Badanie jakości szyfrów blokowych

Gabriel Wachowski, 145275

1. Analiza czasów szyfrowania i deszyfrowania wybranych trybów pracy szyfru AES

Zbadane zostaną czasy szyfrowania trzech tekstów o różnej długości w zależności od wybranego trybu pracy szyfru. Badane tryby pracy to:

- Electronic Code Book (ECB)
- Cipher-Block Chaining (CBC)
- Cipher FeedBack (CFB)
- Output FeedBack (OFB)
- CounTer Mode (CTR)
- Counter with CBC-MAC (CCM) Mode
- EAX Mode
- Galois Counter Mode (GCM)
- Syntethic Initialization Vector (SIV)
- Offset Code Book (OCB)

Wyniki

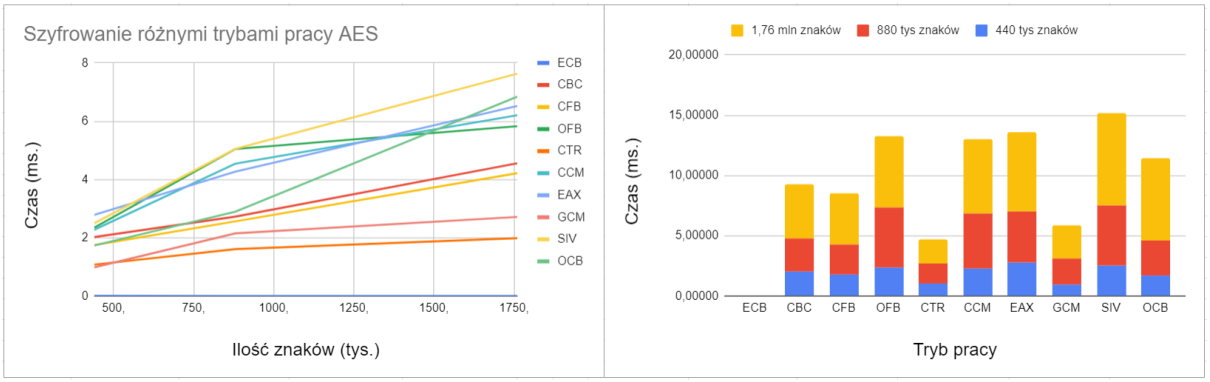
Wyniki dla trzech plików o różnej wielkości prezentują się następująco:

szyfrowanie			
tryb	440 tys znaków	880 tys znaków	1,76 mln znaków
ECB	0,00200	0,00080	0,00030
CBC	2,02410	2,72430	4,55510
CFB	1,74800	2,56150	4,21300
OFB	2,34680	5,04570	5,82960
CTR	1,07630	1,60850	1,98620
CCM	2,27510	4,54440	6,20560
EAX	2,78320	4,26820	6,52530
GCM	0,98550	2,15160	2,71360
SIV	2,49870	5,05290	7,63360
OCB	1,73230	2,89070	6,83980

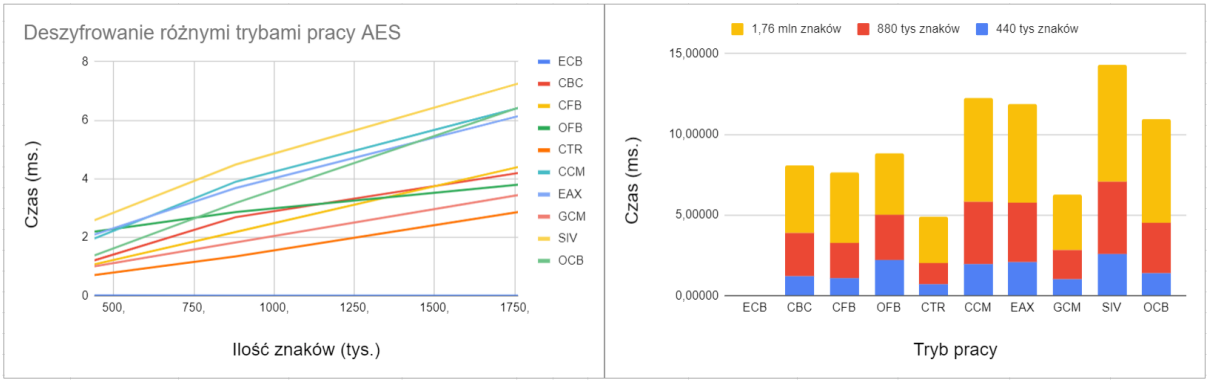
deszyfrowanie			
tryb	440 tys znaków	880 tys znaków	1,76 mln znaków
ECB	0,00100	0,00030	0,00040
CBC	1,20810	2,68280	4,19330
CFB	1,07150	2,17650	4,39640
OFB	2,18740	2,85590	3,79500
CTR	0,70240	1,33990	2,85660
CCM	1,95850	3,89180	6,41310
EAX	2,09110	3,67970	6,13710
GCM	1,00370	1,82080	3,43560
SIV	2,58130	4,48230	7,25090
OCB	1,37790	3,16530	6,42440

