	0,00	1,78
	3,92	4,19
	4,25	4,67
DES (ECB)	0,00	1,00
	3,92	4,24
	4,25	4,68
AES (CBC 128 bit klucz)	0,00	4,64
	3,92	4,69
	4,25	4,67

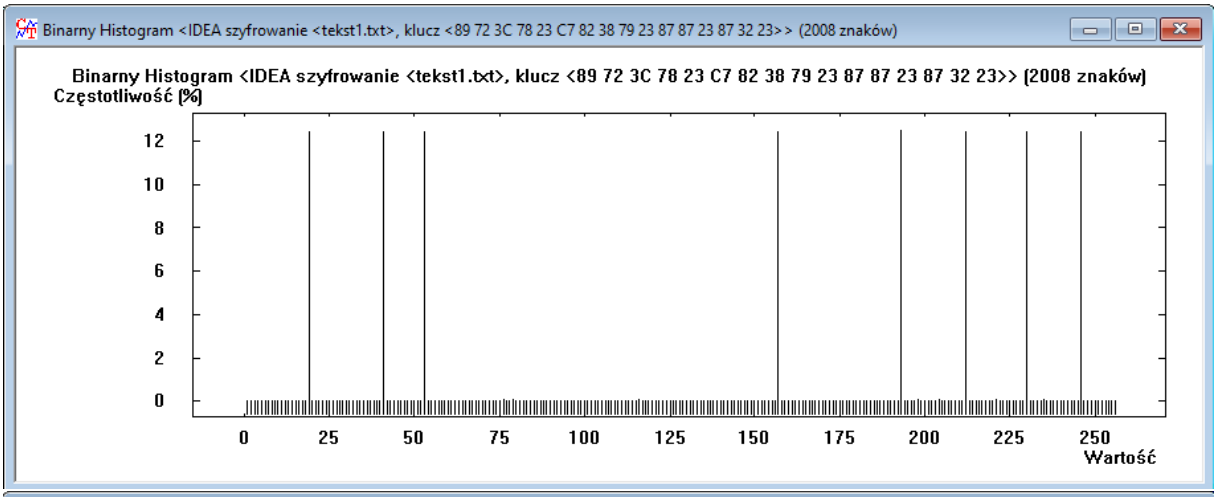
Histogramy dla tekstów jawnych:

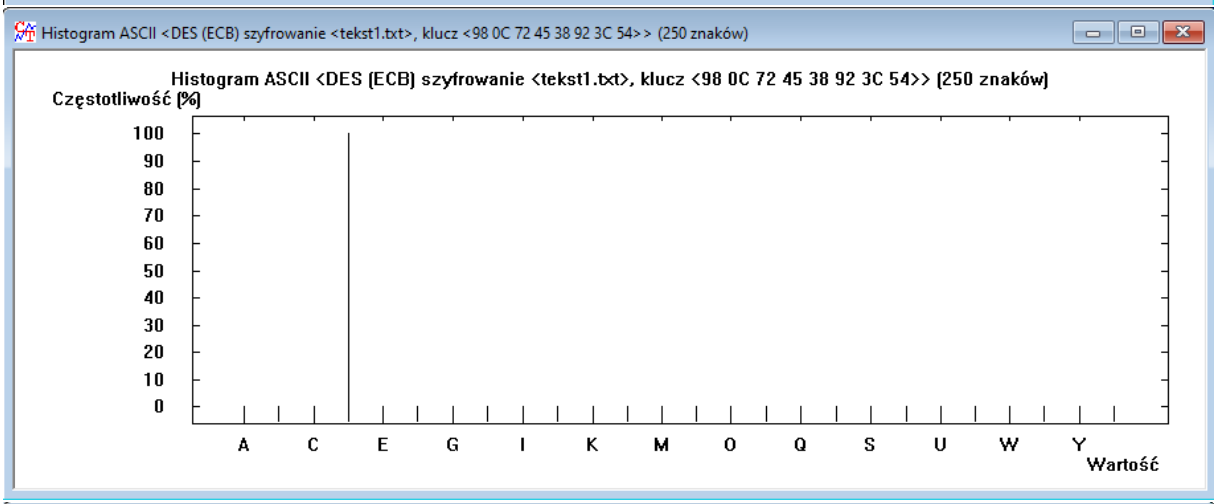


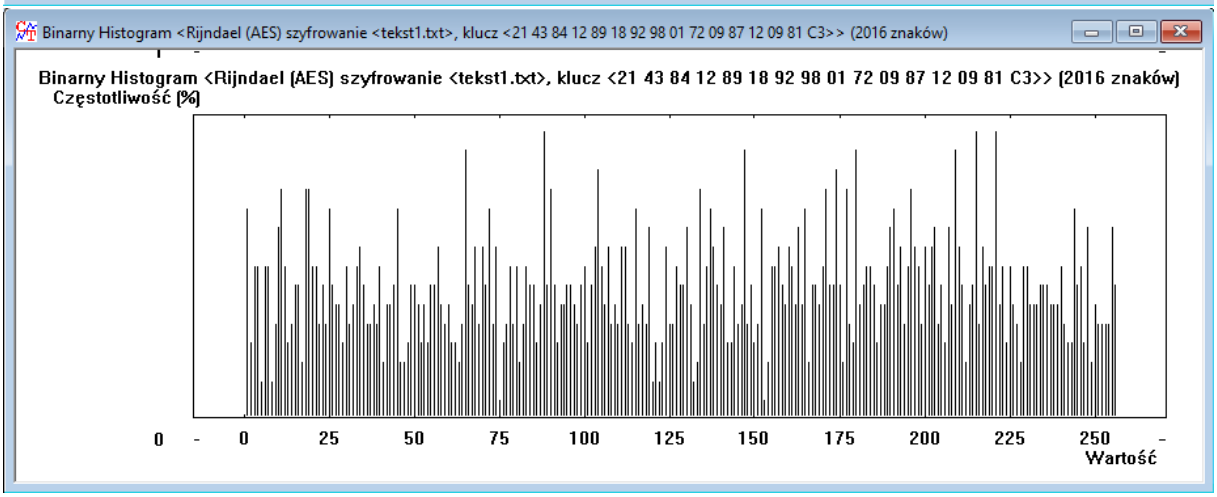




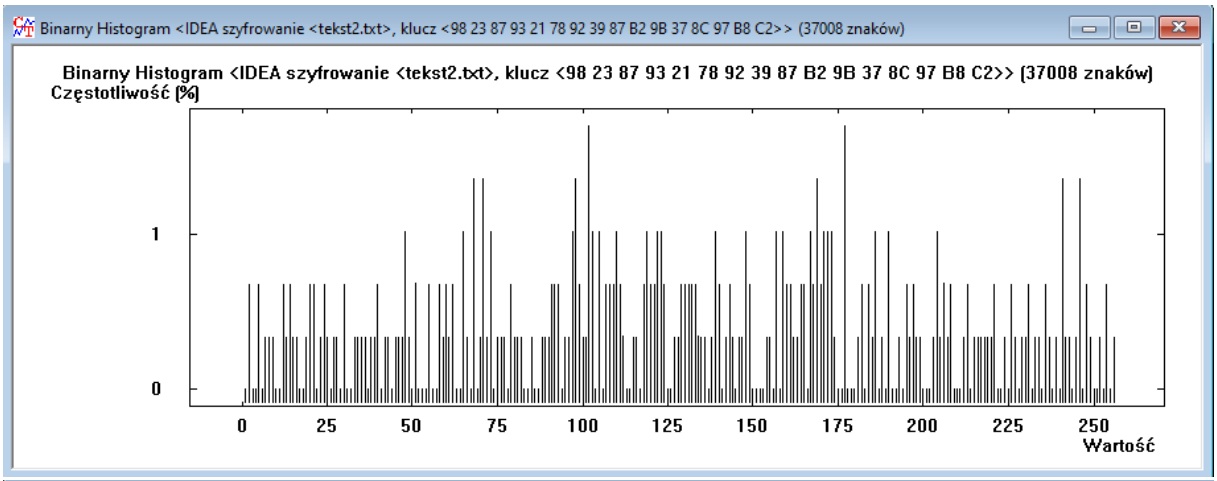
Histogramy dla tekstów tajnych z tekstu 1: (kolejno IDEA, DES (ECB), AES (CBC)):

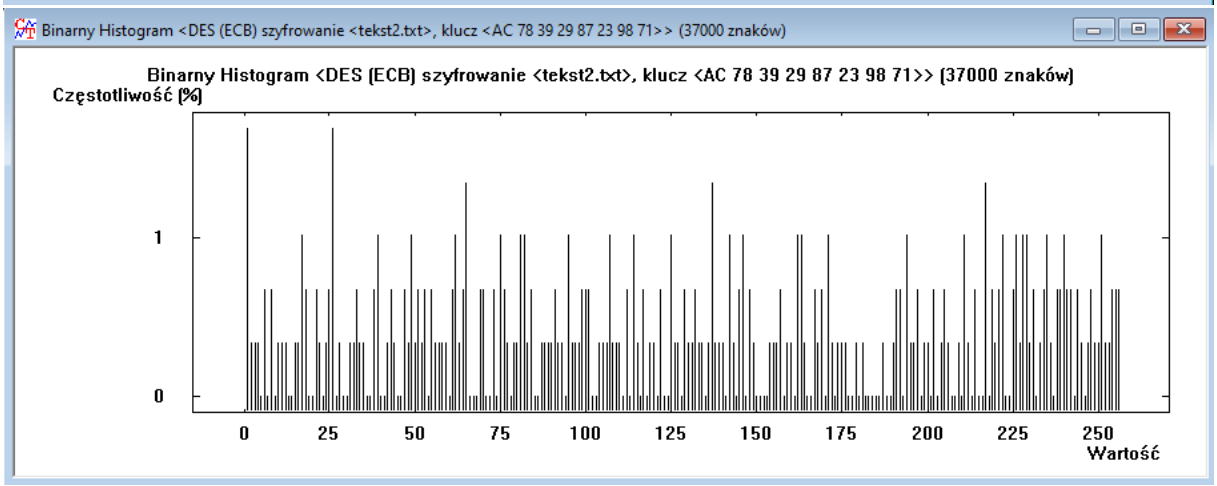


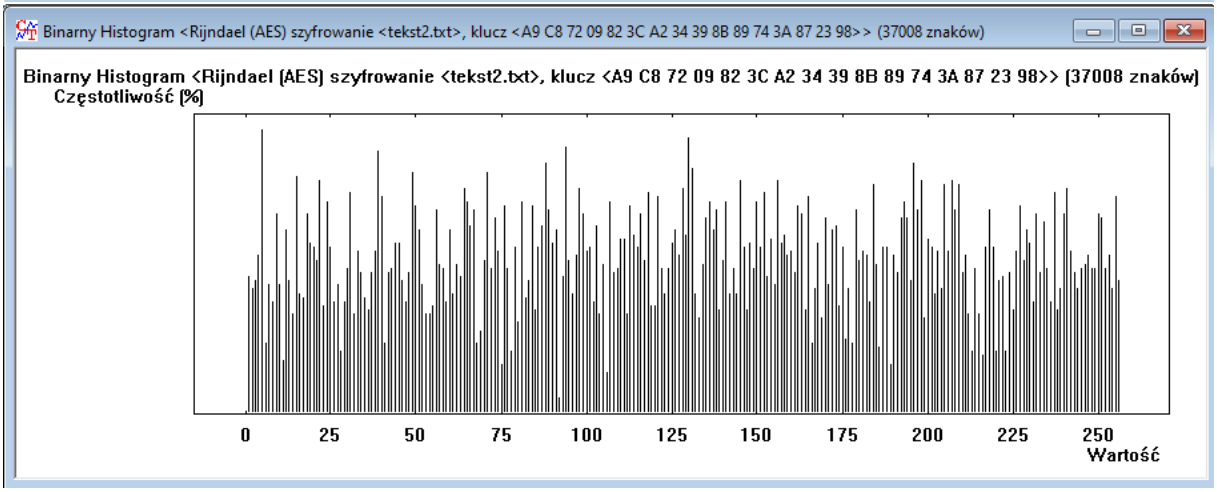




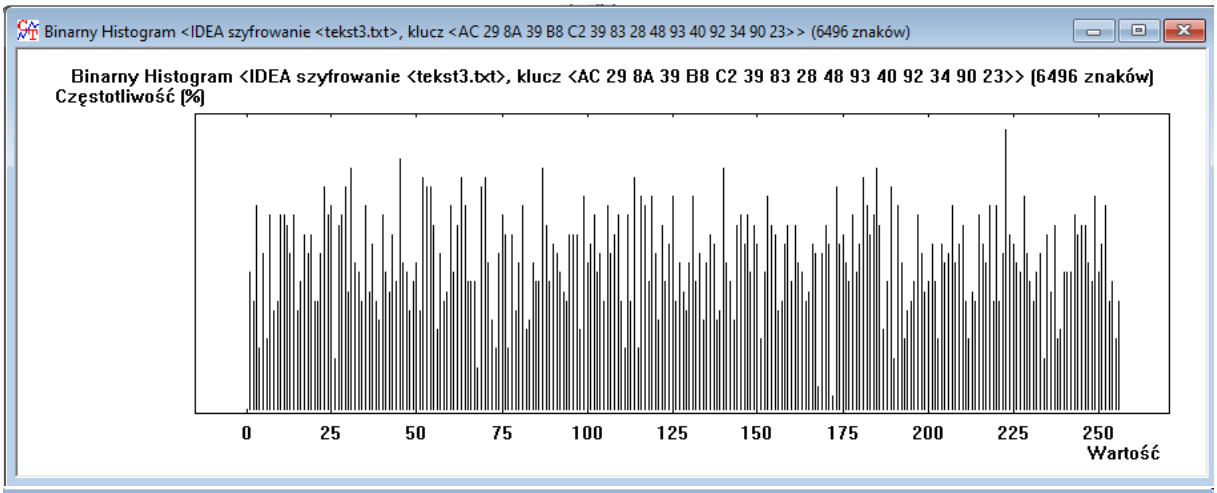
Histogramy dla tekstów tajnych z tekstu 2: (kolejno IDEA, DES (ECB), AES (CBC)):