zdj. tabela wyników dla trybów pracy AES

Dane te zostały przedstawione na wykresach liniowych pokazujących przyrost czasu pracy w zależności od długości szyfrowanego/deszyfrowanego tekstu. Dodatkowo obok został umieszczony wykres słupkowy prezentujący te same dane, ale w formie pozwalającej łatwiej dostrzec sumaryczny czas pracy i różnice osiągów pomiędzy poszczególnymi trybami.

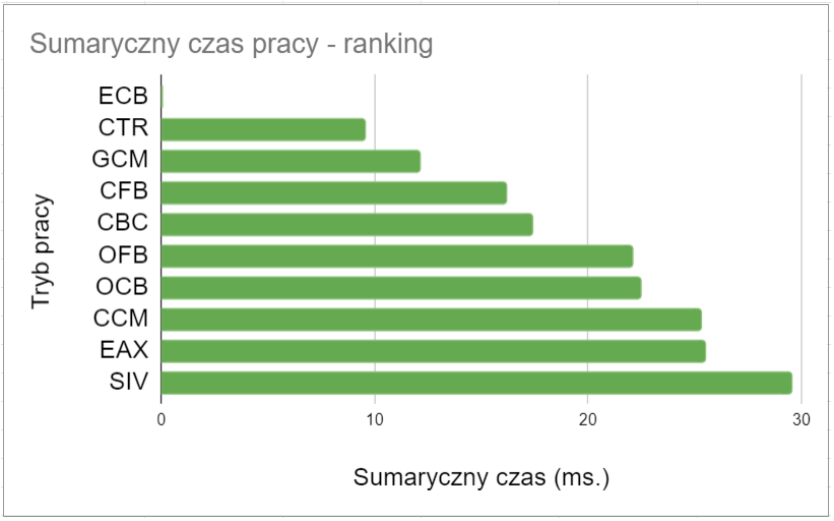


zdj. Wykres czasu od długości szyfrowanego tekstu dla trybów pracy szyfru AES.



zdj. Wykres czasu od długości deszyfrowanego tekstu dla trybów pracy szyfru AES.

Przebiegi czasowe trybów pracy dla każdego pliku zostały zsumowane. Dzięki temu otrzymaliśmy sumaryczny czas szyfrowania i deszyfrowania wszystkich plików. Pozwoliło nam to posortować tryby i przedstawić ranking badanych trybów. Ranking w formie graficznej prezentuje się następująco:



zdj. Ranking trybów pracy w zależności od czasu szyfrowania i deszyfrowania plików wejściowych.