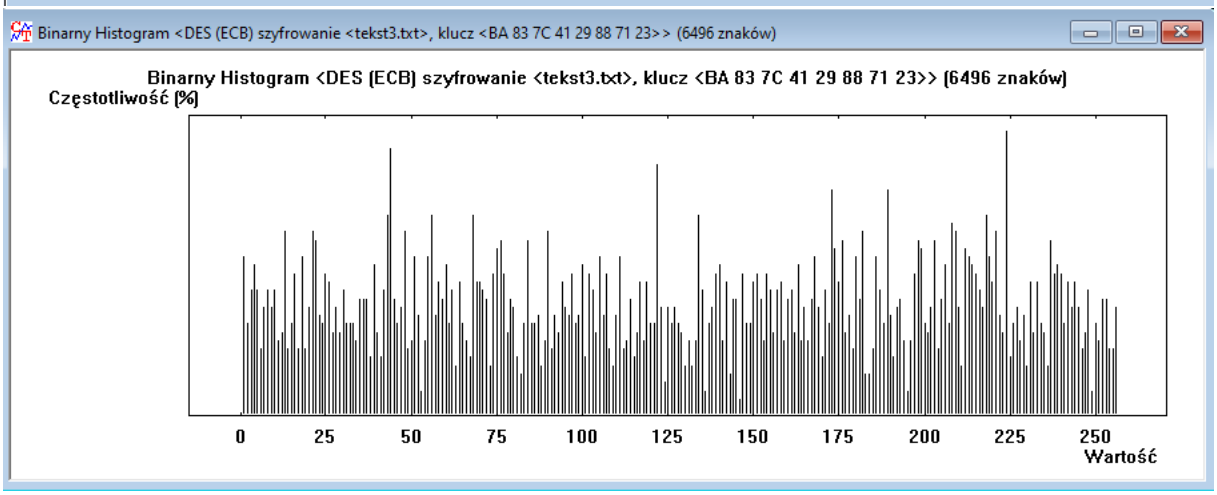


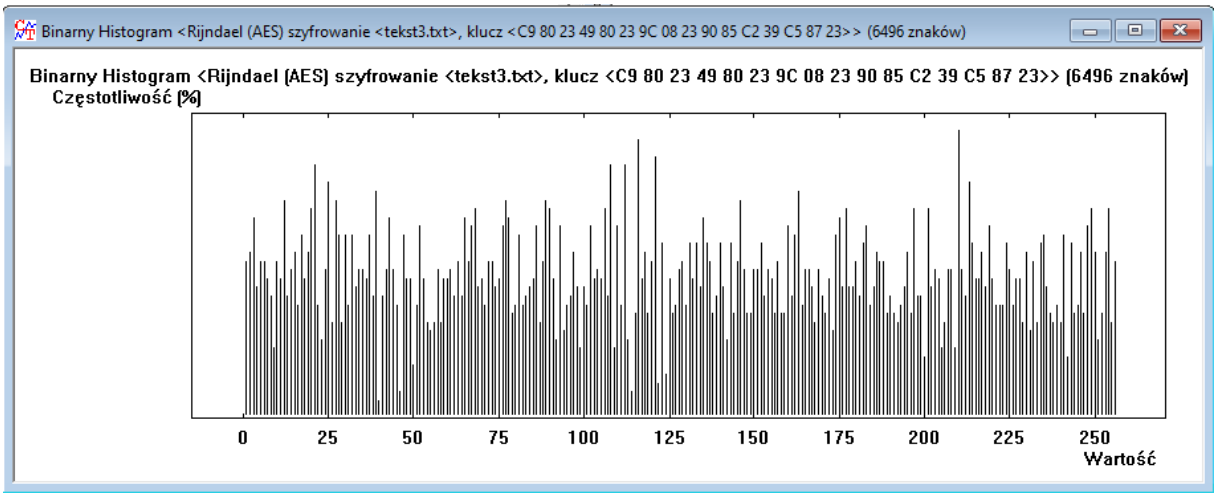




Histogramy dla tekstów tajnych z tekstu 3: (kolejno IDEA, DES (ECB), AES (CBC)):







Zadanie 1.2;

Długości kluczy dla kolejnych algorytmów:

IDEA:

K1 – 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

K2 – 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78

K3 – AC C9 49 08 3C 09 90 87 DF 49 80 7B F0 32 03 29

DES (ECB):

K4 – 00 00 00 00 00 00 00 00

K5 – 12 34 12 34 12 34 12 34

K6 – AC C9 49 08 3C 09 90 87

AES (CBC):

K7 – 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

K8 –

11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11

K9 –

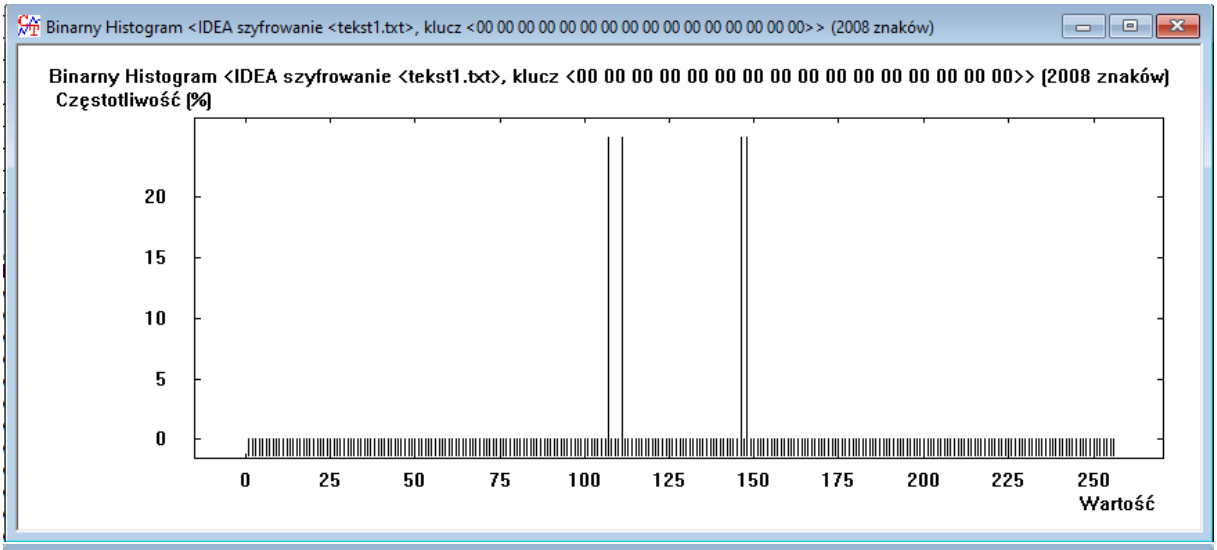
AC C9 49 08 3C 09 90 87 DF 39 A9 BC 10 02 05 8B CC 21 99 12 87 B9 F0 71 7A 4B C0 45 3B 89 6B A8

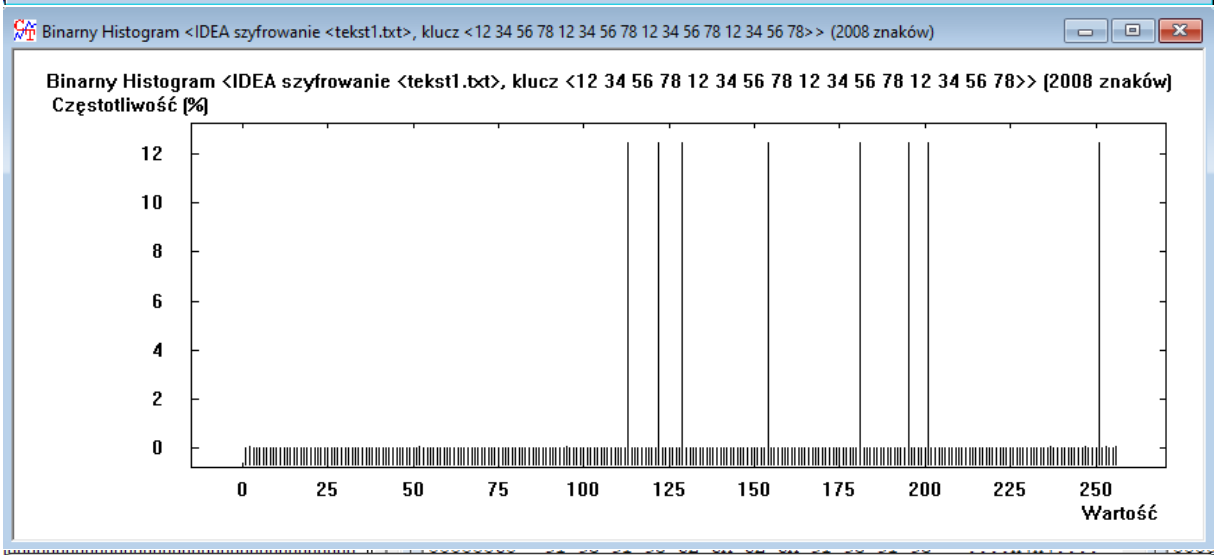
W tabeli poniżej znormalizowano entropię wszystkich tekstów w taki sposób, aby maksymalna możliwa wartość wynosiła 4,70.

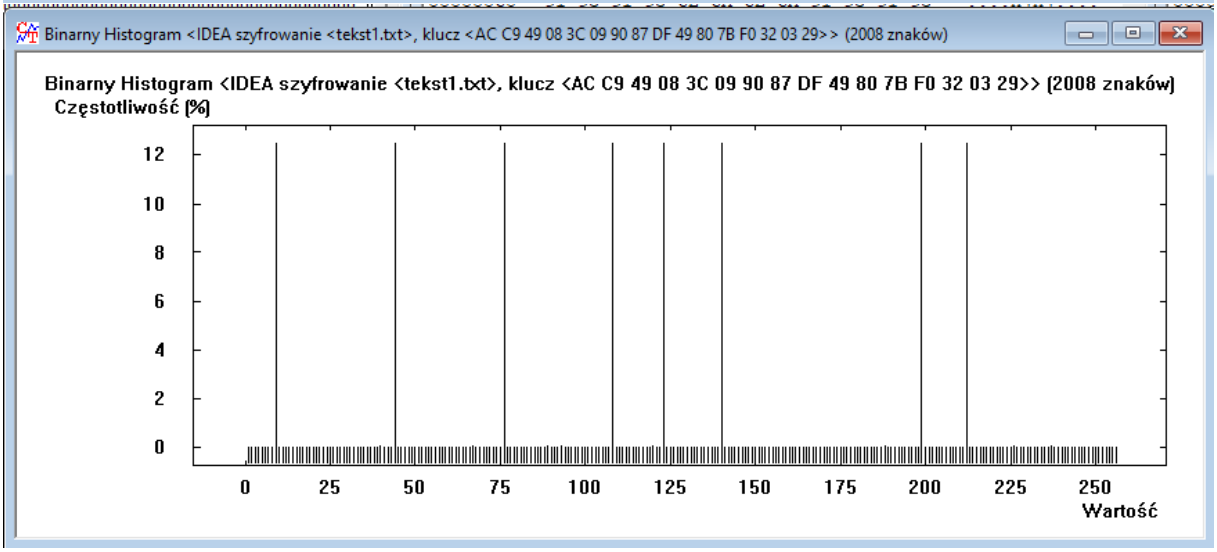
Entropia TT oraz TJ zależnie od szyfru, długości i wartości klucza										
Szyfr		**IDEA**			**DES (ECB)**			**AES (CBC)**		
Klucz		K1	K2	K3	K4	K5	K6	K7	K8	K9
tekst nr.	**TJ**	**TT**								
Tekst1	0,00	1.19	1,78	1,78	1.76	0,00	1.76	4,64	4,64	4,64
Tekst2	3,92	4.16	4,27	4,29	4.23	4.31	4.26	4,69	4,69	4,69
Tekst3	4,25	4,51	4,67	4,68	4.68	4.68	4.67	4,68	4,68	4,68

Histogramy:

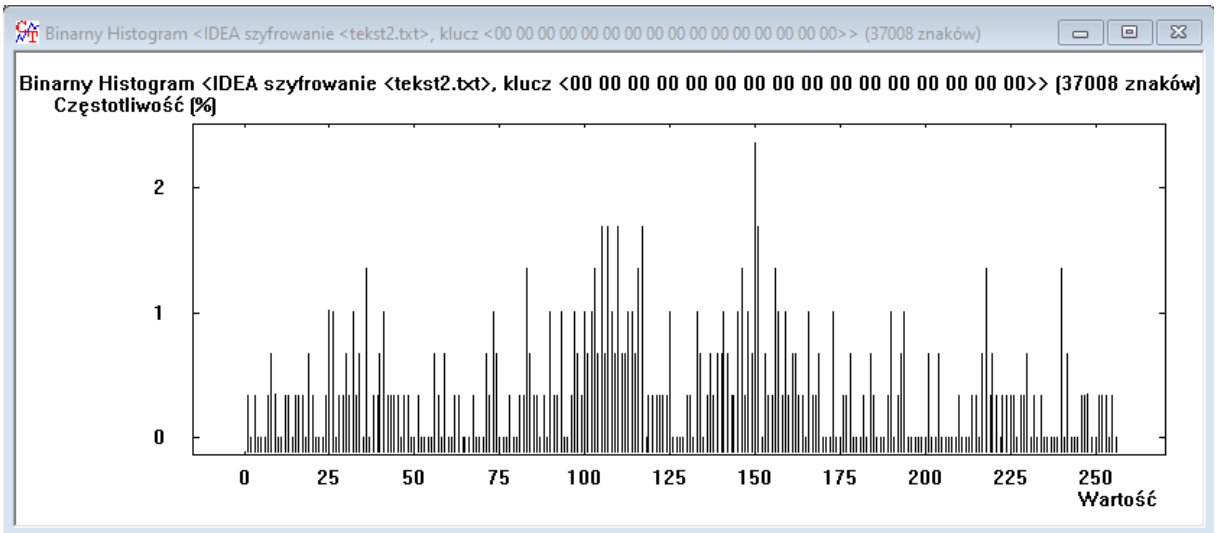
IDEA, tekst 1:

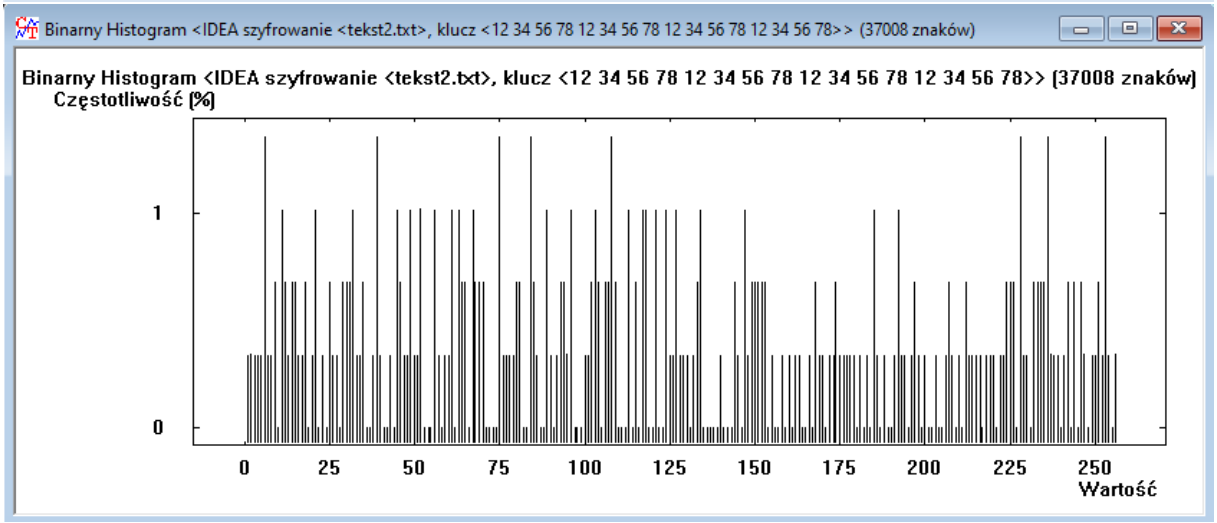


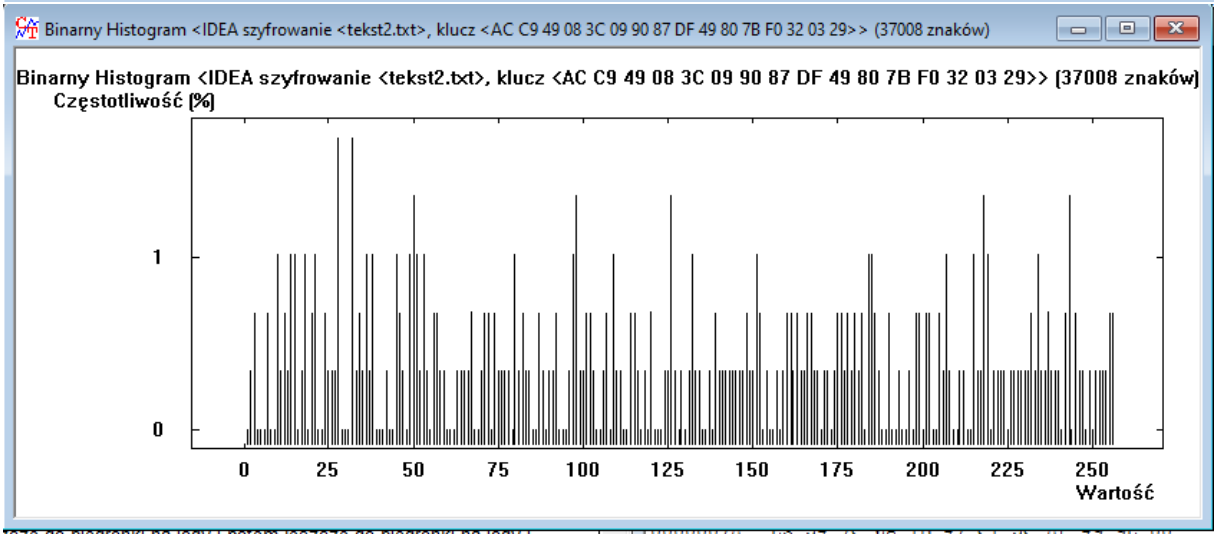




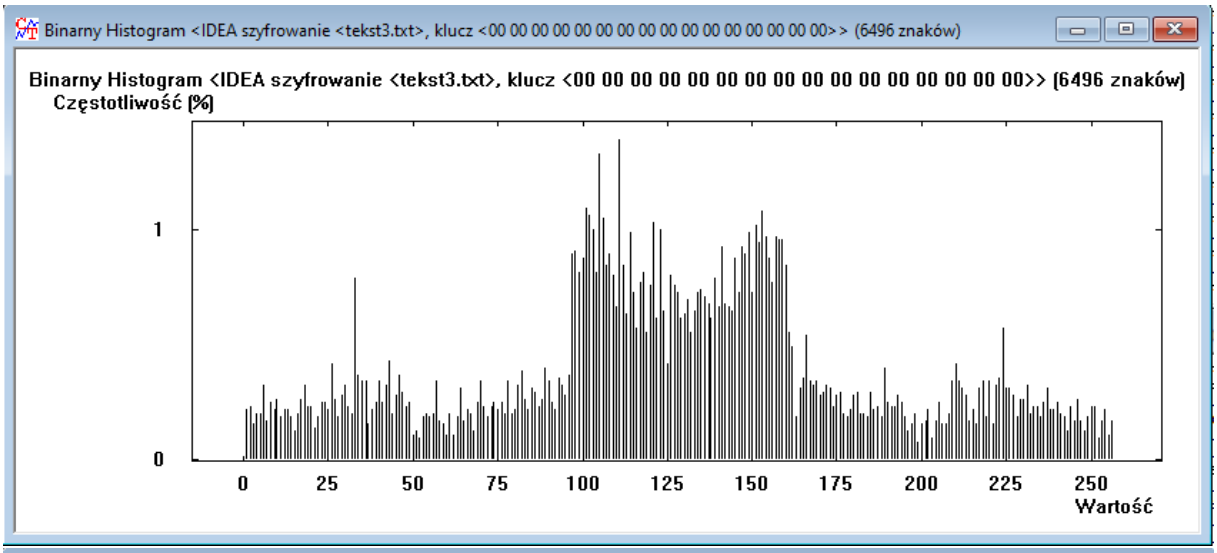
IDEA, tekst 2:

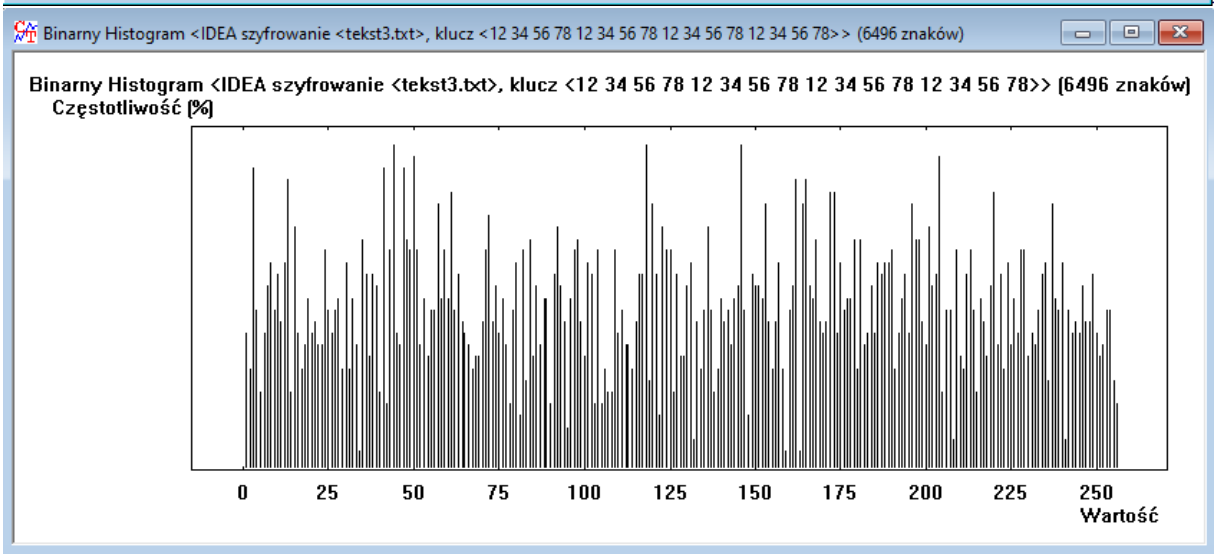

Binarny Histogram <IDEA szyfrowanie <tekst2.txt>, klucz <00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>> (37008 znaków)
Binarny Histogram <IDEA szyfrowanie <tekst2.txt>, klucz <00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>> (37008 znaków)
Częstotliwość [%]
2
1
0
0 25 50 75 100 125 150 175 200 225 250
Wartość

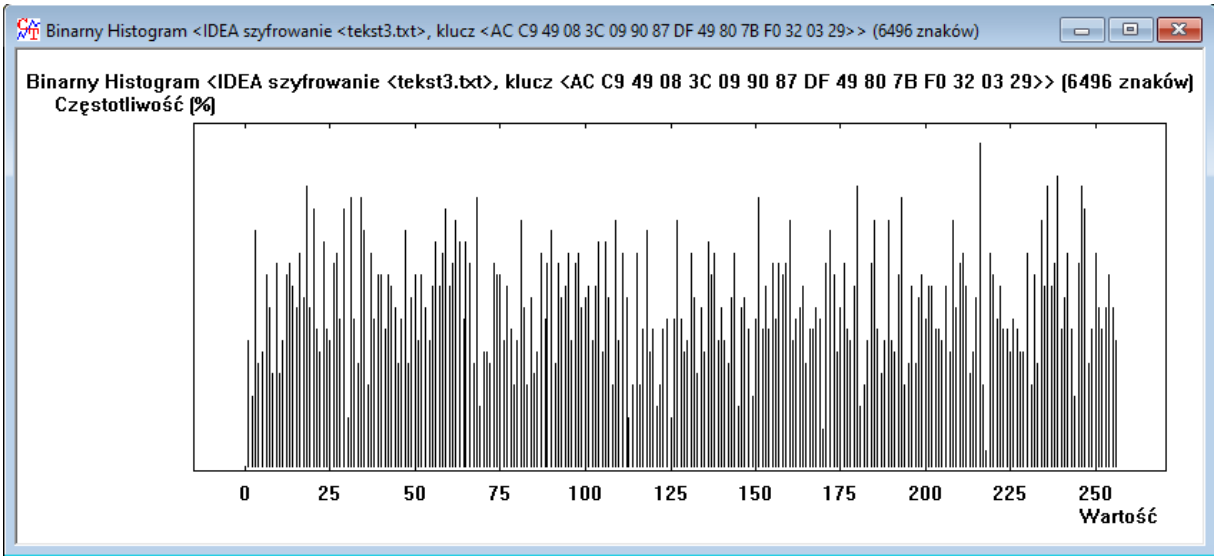

Binarny Histogram <IDEA szyfrowanie <tekst2.txt>, klucz <12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78>> (37008 znaków)
Binarny Histogram <IDEA szyfrowanie <tekst2.txt>, klucz <12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78>> (37008 znaków)
Częstotliwość [%]
1
0
0 25 50 75 100 125 150 175 200 225 250
Wartość


Binarny Histogram <IDEA szyfrowanie <tekst2.txt>, klucz <AC C9 49 08 3C 09 90 87 DF 49 80 7B F0 32 03 29>> (37008 znaków)
Binarny Histogram <IDEA szyfrowanie <tekst2.txt>, klucz <AC C9 49 08 3C 09 90 87 DF 49 80 7B F0 32 03 29>> (37008 znaków)
Częstotliwość [%]
1
0
0 25 50 75 100 125 150 175 200 225 250
Wartość

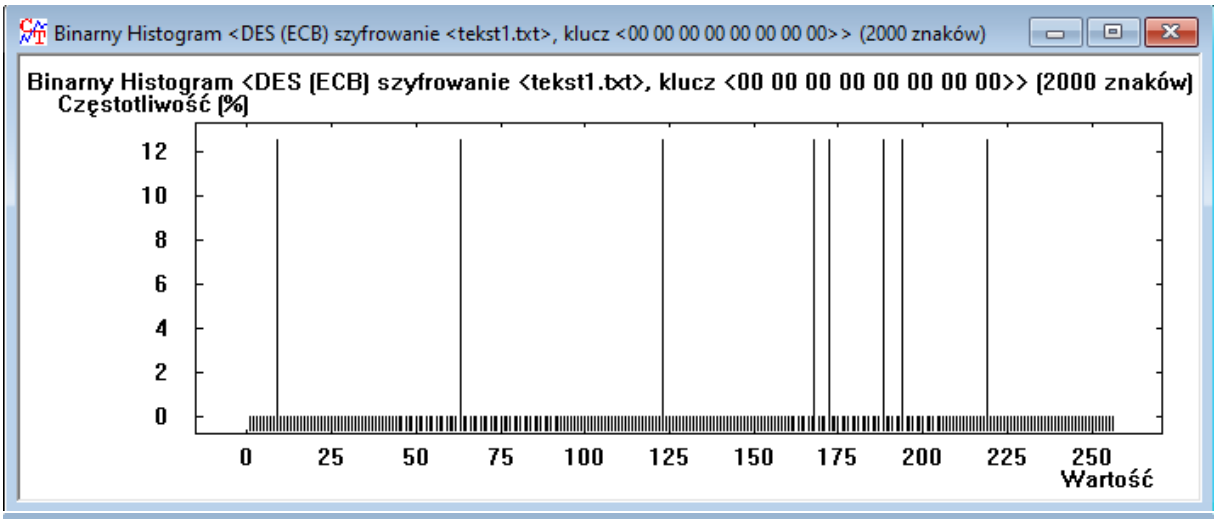
IDEA, tekst 3:

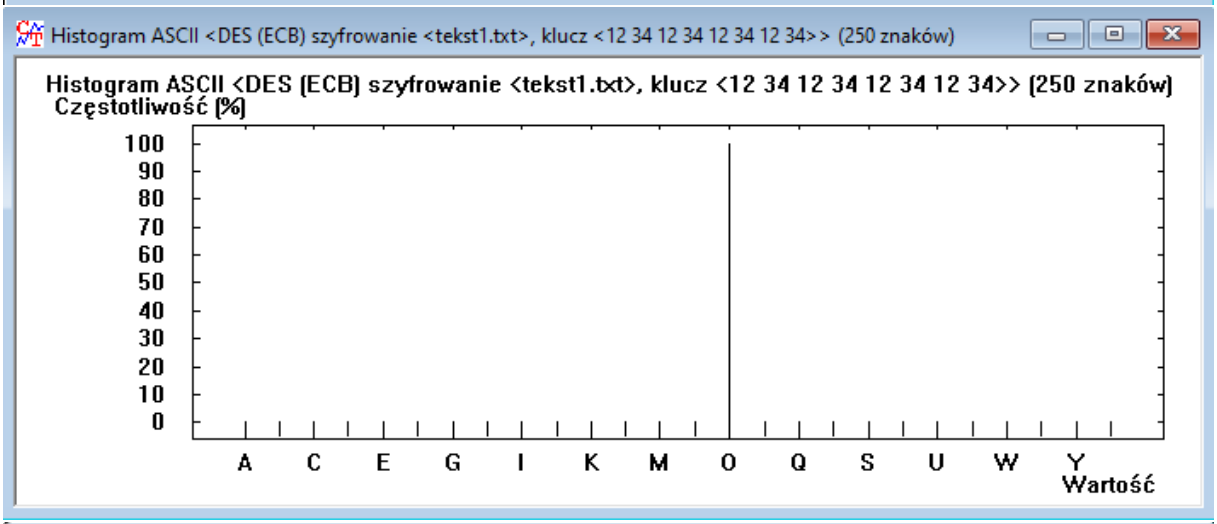

Binarny Histogram <IDEA szyfrowanie <tekst3.txt>, klucz <00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>> (6496 znaków)
Binarny Histogram <IDEA szyfrowanie <tekst3.txt>, klucz <00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>> (6496 znaków)
Częstotliwość [%]
1
0
0 25 50 75 100 125 150 175 200 225 250
Wartość

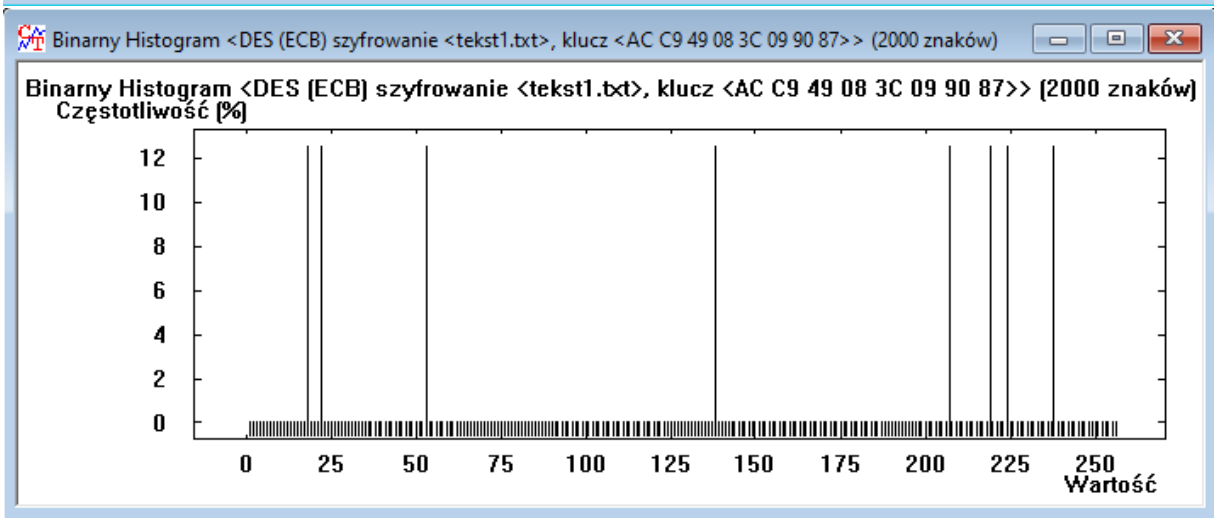

Binarny Histogram <IDEA szyfrowanie <tekst3.txt>, klucz <12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78>> (6496 znaków)
Binarny Histogram <IDEA szyfrowanie <tekst3.txt>, klucz <12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78>> (6496 znaków)
Częstotliwość [%]
0 25 50 75 100 125 150 175 200 225 250
Wartość


Binarny Histogram <IDEA szyfrowanie <tekst3.txt>, klucz <AC C9 49 08 3C 09 90 87 DF 49 80 7B F0 32 03 29>> (6496 znaków)
Binarny Histogram <IDEA szyfrowanie <tekst3.txt>, klucz <AC C9 49 08 3C 09 90 87 DF 49 80 7B F0 32 03 29>> (6496 znaków)
Częstotliwość [%]
0 25 50 75 100 125 150 175 200 225 250
Wartość