Wnioski

Zdecydowanie najbardziej rzucającym się w oczy elementem rankingu jest wynik trybu ECB. Czas zaszyfrowania i odszyfrowania tekstów zawierających łącznie ok. 3 miliony znaków nie przekracza jednej milisekundy. Tryb ten jest niesamowicie szybki. Niestety, jest on również mało bezpieczny i nie powinien być wykorzystywany w aplikacjach. Tryby CTR oraz GCM to tryby licznikowe, opierają swoje działanie na liczniku inkrementowanym co jedną rundę szyfrowania. Oba tryby cechują się wysoką wydajnością. Tryb GCM idealnie nadaje się do przetwarzania w sposób równoległy i jest szeroko stosowany z powodu swoich osiągów[1]. Warto jeszcze zauważyć, że bardzo podobne do siebie tryby, czyli CBC oraz CFB uzyskały prawie identyczne czasy. Podobnie w przypadku par OFB i OCB oraz CCM i EAX. Te tryby są zdecydowanie wolniejsze od czołówki rankingu. Tryby licznikowe (CTR/GCM) poradziły sobie z tym samym zadaniem w prawie dwukrotnie krótszym czasie. Nie trudno się domyślić, że tryby z końca rankingu na ogół zapewniają większą ochronę i są trudniejsze do złamania niż te z krótszymi czasami. Warto jeszcze wspomnieć o ostatnim zawodniku w rankingu czyli trybie SIV (Synthetic Initialization Vector). Jest to zdecydowanie najmniej efektywny tryb z badanych. SIV jest trybem dwuprzebiegowym, dodatkowo nie może on działać w trybie “online”. To znaczy, że aby rozpocząć szyfrowanie należy załadować całą wiadomość do pamięci (w przeciwieństwie do np. trybu GCM), a sama wiadomość przetwarzana jest dwukrotnie. SIV najczęściej służy jako *wrapper* kluczy, które zwykle są małych rozmiarów przez co stosunkowo długi czas przetwarzania nie jest w tym przypadku problemem.

2. Badanie odporności trybów na modyfikacje i manipulacje szyfrogramu

W tym badaniu zaszyfrowana wiadomość poddana zostanie pewnym modyfikacjom polegającym na zmianie ilości bloków, usuwaniu pojedynczych bitów itp. W tym celu wykorzystane zostaną zaimplementowane przeze mnie trzy tryby pracy - ECB, CBC, PBC. Zbadane zostanie jak zachowuje się szyfr podczas deszyfrowania wiadomości, czy wiadomość daje się odczytać i jak różni się ona w stosunku do oryginału. Dzięki temu dowiemy się jak odporny na manipulacje i zmiany jest sam szyfr.
Operacje jakie wykonamy na szyfrogramach to:

1. usunięcie całego bloku,
2. powielenie wybranego bloku,
3. zamiana bloków miejscami,
4. dodanie zupełnie nowego bloku szyfrogramu,
5. zmiana wartości jednego (bitu, bajtu) w bloku,
6. zamiana wewnątrz bloku bajtów miejscami,
7. usunięcie fragmentu bloku (np. 1 bajtu).

[1] “[...] Galois/Counter Mode (GCM) is a mode of operation for symmetric-key cryptographic block ciphers which is widely adopted for its performance. [...]” - Wikipedia

Zachowanie szyfru w danym trybie pracy testowane będzie za pomocą trzech wiadomości wejściowych - znanego tekstu, ciągu tych samych znaków (w tym przypadku litery “a”) oraz tekstu składającego się z kolejnych liter alfabetu (powtórzonego kilkukrotnie w celu rozciągnięcia wiadomości do kilku bloków)

2.1. Usunięcie całego bloku

Z szyfrogramu długości kilku bloków usunięty zostanie drugi w kolejności blok. Następnie szyfrogram zostanie znów połączony w ciąg i odszyfrowany za pomocą tego samego trybu pracy oraz tego samego klucza

<table>
<tr><td>Operacja:</td><td></td><td>Usunięcie bloku z szyfrogramu</td></tr>
<tr><td>Wiadomość:</td><td></td><td>Ala ma kota ale nie tylko bo posiada tez psa, a prolog to najlepszy jezyk programowania</td></tr>
<tr><td rowspan="3">Tryb pracy</td><td>ECB</td><td>Ala ma kota ale iada tez psa, a prolog to najlepszy jezyk programowania</td></tr>
<tr><td>CBC</td><td>Ala ma kota ale \xe3h\xee\xfb\xcb\xca\x9f\xe6\xea\xac\x82N\x8a\x02\xd8kprolog to najlepszy jezyk programowania</td></tr>
<tr><td>PBC</td><td>Ala ma kota ale Fd`a9l)z $p.m<ks_wklv\x7fltotm.+po#\\\x7f} s}6ykts=.{x2Bjsawq-\t\t]\nFH\x15\x03Z</td></tr>
</table>