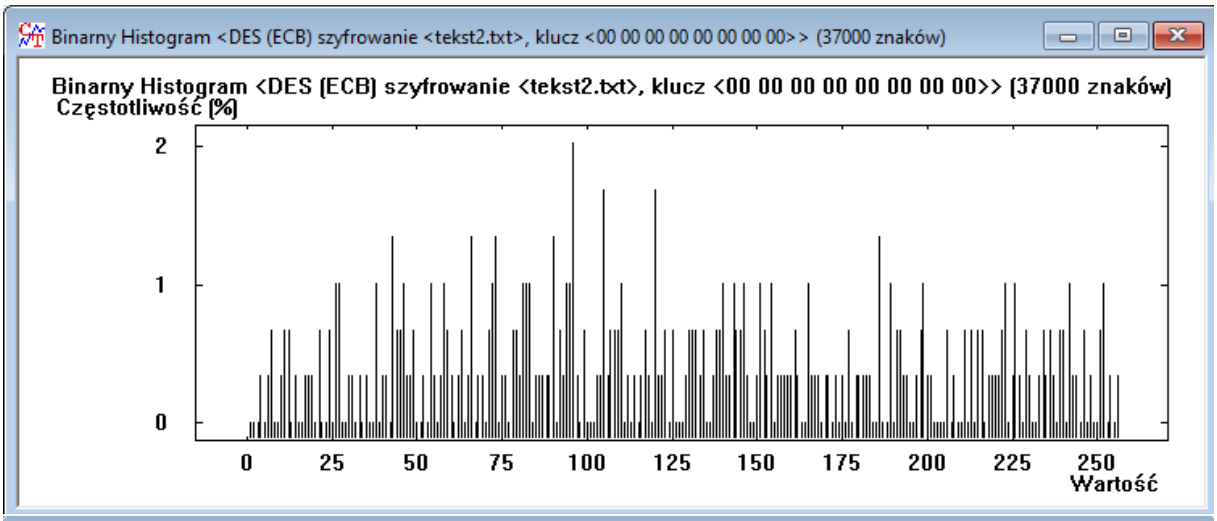
DES (ECB), tekst 1:

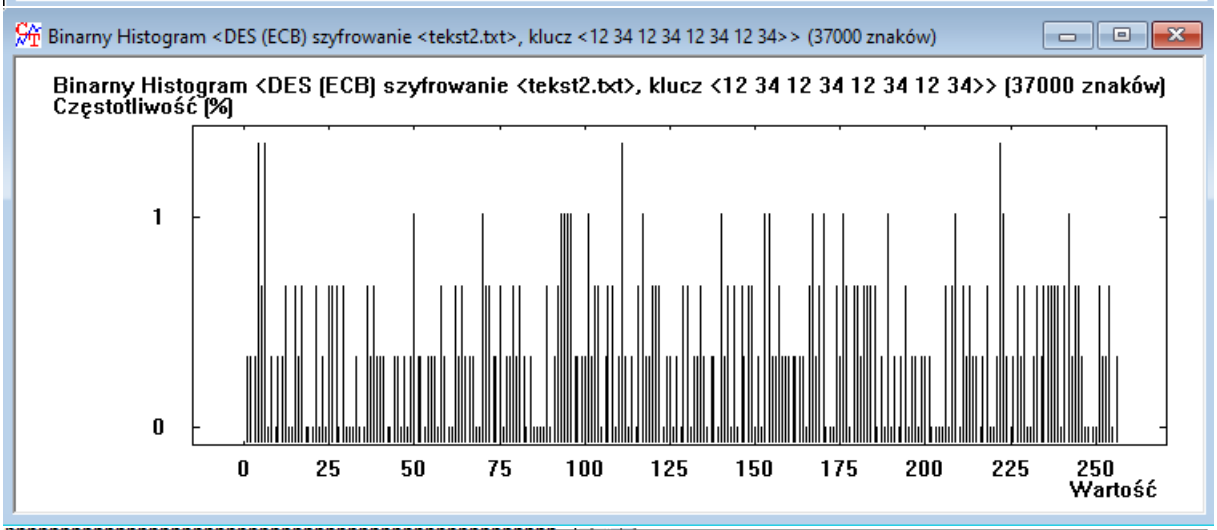


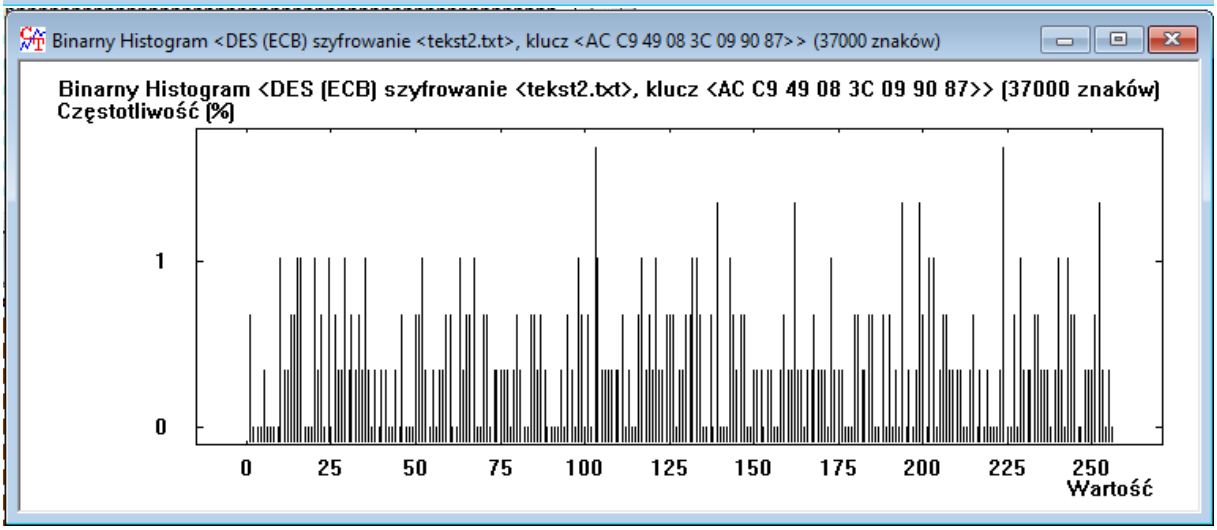




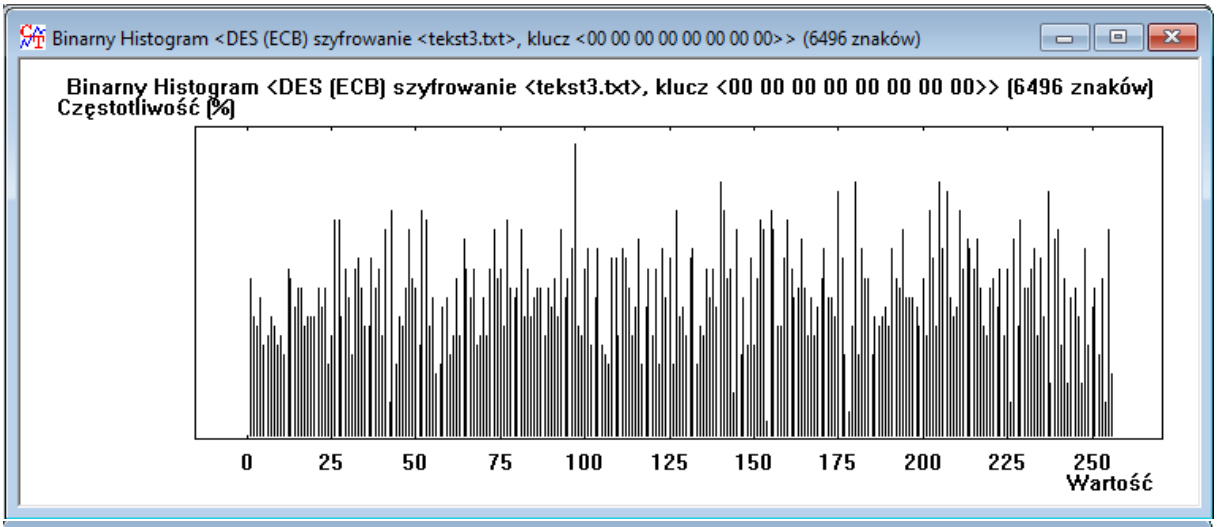
DES (ECB), tekst 2:

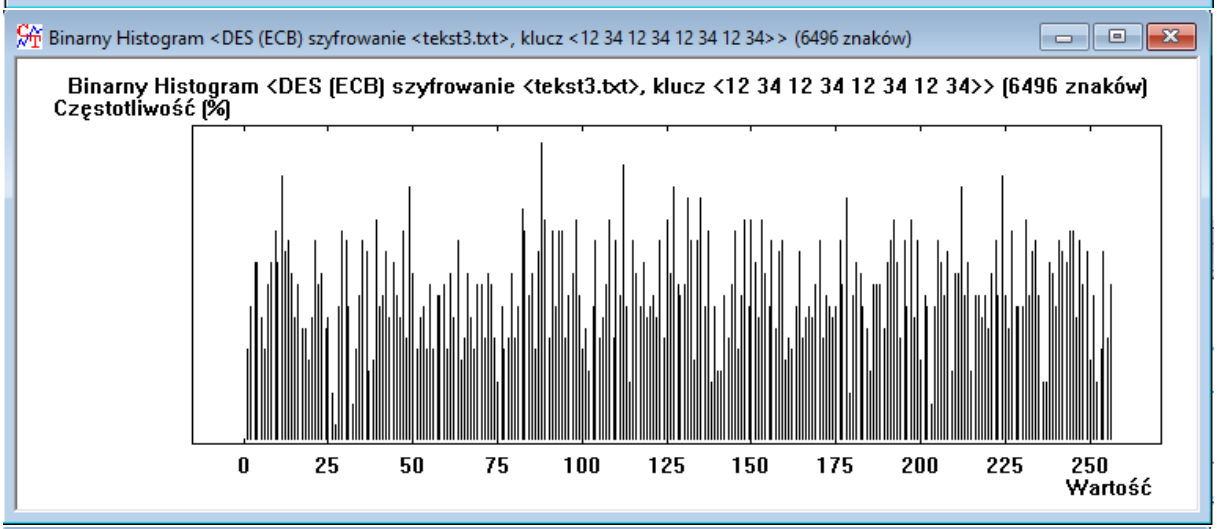


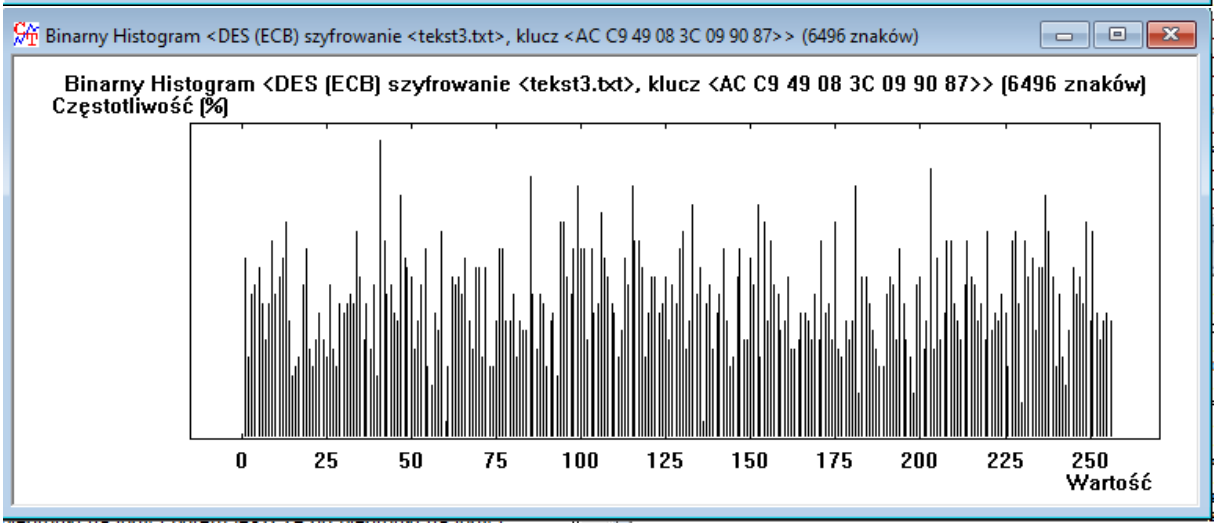




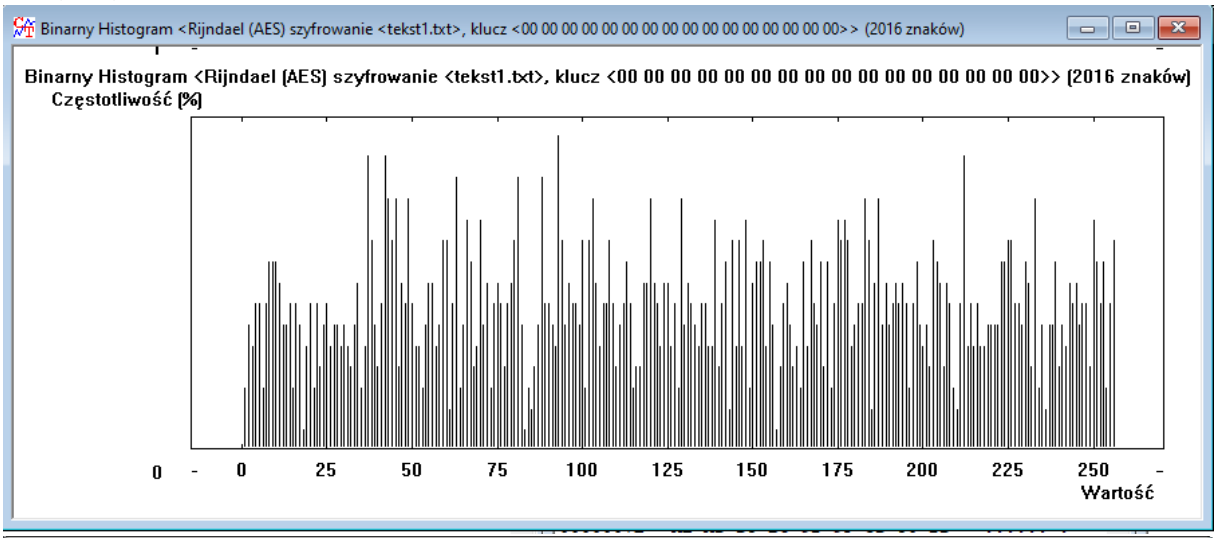
DES (ECB), tekst 3:

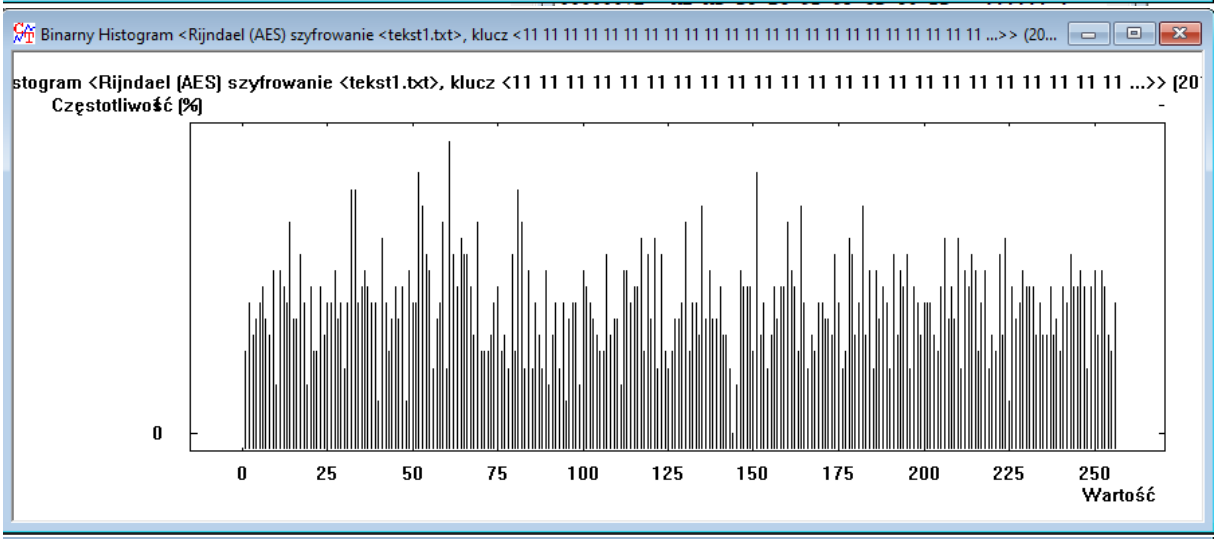


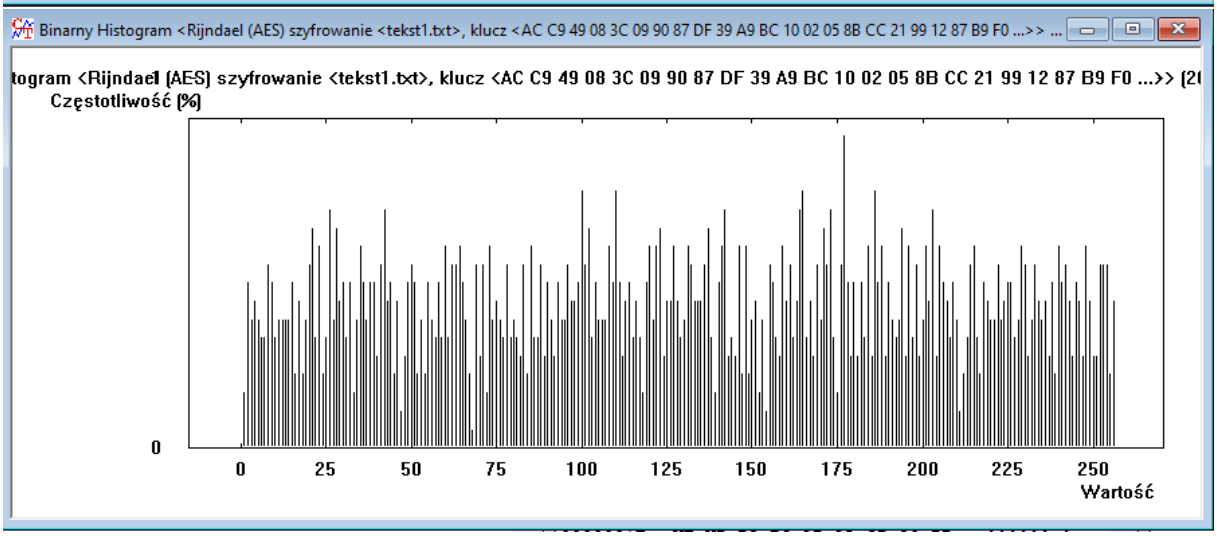




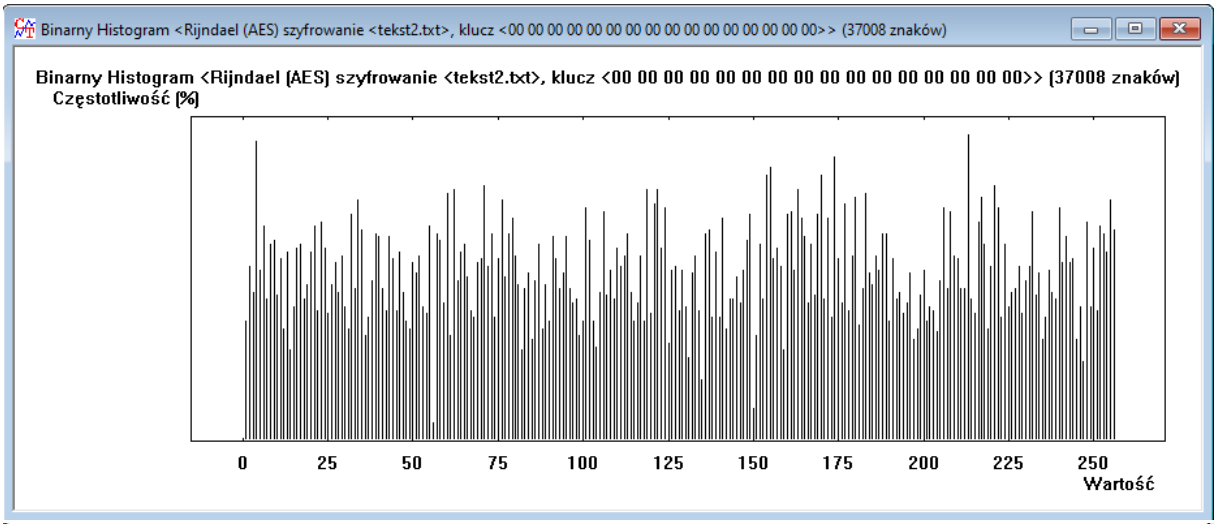
AES (CBC), tekst 1:

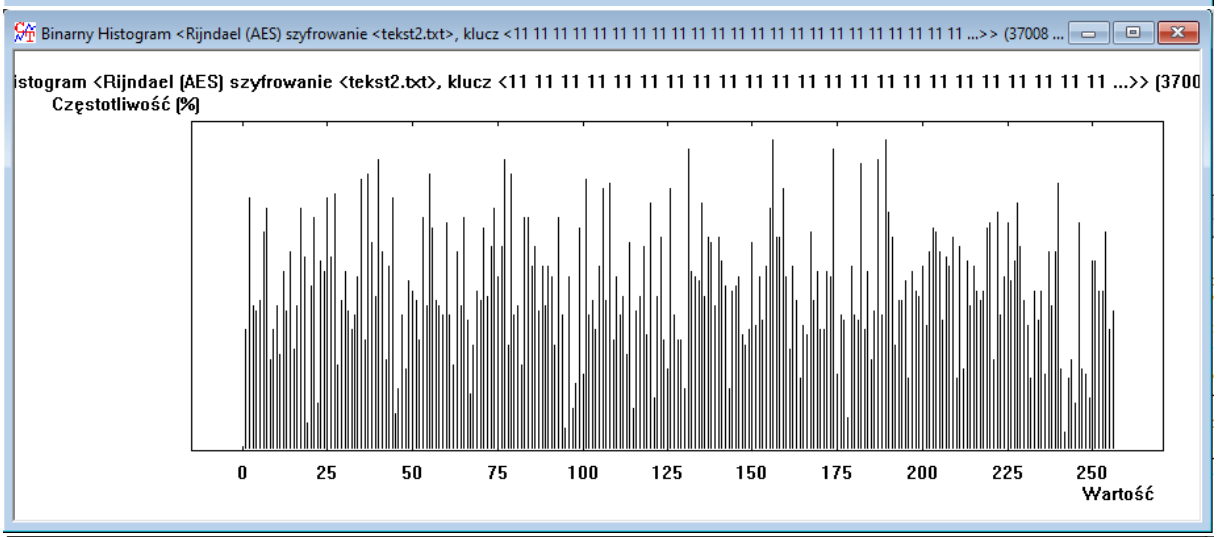


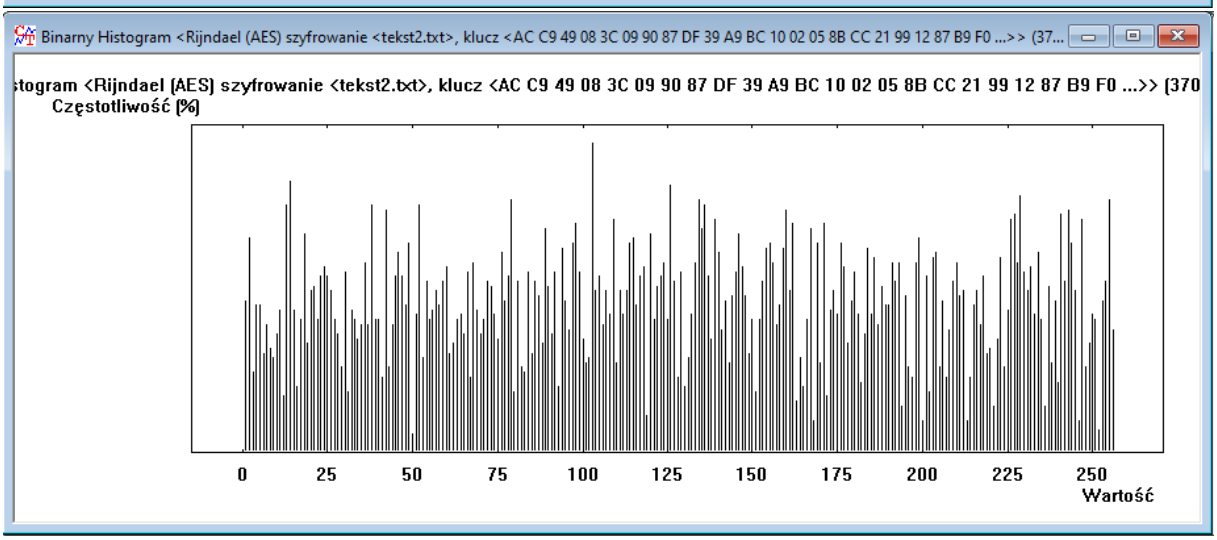




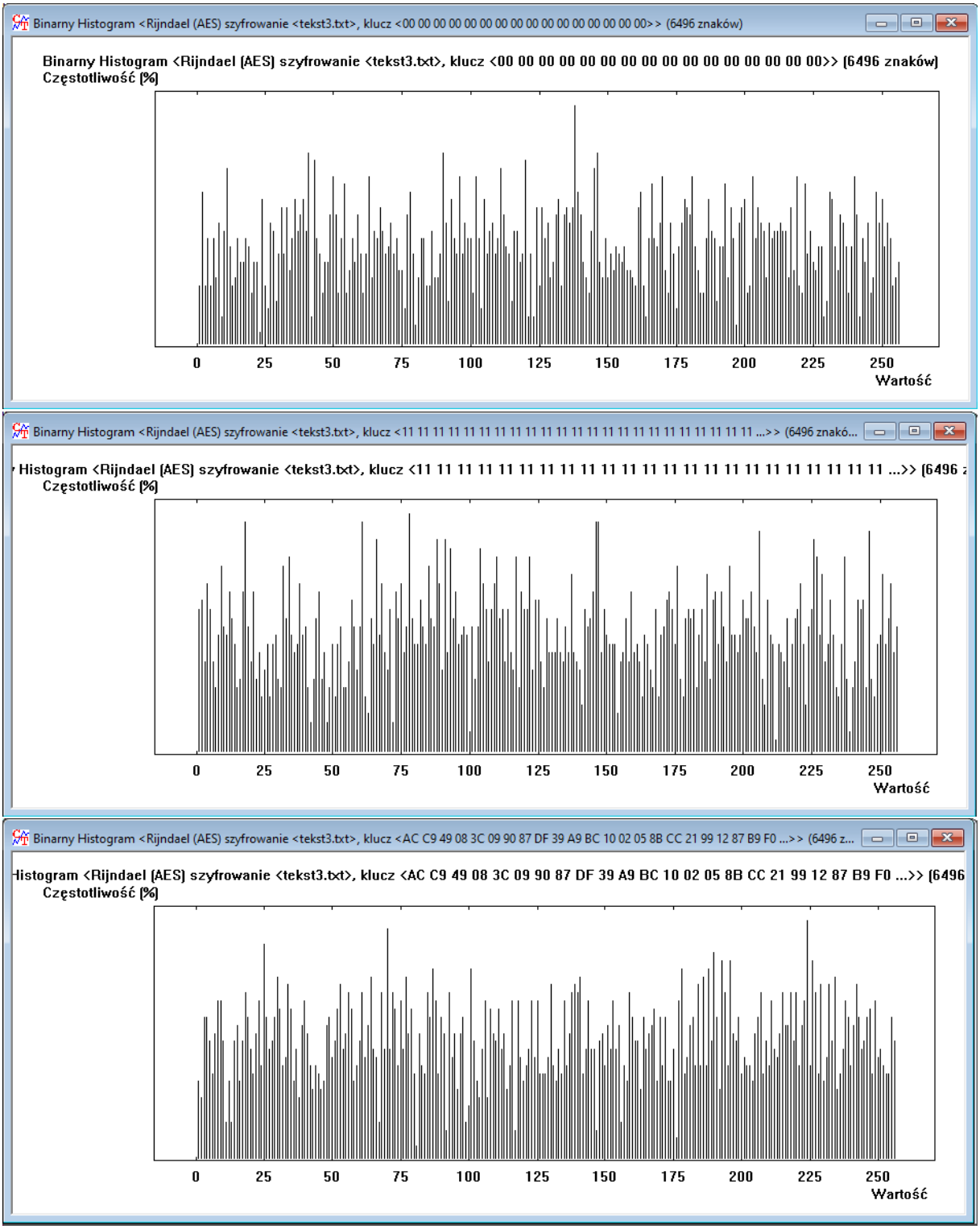
AES (CBC), tekst 2:







AES (CBC), tekst 3:



Pytanie 1.3;

Algorytmy blokowe są stosowane głównie w kryptografii do szyfrowania danych. Wykorzystuje się je w generatorach liczb pseudolosowych, adresach MAC oraz w mechanizmach do sprawdzania integralności danych. Podstawowe zastosowania algorytmów blokowych to:

- szyfrowanie danych (np. plików lub wiadomości)
- bezpieczeństwo transmisji w sieciach (np. VPN, HTTPS)
- szyfrowanie dysków (np. BitLocker)
- systemy bankowe i płatności online (szyfrowanie danych kart płatniczych)

Popularne algorytmy blokowe:

- AES (Advanced Encryption Standard) - najczęściej stosowany algorytm blokowy, uznawany za standard
- DES (Data Encryption Standard) - obecnie uznawany za przestarzały, zastąpiony przez AES
- 3DES (Triple DES) - wzmocniona wersja DES, ale powoli wychodzi z użycia
- Blowfish - stosowany w starszych systemach
- Twofish - następca Blowfish, również bezpieczny, ale mniej popularny niż AES

Standardowe i bezpieczne parametry:

- Długość bloku: 128 bitów (AES), choć niektóre algorytmy (np. Blowfish) używają 64-bitowych bloków, co jest uznawane za raczej mniej bezpieczne
- Długość klucza: AES obsługuje klucze o długości 128, 192 lub 256 bitów. Długość klucza 128 bitów jest uznawana za wystarczającą dla większości zastosowań, ale 256-bitowe klucze są stosowane w szczególnie wrażliwych aplikacjach.

Pytanie 1.4;

Dla wszystkich algorytmów blokowych entropia wzrosła po zaszyfrowaniu tekstów jawnych. W przypadku tekstu nr. 1 algorytmy DES i IDEA spowodowały najmniejszy wzrost entropii, a zaszyfrowany tekst wciąż posiadał znaki powtarzających się znaków, co widać dokładnie na histogramach dla tego tekstu. Z kolei na AES entropia wzrosła znacząco, zbliżając się za każdym razem do maksymalnej (4,64/4,7 – po przeskalowaniu), a słupki histogramów zdawały się być bardziej wyrównane w swoim rozkładzie, a znaki nie były tak powtarzalne jak dla dwóch poprzednich algorytmów.