<table>
<tr><td>Operacja:</td><td></td><td>Usunięcie bloku z szyfrogramu</td></tr>
<tr><td>Wiadomość:</td><td></td><td>aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa</td></tr>
<tr><td rowspan="3">Tryb pracy</td><td>ECB</td><td>aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa</td></tr>
<tr><td>CBC</td><td>aaaaaaaaaaaaaaaa\xbcH\xccj\x9c\xdb!C\xb7\x02\xc5u\xb5F\x88gaaaaaaaaaaaaaaaaaaaaaaaaaaa</td></tr>
<tr><td>PBC</td><td>aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa</td></tr>
</table>

<table>
<tr><td>Operacja:</td><td></td><td>Usunięcie bloku z szyfrogramu</td></tr>
<tr><td>Wiadomość:</td><td></td><td>abcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz</td></tr>
<tr><td rowspan="3">Tryb pracy</td><td>ECB</td><td>abcdefghijklmnophijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz</td></tr>
<tr><td>CBC</td><td>abcdefghijklmnopS.\xdd?\xa31\xf7lY\xc2\x1a[0\xeeU\xd0yzabcdefghijklmnopqrstuvxyz</td></tr>
<tr><td>PBC</td><td>abcdefghijklmnopxyz{|}q~cz{|}~\x7foijqrstzwtc`ebgdy\x7f`abcdjgkrs\n\x0c\x0e\x0c\x12</td></tr>
</table>

ECB - ponieważ nie ma powiązania pomiędzy blokami, pozostałe bloki zostały normalnie rozszyfrowane. Końcowa wiadomość po prostu nie posiada usuniętych znaków

CBC - blok *i* wpływa na *i+1* blok. Pierwszy blok został poprawnie rozszyfrowany, następnie poprzez usunięcie drugiego bloku, blok trzeci został rozszyfrowany na podstawie bloku pierwszego co spowodowało, że rozszyfrowany blok trzeci to losowe bajty. Dalsze bloki zostały rozszyfrowane bez problemu, ponieważ bazują one na (poprawnie) zaszyfrowanej a nie (błędnie) odszyfrowanej wiadomości

PBC - tryb ten również zależy od poprzedniego bloku, ale w przeciwieństwie do CBC jest to blok po rozszyfrowaniu. Pierwszy blok rozszyfrowano poprawnie, blok drugi usunięto, więc blok trzeci został rozszyfrowany bazując na bloku pierwszym, co spowodowało że blok nie został poprawnie odszyfrowany. Ponieważ kolejne bloki bazowały na błędnie odszyfrowanej wiadomości, dalsza wiadomość również została źle odszyfrowana. Ciekawym przykładem jest ciąg samych liter, które mimo usunięcia bloku nie spowodowały dalszej kaskady błędów w odszyfrowywaniu wiadomości. Zadziało się tak dlatego, że odszyfrowany tekst bloku pierwszego jest taki sam jak usuniętego bloku drugiego, przez co dalsze odszyfrowywanie odbyło się bez przeszkód.

2.2. Zduplikowanie jednego bloku

Pierwszy blok został powielony

Operacja:		**Zduplikowanie jednego bloku**
Wiadomość:		**Ala ma kota ale nie tylko bo posiada tez psa, a prolog to najlepszy jezyk programowania**
Tryb pracy	ECB	Ala ma kota ale Ala ma kota ale nie tylko bo posiada tez psa, a prolog to najlepszy jezyk programowania
	CBC	Ala ma kota ale \x90)\xfeveo\x84\xb3\x97\x07\xe6\xb4.\xe3\x94\xdanie tylko bo posiada tez psa, a prolog to najlepszy jezyk programowania
	PBC	Ala ma kota ale \xc8\xcd\x84\xca01\xb5\x807G]]\xf4\x8e\x01\xc3\xe7\xc8\x80\xca))\xf9\x807\x13^\x12\xb5\x92\x0b\x90\xe0\xc0\x81\x8b}$\xf0\x91x(...dalej losowe bajty