Dla tekstu nr. 2 entropia także była wyższa w przypadku użycia AES w porównaniu do DES czy IDEA, ale różnica ta była o wiele mniejsza niż w przypadku tekstu nr. 1. Histogramy dla AES wydają się być bardziej wyrównane, gdzie w przypadku DES i IDEA niektóre znaki występowały znacznie częściej co było bardzo zauważalne na histogramach.

Dla tekstu nr. 3 entropia wzrosła w podobny sposób dla wszystkich algorytmów. Wszystkie histogramy wydają się być zbliżone, niezależnie od użytego algorytmu. IDEA najgorzej zmienia entropię dla klucza składającego się z samych zer (tworzyła widoczne wzniesienie na środku rozkładu). Dla AES zmiana entropii była znacząca, niezależnie od rodzaju klucza czy natury tekstu, a histogramy były wypłaszczone w każdym jego szyfrowaniu.

Pytanie 1.5;

Dla niektórych algorytmów klasycznych entropia albo nie ulegała zmianie wcale, albo dla takiego adfgvx nawet malała. Wzrost entropii zależał od długości i poziomu skomplikowania klucza. Histogramy często zachowywały swój kształt, ale były przesunięte, co mogło ujawnić użyty algorytm (np. Cezar).

Dla algorytmów blokowych z kolei entropie rośnie znacząco, co utrudnia łamanie szyfrów prostymi metodami, jak analiza histogramów, które są w tym przypadku dużo bardziej wyrównane w poziomie.

Pytanie 1.6;

Dla algorytmu z możliwością zmiany długości klucza AES wydłużenie klucza nie dało żadnej zauważalnej zmiany w entropi. Różnica ta jest prawdopodobnie minimalna i niezauważalna zarówno przy badaniu samego tekstu jak i jego histogramu.

Pytanie 1.7;

Dla zaszyfrowanego tekstu entropia w dużej mierze zależy od entropii tekstu jawnego. Jednak dla lepszych algorytmów takich jak AES nie ma to większego znaczenia – entropia pozostaje wysoka, niezależnie od tego czy tekst jawny jest jednorodny czy bardziej złożony.

Pytanie 1.8;

Dla szyfrowania IDEA wartość klucza zdecydowanie wpływa na entropię tekstu zaszyfrowanego. Szczególnie jest to widoczne dla klucza K1, gdzie entropia jest nieco niższa od pozostałych kluczy dla tego algorytmu. Natomiast w pozostałych kluczach zmiana klucza nie miała znacznego wpływu na entropię tekstu.

Pytanie 1.9;

Dla wszystkich algorytmów szyfrujących entropia tekstu zaszyfrowanego była zależna od użytego algorytmu. AES powodował zwiększenie entropii do wartości zbliżonej do tej maksymalnej (4,70 po przeskalowaniu) niezależnie długości klucza czy rodzaju szyfrowanego tekstu. Dla DES oraz IDEA widać było bardzo silną zależność między entropią tekstu jawnego a tajnego. Algorytmy te radziły sobie o wiele gorzej z szyfrowaniem tekstów nr. 1 i nr. 2 niż tekstu nr. 3 ze względu na ich większą jednorodność.

Zad 2. Tryby pracy algorytmów blokowych.

Zadanie 2.1-2.4;

Użyłem poniższych tekstów do kolejnych zadań:

Tekst 1:

Litera „n” powtórzona 2000 razy.

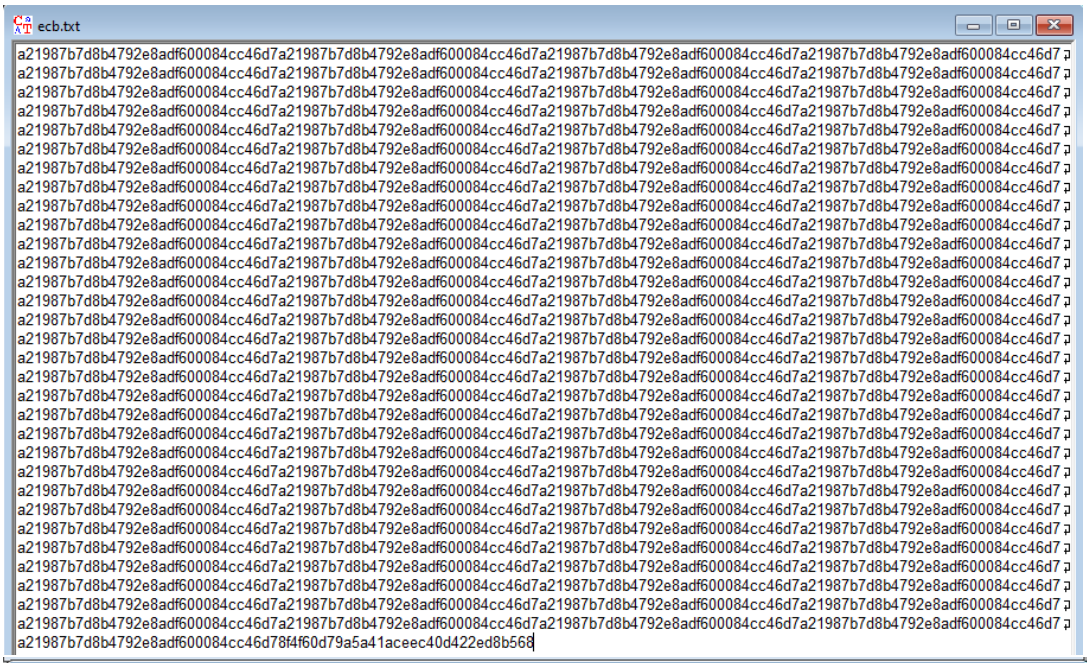
Do analizy wybrałem algorytm AES, w wariancie 128-bitowym.

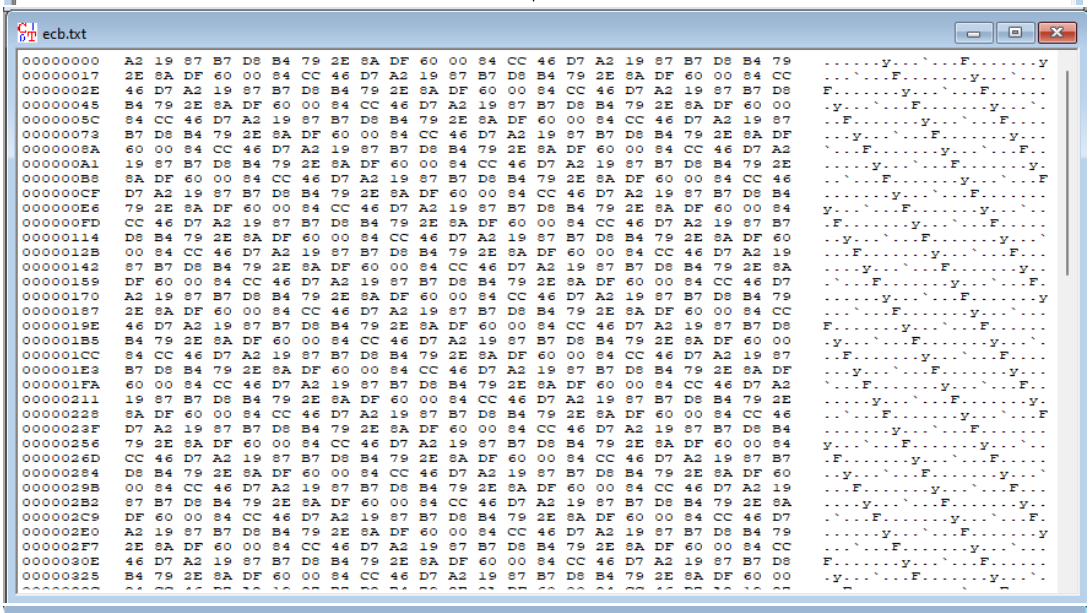
Klucz: B3 A4 09 34 3B C2 72 7B 34 C9 FF 87 96 05 3A F2

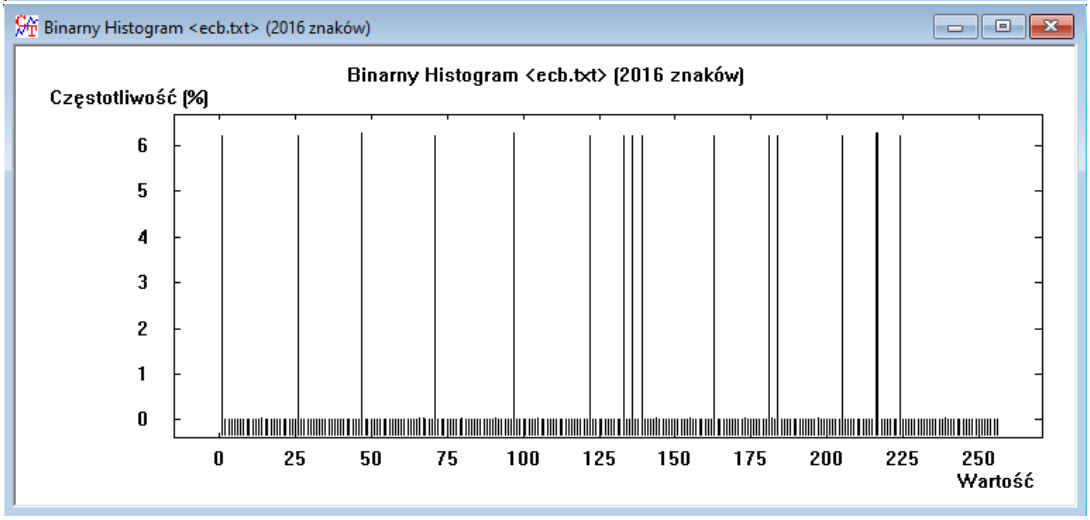
Wektor: 2B 17 A8 3B 91 C9 4F 37 E1 29 4D 79 12 A4 79 A1

ECB:

Entropia: 4,05/8 (niska), widoczne powtórzenia tekstu, histogram zdecydowanie nierównomierny.

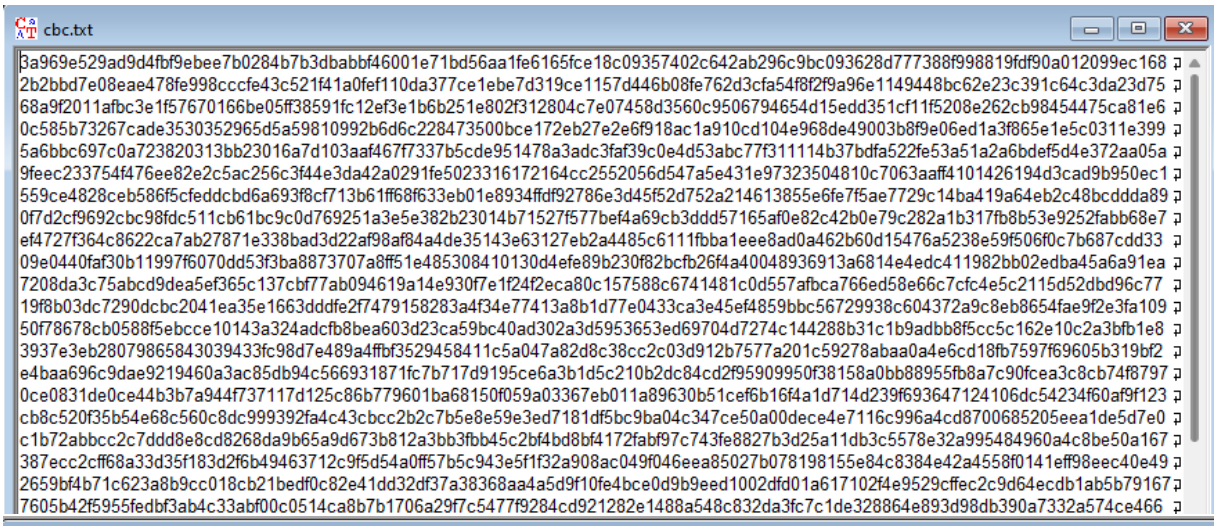


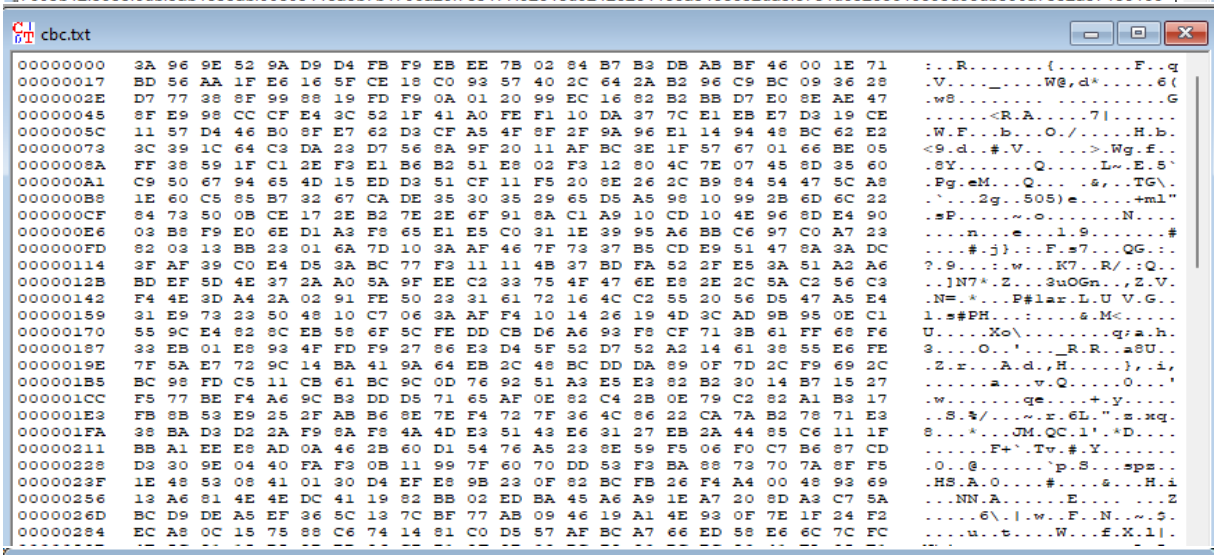


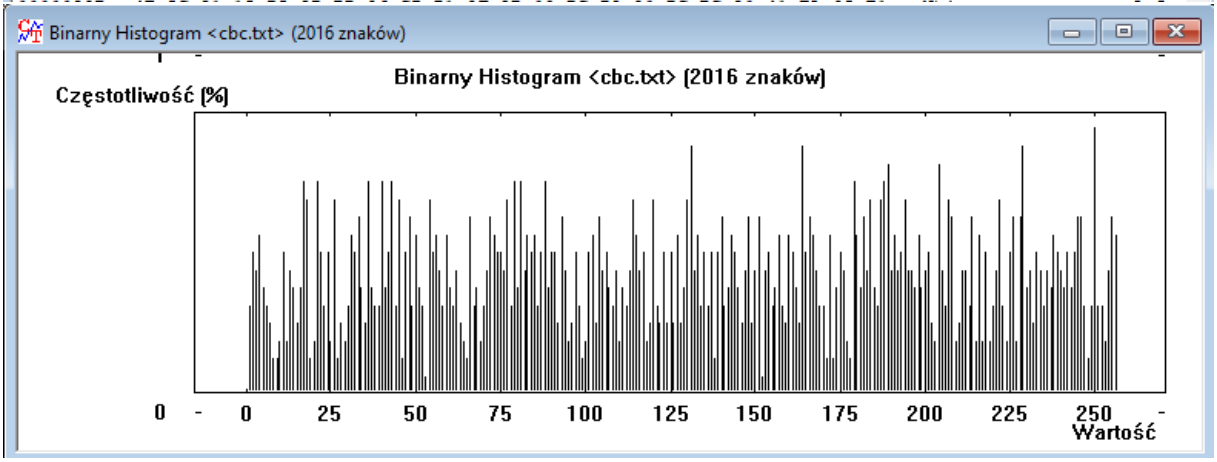


CBC:

Entropia: 7,9/8 (wysoka), brak widocznych powtórzeń tekstu, histogram wyrównany

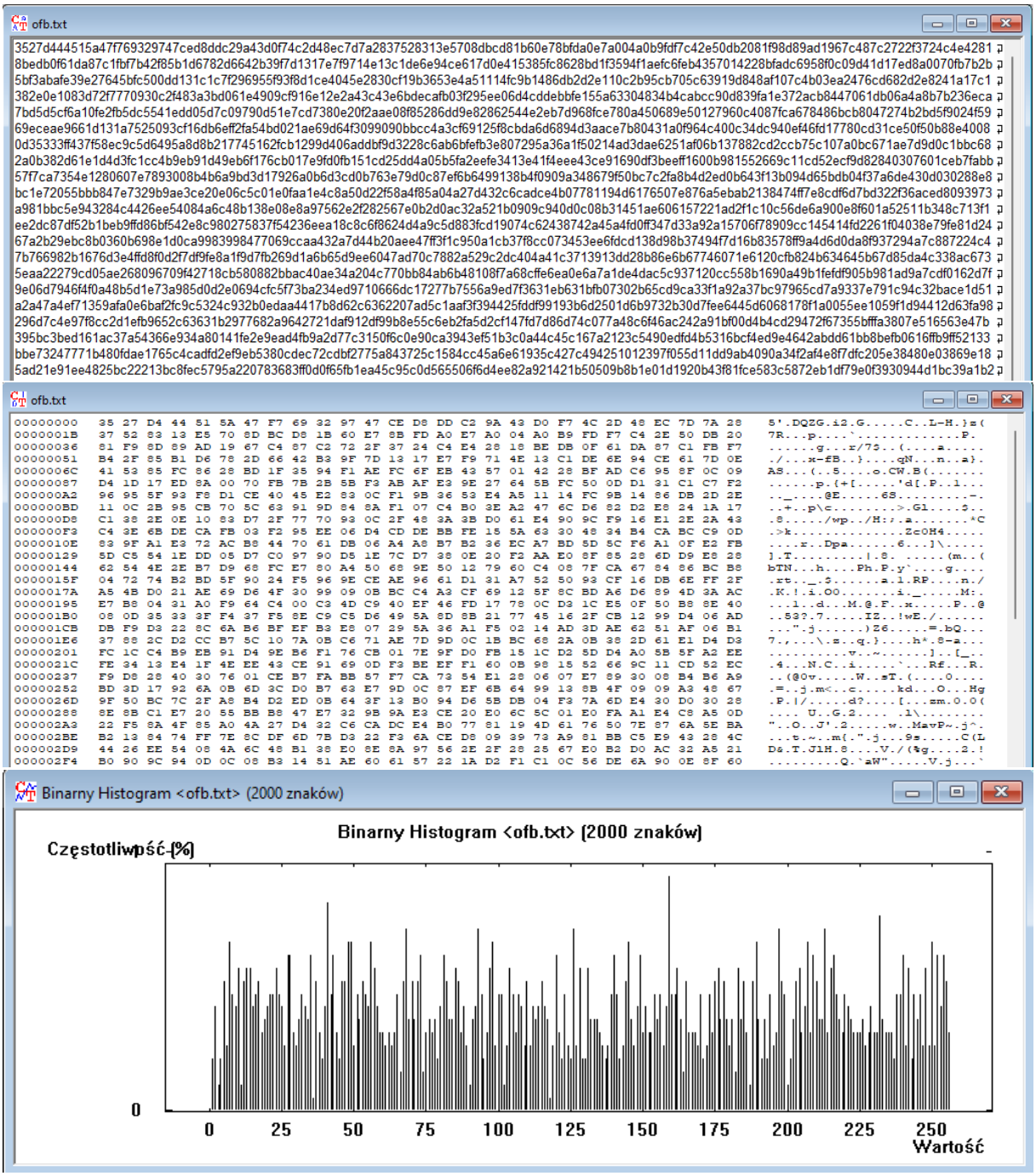




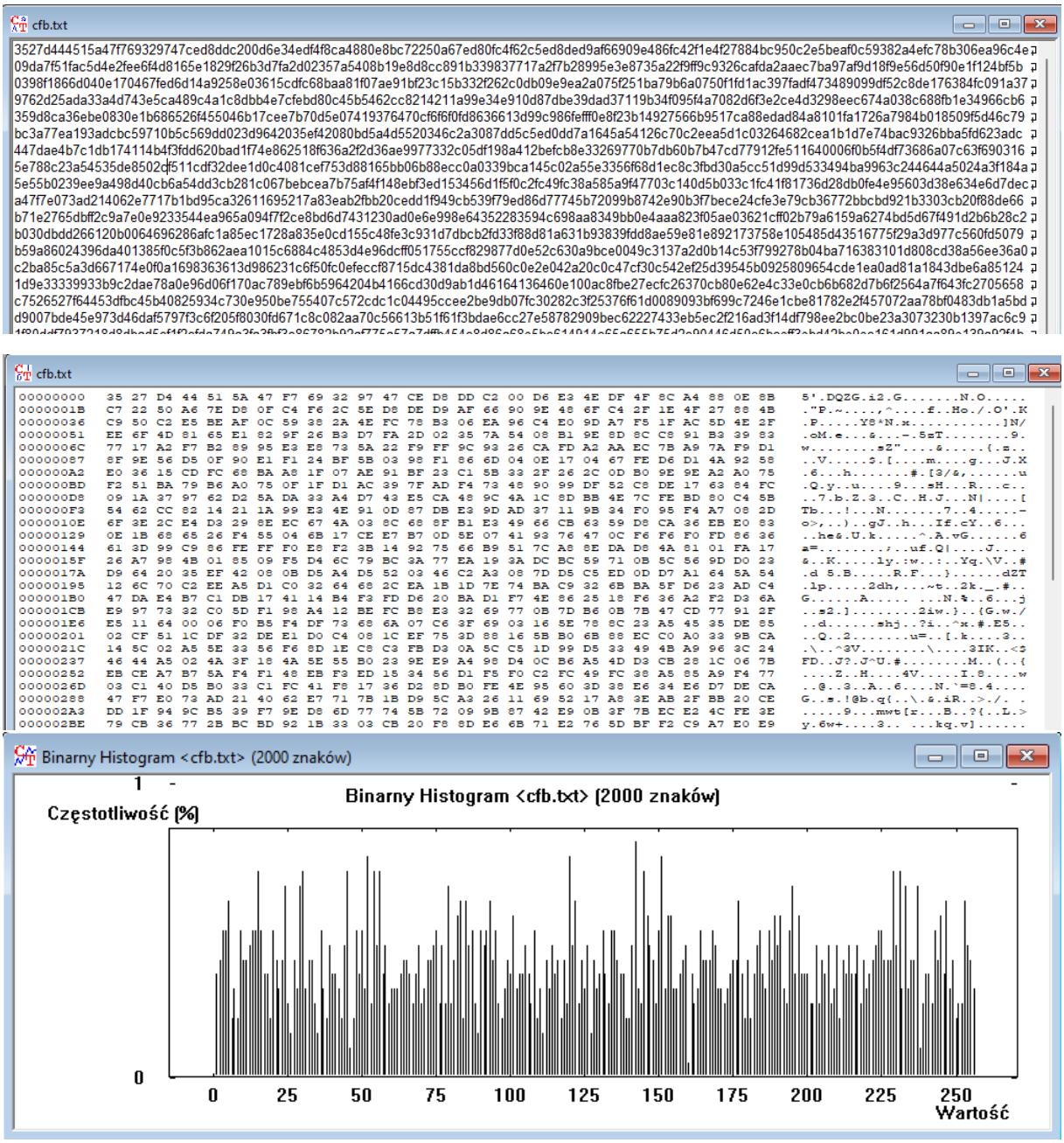


OFB:

Entropia: 7,89/8 (wysoka), brak widocznych powtórzeń tekstu, histogram wyrównany



CFB:
Entropia: 7,91/8 (wysoka), brak widocznych powtórzeń tekstu, histogram wyrównany (początek tekstu taki sam jak OFB)



Wnioski:

Najgorzej z zaszyfrowaniem tekstu nr. 1 poradził sobie ECB, wyraźnie widać powtarzające się 128bitowe bloki tekstu. CBC, OFB i CFB są jednak dużo lepsze i nie da się zauważyć natury tekstu jednorodnego (wysoka entropia, wyrównane histogramy)

Zadanie 2.5;

Użyłem poniższego tekstu:

Tekst 4:

„litwo ojczyzno moja ty jestes jak zdrowie ile cie trzeba cenic ten tylko sie dowie kto cie stracil dzis pieknosc twa w calej ozdobie widze i opisuje bo tesknie po tobie panno swieta co jasnej bronisz czestochowy i w ostrej swiecisz bramie ty co grod zamkowy nowogrodzki ochraniasz z jego wiernym ludem jak mnie dziecko do zdrowia powrocilas cudem gdy od placzacej matki pod twoje opieke ofiarowany martwa podnioslem powieke i zaraz moglem pieszo do twych swiatyn progu isc za wrocone zycie podziekowac bogu”

Do analizy wybrałem algorytm AES (CBC), w wariancie 128-bitowym.

Klucz: B3 A4 09 34 3B C2 72 7B 34 C9 FF 87 96 05 3A F2

a) zmień kilku bitów w różnych bajtach

zmiana jednego bitu w różnych bajtach blisko siebie:

```
!«•¨•z¼GuÏSTX•ñESO•ïja ty jestes jak zdrowie ile cie trzeba cenic ten tylko sie
dowie kto cie stracil dzis pieknosc twa w calej ozdobie widze i opisuje bo
tesknie po tobie panno swieta co jasnej bronisz czestochowy i w ostrej swiecisz
bramie ty co grod zamkowy nowogrodzki ochraniasz z jego wiernym ludem jak mnie
dziecko do zdrowia powrocilas cudem gdy od placzacej matki pod twoje opieke
ofiarowany martwa podnioslem powieke i zaraz moglem pieszo do twych swiatyn
progu isc za wrocone zycie podziekowac bogu
```

Zmiana jednego bitu w różnych bajtach daleko siebie:

```
jF••];NULËÇ2¼µlk &USja ty jkstes jak zdrowie ile cie trzeba cenic ten tylko sie dowie kto cie stracil
dzis pieknosc twa w calej ozdobie widze i opisuje bo tesknie po tobie panno swieta co jasnej bronisz
czestochowy i w ostrej swiecisz bramie tîSYN••²pw$µ+*(ETBSSCRTXÆy nowogrodlkj achraniasz z jego wiernym ludem
jak mnie dziecko do zdrowia powrocý{Ô#ESK••9&J¼¹<nZóÔÌo®!U±•Ñã•2•fÒi p•d twoje opieZ°CAN}ÔjtóOã•STX
ÄEñxrtwa podnio~lem powieke i zaraz moglem pieszo do twych swiatyn progu isc za wrocone zycie
podziekowac bogu
```

Dla zmienionego jednego bitu, po odszyfrowaniu zmieniony jest 16-znakowy fragment tekstu, pozostała część jest nienaruszona. Dla zmienionych kilku bitów w różnych bajtach, to gdy są one w jednej 128-bitowej grupie (blisko), to zmienia się tylko jeden 16-znakowy fragment, zaś gdy zmieniane bity są w odległości większej niż 128 bitów, to zmienianych jest klika grup 16-znakowych (do których należą zmienione bity), pozostała część tekstu nie ulega zmianie.

b) dodanie jednego bajtu

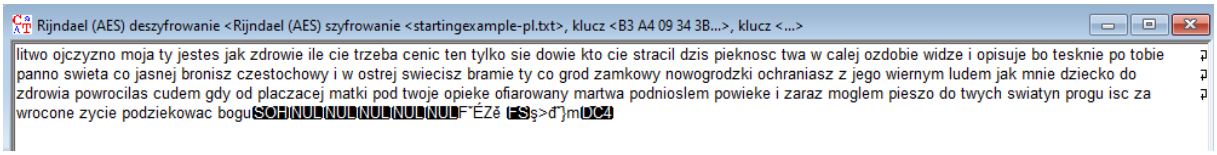
Dodanie bajtu zaburzy strukturę bloków szyfru, powodując przesunięcie reszty danych i błędne odszyfrowanie całości. Ponieważ AES wymaga, aby długość danych była wielokrotnością 16 bajtów, algorytm może zgłosić błąd związany z rozmiarem bloku (dla internetowych programów szyfrujących)

Output

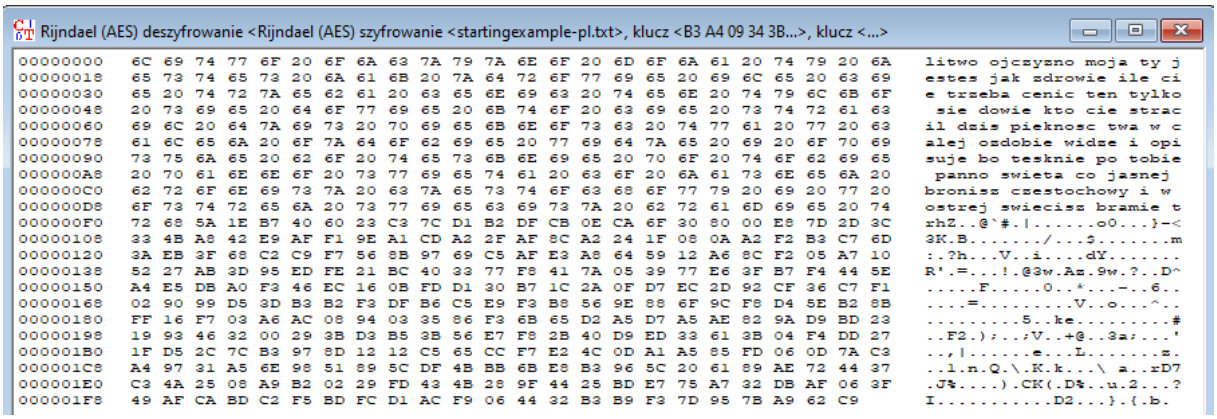
```
Unable to decrypt input with these parameters.
```

Jednak po wrzuceniu tego w Cryptoola mamy kolejno:

-dla bajtu dodanego na końcu wiadomości, całość tekstu jest roszyfrowana poprawnie, z dodanym szumem na końcu



-dla bajtu dodanego w środku wiadomości, fragment tekstu aż do zmiany jest rozszyfrowany poprawnie, a dalej jest szum ponieważ bajty w blokach są przesunięte i nie tworzą już wspólnej całości



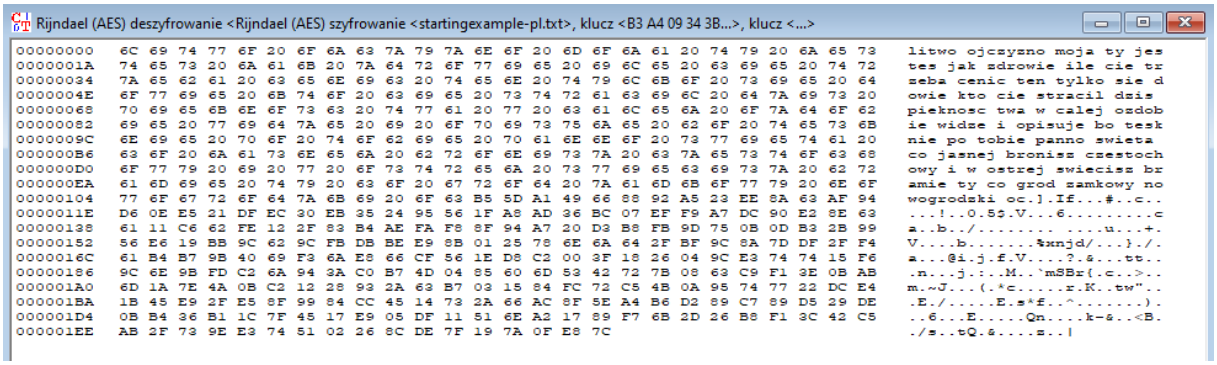
c) usunięcie jednego bajtu

Analogicznie do dodania bajtu z punktu b) nie działa to dla szyfrowania online.