Operacja:		**Zduplikowanie jednego bloku**
Wiadomość:		**aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa**
Tryb pracy	ECB	aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
	CBC	aaaaaaaaaaaaaaaaB\x96@%N$\xf8\x9coL\xe3\x02\x81R\x16\xfaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
	PBC	aaaaaaaaaaaaaaaa\xc8\xcd\x84\xca01\xb5\x807G]]\xf4\x8e\x01\xc3\xc8\xcd\x84\xca01\xb5\x807G]]\xf4\x8e\x01\xc3\xc8\xcd\x84\xca01\xb5\x807G ...dalej losowe bajty

Operacja:		**Zduplikowanie jednego bloku**
Wiadomość:		**abcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz**
Tryb pracy	ECB	abcdefghijklmnopabcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz
	CBC	abcdefghijklmnop\xa90a\xd2r\xd2/nm\xbd&\xbb\x9e\xb7\xe2\x12qrstuvxyzabcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz
	PBC	abcdefghijklmnop\xc8\xcd\x84\xca01\xb5\x807G]]\xf4\x8e\x01\xc3\xd8\xdd\x94\xda !\xaa\x91$LTR\xfd\x85\x08\xd4\xc1\xc6\x8d\xc59:\xbc\x87.\\DB\ ...dalej losowe bajty

ECB - powielony blok nie wpłynął znacząco na odszyfrowaną wiadomość, część tekstu która była zaszyfrowana w zduplikowanym bloku również została rozszyfrowana. Wynik przypomina skopiowanie i powielenie części tekstu w środku wiadomości

CBC - podobna sytuacja jak podczas usunięcia bloku. Tylko drugi (zduplikowany) blok został niepoprawnie rozszyfrowany. Warto zauważyć, że nie utraciliśmy żadnej informacji w stosunku do oryginalnej wiadomości. Powielony blok raz został odszyfrowany poprawnie, a raz nie.

PBC - W tym przypadku znów powstała nam kaskada błędów nie pozwalająca odczytać pozostałej wiadomości. Tryb PBC szyfruje pierwszy blok za pomocą tzw. wektora inicjującego. Dopiero kolejne bloki odszyfrowywane są za pomocą wiadomości z poprzednich bloków. Spowodowało to, że dla tekstu składającego się z tych samych liter nie powstała nam anomalia polegająca na odczytaniu dalszych bloków w sposób poprawny. Gdyby powielony został inny blok niż pierwszy, wiadomość wyglądałaby podobnie do tej z pkt. 2.1.

2.3. Zamiana dwóch bloków miejscami

Drugi i trzeci blok szyfrogramu zostały zamienione miejscami

Operacja:		**Zamiana dwóch bloków miejscami**
Wiadomość:		**Ala ma kota ale nie tylko bo posiada tez psa, a prolog to najlepszy jezyk programowania**
Tryb pracy	ECB	Ala ma kota ale iada tez psa, a nie tylko bo posprolog to najlepszy jezyk programowania
	CBC	Ala ma kota ale \xa0.\x11\xcbZO\xa7H\x7f\xcal\xa8tM\xb9\xf67C\t\x13\x83H\xb2\xb5\x0f\x93)1h\xcb_\xf0\xe0\x17v\xf5\xe2m<\x98P):\xf6z\xba\x8d%szy jezyk programowania
	PBC	Ala ma kota ale Fd`a9l)z $p.m<ksiada tez psa, a prolog to najlepszy jezyk programowania