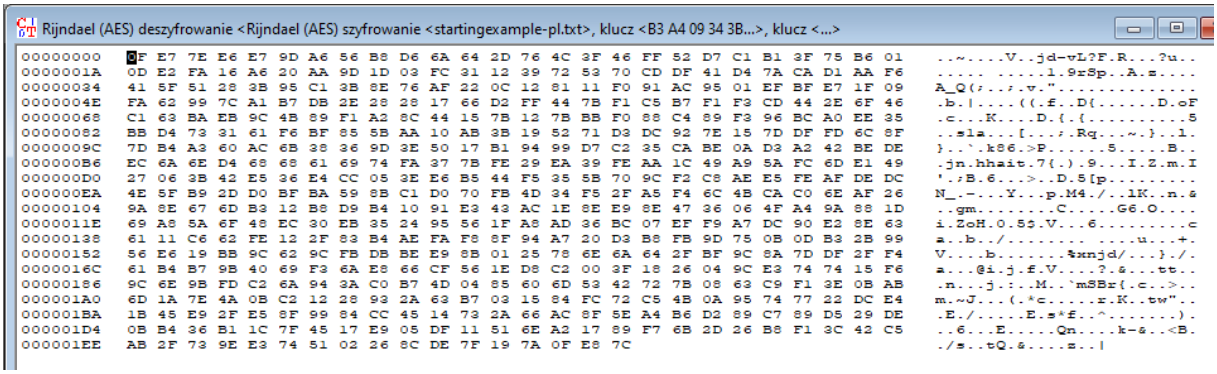
Output

```
Unable to decrypt input with these parameters.
```

W Cryptool również analogicznie widzimy, że tekst zachowuje się normalnie aż do momentu zmiany bajtu, po czym zamienia się w chaos z uwagi na przesunięte bajty w blokach.



Oczywiście jeśli usuniemy jakikolwiek bajt z pierwszych 16 bajtów, wtedy cała wiadomość zostaje przesunięta i nie będziemy w stanie nic rozszyfrować:



Wniosek: Usunięcie i dodanie bajtów ma podobny wpływ i charakteryzuje się głównie normalnym działaniem aż do miejsca zmiany, po którym to ze względu na przesunięcie w lewo lub prawo o jeden wszystkich pozostałych bajtów, dalsze rozszyfrowanie bloków bajtów staje się bezskuteczne.

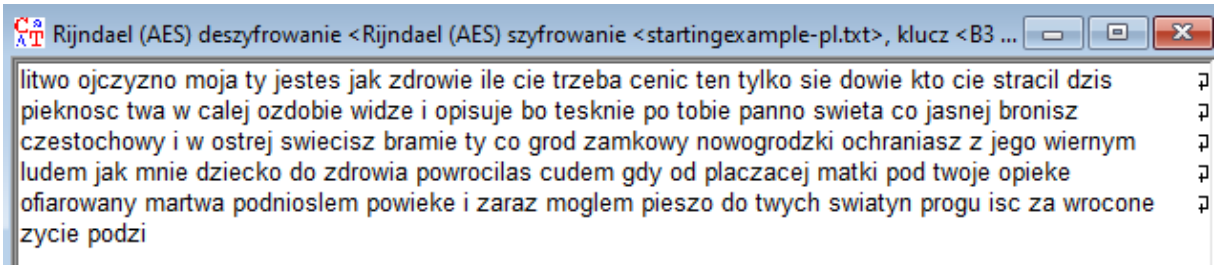
d) dodanie/usunięcie fragmentu tekstu równego długości bloku algorytmu

Dla usuniętego środkowego fragmentu o długości 16 bajtów (długość bloku tego algorytmu) możemy zauważyć, że zmianie uległy jedynie znaki, znajdujące się pomiędzy dwoma blokami algorytmu (oprócz zmienionych znaków sam tekst jest krótszy o 16 znaków). Reszta tekstu pozostała bez zmian.

```
litwo ojczyzno moja ty jestes jak zdrowie ile cie trzeba ce
nic ten tylko sie dowie kto cie stracil dzis pieknosc twa w
 calej ozdobie widze i opisuje bo tesknie po tobie panno sw
ieta co jasnej bronisz czestochowy i w ostrej swiecisz bram
ie ty co grod zamkowy nowogrodzki ochraniasz z jego wiernym
 ludem jaEFX.)m.s.u..9...{.$..nudem gdy od placzacej matki
pod twoje opieke ofiarowany martwa podniosłem powieke i zar
az moglem piesso do twych swiatyn progu isc za wrocone zyci
e podziekowac bogu
```

Dla usuniętego fragmentu o długości 16 bajtów od końca tekstu tajnego, możemy zauważyć, że tekst nie uległ zmianie, oprócz faktu że informacja o ostatnich 16 znakach została stracona. Analogicznie dla każdej innej konfiguracji, w której usuwamy przedział 16 bajtów, które definiują w całości pojedynczy

blok algorytmu.



Dla bajtów dodanych:

(na końcu tekstu tajnego)

```
litwo ojczyzno moja ty jes
tes jak zdrowie ile cie tr
zeba cenic ten tylko sie d
owie kto cie stracil dzis
pieknosc twa w calej ozdob
ie widze i opisuje bo tesk
nie po tobie panno swieta
co jasnej bronisz czestoch
owy i w ostrej swiecisz br
amie ty co grod zamkowy no
wogrodzki ochraniasz z jeg
o wiernym ludem jak mnie d
ziecko do zdrowia powrocil
as cudem gdy od placzacej
matki pod twoje opieke ofi
arowany martwa podnioslem
powieke i zaraz moglem pie
szo do twych swiatyn progu
 isc za wrocone zycie podz
iekowac bogu.........)....
..{e..0
```

(w środku tekstu tajnego)

```
litwo ojczyzno moja ty jestes jak
 zdrowie ile cie trzeba cenic ten
 tylko sie dowie kto cie stracil
dzis pieknosc twa w calej ozdobie
 widze i opisuje bo tesknie po to
bie panno swieta co jasnej bronis
z czestochowy i w ostrej s....7.0
....Q.J..P...T.}..9...M....co gro
d zamkowy nowogrodzki ochraniasz
z jego wiernym ludem jak mnie dzi
ecko do zdrowia powrocilas cudem
gdy od placzacej matki pod twoje
opieke ofiarowany martwa podniosl
em powieke i zaraz moglem pieszo
do twych swiatyn progu isc za wro
cone zycie podziekowac bogu
```

Dla tych fragmentów 16 bajtowych, które zostały dodane w odpowiednich blokach (modulo 16 bajtów, tak aby nie zaburzać rozszyfrowania innych bloków) - tekst pozostaje niezmienny z dodanym fragmentem „szumu” na jeden dodatkowy blok. Jeśli jednak dodamy nasz 16 bajtowy blok, pomiędzy dwa inne bloki, „zaszumione” zostaną oba te bloki, co będzie skutkować podwójną ilością znaków które się w nic nie układają.

Podsumowanie:

a) Dla trybu ECB zmiana bitu wpływa jedynie na dany blok, ponieważ każdy blok jest szyfrowany niezależnie. Dla trybu CBC zmiana bitu wpływa zarówno na dany blok w którym znajduje się bit jak blok kolejny, ponieważ CBC stosuje XORowanie kolejnych bloków ze sobą. Dla OFB i CFB zmiana bitu wpływa tylko na odpowiednia część odszyfrowanego strumienia, ale wpływa na kolejne bloki.

b) c) Dla trybów ECB i CBC dodanie/usunięcie bajtów powoduje przesunięcie bloków, co uniemożliwia prawidłowe odszyfrowanie od miejsca zmiany. Oba tryby są podobnie wrażliwe na takie zmiany. Dla trybów OFB i CFB dodanie/usunięcie bajtów również prowadzi do zniekształceń w reszcie tekstu, ale z powodu zaburzenia synchroniczności między szyfrowanym tekstem a generowanym strumieniem.

d) Dla trybu ECB dodanie/usunięcie bloku danych nie wpływa na resztę szyfrogramu, ponieważ bloki są niezależne. Dla trybu CBC dodanie/usunięcie całego bloku wpływa na resztę szyfru (ale tylko te po, a nie przed momentem zmiany), ponieważ dalsze bloki nie są niezależne od bloków poprzednich. Dla trybów OFB i CFB dodanie/usunięcie bloku danych zaburzają synchronizację i powodują błędy w dalszej części strumienia danych

Zadanie 2.6;

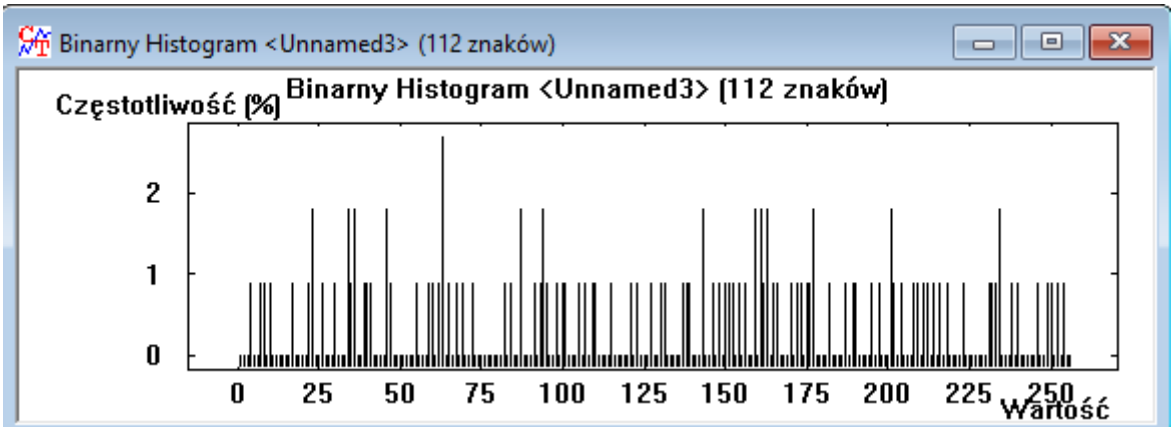
a) Gołym okiem widać, że długość bloku wynosi 16 bajtów, ponieważ ten sam fragment tekstu jest powtarzany co okres 16 bajtów. Wynika z tego, że bloki są o długości 8 znaków. Przeprowadzone wcześniej eksperymenty wskazują na to, że mamy tu do czynienia z przypadkiem szyfrowania tekstu jednorodnego trybem ECB.

b) W tym tekście także widać cykliczne powtórzenia ciągów bajtów, jednak tym razem są one co 32 bajty. Wynika z tego, ze bloki są o długości 16 znaków. Przeprowadzone wcześniej eksperymenty wskazują na to, że ponownie mamy tu do czynienia z przypadkiem szyfrowania tekstu jednorodnego trybem ECB.

c) W tekście brakuje jednego bitu w ostatnim jego bloku (ostatnie kilka znaków jest napisane inną czcionką co sugeruje jakąś zmianę na ostatni moment). Z tego powodu do poniższej analizy została wstawiona na sam koniec wartość ‘0’, tak aby ostatni bajt miał postać „50” w HEX.

```
6a239123a19647032e3e637ab0b02d56
26b593edf8445c07e9e79e408ec89e56
16d9e89bef28c92d61ab6d82d39921ac
8109e9d03dafd2fd6416a0ba97513ede
0615a2688a785319d75d95a4a9aed55e
e6c221cfa51d7e89f93a5dfbc4a2bc22
368ecba0bd42c8f56c2710723b883e5
```

Entropia niska (6,53), histogram dosyć niewyrównany, ale ciężko stwierdzić przez krótki fragment.



Może to sugerować użycie jakiegoś „słabszego” algorytmu np. IDEA, oraz użycie trybu ECB lub CBC. Gołym okiem nie da się zauważyć żadnych cykliczności, więc trudno wskazać długość bloku algorytmu. Na końcowy kształt tekstu tajnego mogło również wpłynąć zróżnicowanie tekstu jawnego.

Pytanie 2.7;

Dla trybów CBC, OFB oraz CFB entropia jest na bardzo wysokim poziomie (7,89-7,91/8), natomiast tryb ECB charakteryzuje się bardzo niską entropią (4,05/8). Kryptogram trybu ECB posiada widoczne, cykliczne powtórzenia ciągów liter, natomiast kryptogramy CBC, OFB i CFB nie posiadają już takiej cechy.

Pytanie 2.8;

Dla trybu ECB oraz CBC niepoprawnie rozszyfrowane są jedynie specyficzne bloki 16 lub 32 znaków, w których zostały przekłamane bity. Pozostała część tekstu nie uległa zmianie.

Dla trybu OFB niepoprawnie rozszyfrowywane są jedynie znaki, w których znajdują się przekłamane bity.

Dla trybu CFB niepoprawnie rozszyfrowane są znaki z przekłamanym bitem oraz blok tekstu w odległości 16 bajtów.

Pytanie 2.9;

Dla trybu CBC wiadomość zostaje „zaszumiona” od momentu w którym nastąpiła strata (a konkretniej w bloku, w którym nastąpiło pierwsze przesunięcie pozostałych bajtów).

Dla trybu ECB utrata wiadomości zaczyna się od bloku, w którym nastąpiło usunięcie pierwszego bajta, do wystąpienia bloku gdzie został usunięty ostatni bajt.

Dla trybów OFB i CFB utrata wiadomości zaczyna się od pierwszego usuniętego bajta i trwa do końca wiadomości, ponieważ reszta bajtów zostaje przesunięta.

Pytanie 2.10;

ECB jest prostszy i szybszy w deszyfrowaniu, ale mniej bezpieczny, ponieważ ujawnia wzorce w jednorodnych i długich tekstach. Jest odporny na zakłócenia, więc dobrze nadaje się do szybkich transmisji danych, choć lepiej sprawdza się z tekstami zróżnicowanymi.

CBC zapewnia większe bezpieczeństwo i ukrywa naturę tekstu, ale jest bardziej podatny na błędy, ponieważ zakłócenie jednego bitu może zepsuć cały szyfrogram. Nadaje się do transmisji, które wymagają bezpieczeństwa (np. VPN) albo szyfrowania poufnych danych (np. wiadomości email) oraz do szyfrowania plików, gdzie bezpieczeństwo jest kluczowe, a zakłócenia transmisji są mało prawdopodobne.

Pytanie 2.11;

Dla ECB jest możliwe, ponieważ każdy blok danych jest szyfrowany oraz deszyfrowany niezależnie od innych. Bloki więc, można podzielić na kilka części, które będą przetwarzane równolegle na różnych komputerach, a wynik końcowy będzie taki sam, jak w przypadku przetwarzania na jednym komputerze.

Dla CBC równoległe szyfrowanie i deszyfrowanie nie jest możliwe. Szyfrowanie nie jest możliwe, ponieważ każdy blok jest ściśle zależny od poprzedniego (z powodu XORa z poprzednim szyfrogramem). Deszyfrowanie również nie jest możliwe, ponieważ konieczne jest uprzednie odszyfrowanie poprzedniego bloku, więc również wymaga sekwencyjności.

Dla OFB i CFB nie jest możliwe, ponieważ każdy blok zależy od poprzedniego (przy OFB generowany ciąg musi być sekwencyjny, a w CFB sprzężenie zwrotne wymaga przetwarzania poprzednich bloków).

Wniosek:

ECB to jedyny tryb, w którym szyfrowanie i deszyfrowanie można prowadzić równolegle.