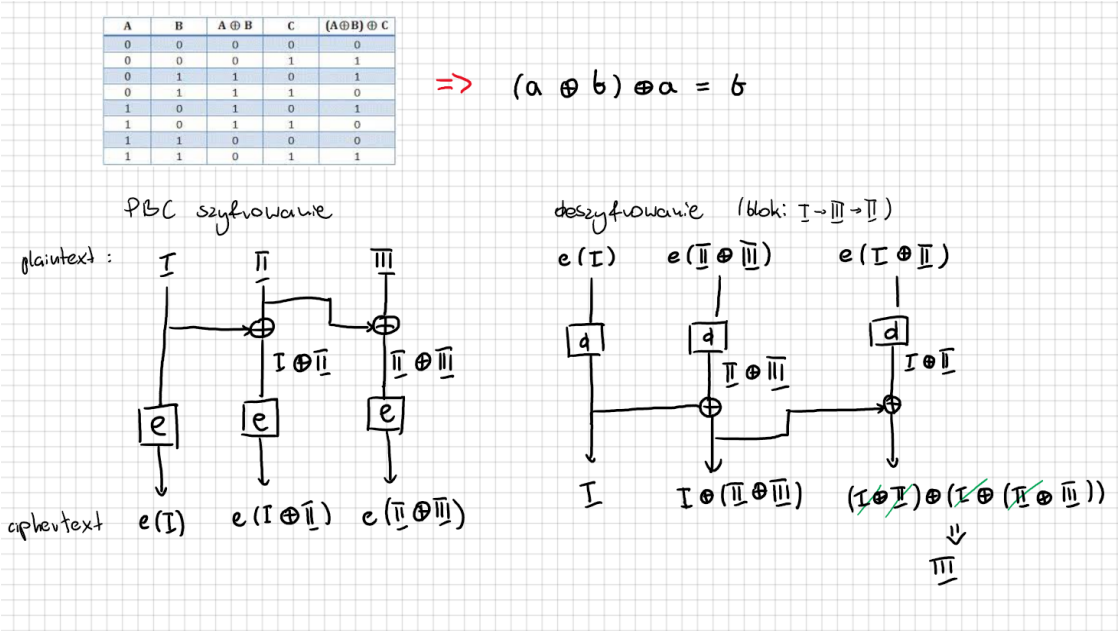
Operacja:		**Zamiana dwóch bloków miejscami**
Wiadomość:		**aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa**
Tryb pracy	ECB	aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
	CBC	aaaaaaaaaaaaaaaa\xd8t\xbc]\xfda2\xcd7s\xd5z\x076\xf0N\xf0\xaez~:Y\x18+\xbe\t\xc2\x92\xef60\xd5I\xbb\xa7B\xa6YK\x87\xe8\x1bv\x89\x89a\xa1\xfaaaaaaaaaaa
	PBC	aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

<table>
<tr><td>Operacja:</td><td></td><td>Zamiana dwóch bloków miejscami</td></tr>
<tr><td>Wiadomość:</td><td></td><td>abcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz</td></tr>
<tr><td rowspan="3">Tryb pracy</td><td>ECB</td><td>abcdefghijklmnophijklmnopqrstuvxqrstuvxyzabcdefgyzabcdefghijklmnopqrstuvxyz</td></tr>
<tr><td>CBC</td><td>abcdefghijklmnop\xdd6\xe5\xb0\xff\xd9w\xb3\x99\x84\x01\x11S\xe3\xb1\xbb}\xdb\x17\x8an\xdcW\xc3#\x81\xf6\x80L8\xe0\xd1\xc0\x8c\x8aG\xebzS\x00\xd7}\x8e\xebd\xa7,\x1bopqrstuvxyz</td></tr>
<tr><td>PBC</td><td>abcdefghijklmnopxyz{|}q~cz{|}~\x7fohijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz</td></tr>
</table>

ECB - w tym trybie standardowo rozszyfrowano wszystkie bloki, część wiadomości jest w złej kolejności, dokładnie tak jak zamienione były bloki

CBC - ponieważ drugi i trzeci blok zostały zamienione miejscami, nie udało się ich poprawnie rozszyfrować. Drugi blok został rozszyfrowany bazując na szyfrogramie trzeciego bl. (a powinien pierwszego bloku), trzeci blok zamiast bazować na bloku drugim, bazował na pierwszym. Nie udało się również rozszyfrować bloku czwartego, który bazował na bloku drugim.

PBC - w tym przypadku da się zaobserwować bardzo ciekawą anomalię. Zauważmy, że blok trzeci (znajdujący się na drugiej pozycji) nie został poprawnie odszyfrowany, ale blok, który oryginalnie był drugi (ale wskutek zamiany znalazł się na trzeciej pozycji) po odszyfrowaniu zawiera wiadomość którą oryginalnie szyfrował blok trzeci. Innymi słowy blok drugi został tak odszyfrowany, że zmienił się w pasujący blok trzeci. Zadziało się tak wskutek znoszących się wzajemnie operacji XOR, co dokładnie zaprezentuję na schemacie poniżej:



* w celu uproszczenia schematu pominięty został wektor inicjujący

Kolejną anomalią trybu PBC jest to, że tekst składający się z tych samych liter został pomimo zamiany bloków w całości poprawnie odszyfrowany. Podobni jak w punkcie 2.1, odszyfrowany tekst drugiego bloku jest taki sam jak trzeciego bloku przez co nie powodował on błędów w kolejnych blokach

2.3. Dodanie nowego bloku do szyfrogramu

Do szyfrogramu na drugiej pozycji dodano nowy blok z zaszyfrowaną wiadomością o treści "*-thisisnewblock-*"

Operacja:		**Dodanie nowego bloku do szyfrogramu**	
Wiadomość:		**Ala ma kota ale nie tylko bo posiada tez psa, a prolog to najlepszy jezyk programowania**	
Tryb pracy	ECB	Ala ma kota ale -thisisnewblock-nie tylko bo posiada tez psa, a prolog to najlepszy jezyk programowania	
	CBC	Ala ma kota ale \x18[\xf0\xbanSL\xa0s\xee\xda\r\xc2\xfa\xcdP\x8d\xbc\xb4\x95\x81T\x7f\xe8\x91\xa4wf\xc8R)\x0eiada tez psa, a prolog to najlepszy jezyk programowania	
	PBC	Ala ma kota ale l\x18\tl\x1e\x08S\x05\n\x03\x03L\x0e\x0f\x0e\rC\x1d\rl\x07\x10\x1f\x05\nW\x00\x03O\x13\x04^D\x15\x0c\x08S\x1d\x16\x14E\x07\x11\rCC\n\r]\x06\x07\x05\x1c\x0eS\x1a\n\	...dalej losowe bajty

Operacja:		**Dodanie nowego bloku do szyfrogramu**	
Wiadomość:		**aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa**	
Tryb pracy	ECB	aaaaaaaaaaaaaaaa-thisisnewblock-aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	
	CBC	aaaaaaaaaaaaaaaa\x83D\xe8\xd9\x89\x0f\xf8\x85\xc1\xe3Z\xce\x18-\xa2\xc6\x19\xab\xa8\xb7s\x10\xc6\xc7-\xe8\xf4\xabS\x94H\x8aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	
	PBC	aaaaaaaaaaaaaaaaL\x15\t\x08\x12\x08\x12\x0f\x04\x16\x03\r\x0e\x02\nLL\x15\t\x08\x12\x08\x12\x0f\x04\x16\x03\r\x0e\x02\nLL\x15\t\x08\x12\x08\x12\x0f\x04\x16\x03\r\x0e\x02\nLL\x15\t\	...dalej losowe bajty

Operacja:		**Dodanie nowego bloku do szyfrogramu**	
Wiadomość:		**abcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz**	
Tryb pracy	ECB	abcdefghijklmnop-thisisnewblock-qrstuvxyzabcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz	
	CBC	abcdefghijklmnop\x00@pVW\xd1\xf9h\xd1\r@\x8c\xdd\xdd\x1e\xfe\x8a\xbc"-\xb9\xd9\xde2&\x06\xed\xeb\x93`\xf3\xb4hijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz	
	PBC	abcdefghijklmnopL\x16\x0b\r\x16\x0f\x14\x06\x0c\x1d\t\x00\x02\r\x04]\\\x06\x1b\x1d\x06\x1f\x0b\x17\x1f\x16\x00\x0f\x0b\x06\rJE\x1d\x02\x02\x1f\x04\x1d\x01\x15\x06\x10\x1f\x1b\x16\x1d\	...dalej losowe bajty

ECB - tryb ECB ponownie bez problemu odszyfrował wszystkie bloki. Da się zauważyć treść wstawionego bloku w środku oryginalnego tekstu

CBC - w tym trybie niepoprawnie odszyfrowano blok drugi (z wstawioną wiadomością) oraz blok kolejny. Nowy blok z wiadomych względów został źle odszyfrowany - jego szyfrogram nie uwzględniał poprzedzającego go bloku pierwszego. Blok trzeci, za nowo wstawionym blokiem również nie udało się odszyfrować. Podczas szyfrowania zależał on od szyfrogramu pierwszego bloku, a podczas odszyfrowywania poprzedzającym go blokiem nie był blok pierwszy, a nowo wstawiony blok przez co nie udało się go odczytać w sposób prawidłowy. Pozostałe bloki zostały odszyfrowane prawidłowo.

PBC - pierwszy blok odszyfrowano poprawnie. Nowo wstawiony blok spowodował natomiast kaskadę błędów analogicznie do tej opisanej w punkcie 2.1. Dalsza wiadomość to ciąg losowych bajtów nie pozwalających na odczytanie oryginalnej wiadomości.

2.4 Zmiana jednego bitu szyfrogramu

Zmieniono bit znajdujący się na 147 pozycji szyfrogramu, przez co trzeci bajt drugiego bloku uległ zmianie.

Operacja:		**Zmiana jednego bitu szyfrogramu**	
Wiadomość:		**Ala ma kota ale nie tylko bo posiada tez psa, a prolog to najlepszy jezyk programowania**	
Tryb pracy	ECB	Ala ma kota ale y\\\n\xb0\x8d[08C\xc1+\xc7\xe77\xbe\xe7iada tez psa, a prolog to najlepszy jezyk programowania	
	CBC	Ala ma kota ale \x0c\xccc\x1d\x96^\x83mN\xb1E\xe2Z\x9d\x91?iaDa tez psa, a prolog to najlepszy jezyk programowania	
	PBC	Ala ma kota ale \x99)\x14\x8d\xa7P\xc7J\xb7\xfc\xbc\xeba<\xc5E\x9e!\x15\xcc\xf3]\xce[\xf8\xac\xad\xe5ml\xcb\x16\x872\x1e\xc1\xbcN\x8bU\xb7\	...dalej losowe bajty

Operacja:		**Zmiana jednego bitu szyfrogramu**	
Wiadomość:		**aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa**	
Tryb pracy	ECB	aaaaaaaaaaaaaaaa\xae\xbeH\x8a\xf7\x7fca\x17\xf6\x8f\xb8X\x18\xa3>aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	
	CBC	aaaaaaaaaaaaaaaa\x93\x0f\x1aT\xf6L\xa3e\x1d\r\xc5{\xd2\x1f*\x9eaaAaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	
	PBC	aaaaaaaaaaaaaaaa:\x92\xf6\x87\xc4c[\x05\x13\x92\xefS\x8c\xd1\x14\xfe:\x92\xf6\x87\xc4c[\x05\x13\x92\xefS\x8c\xd1\x14\xfe:\x92\xf6\x87\xc4c[\	...dalej losowe bajty

Operacja:		**Zmiana jednego bitu szyfrogramu**	
Wiadomość:		**abcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz**	
Tryb pracy	ECB	abcdefghijklmnop\xcb\xe4\xb4<\t@\xcd{\x9e\xf5\xb6g\xa1\xba\xaa\x8dhijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz	
	CBC	abcdefghijklmnop9\x8d\x8aX\xd7\x82\xb1\xd0[(,\n\xdf\xd8\xc7\xd1hiJklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz	
	PBC	abcdefghijklmnop\xba\xd3\xa9\xbf\xa1\xfa\xfe\xba%"\x12\x18\x12\x92\xc1 \xa3\xc8\xb0\xa0\xb8\xe1\xe8\xac/7\x02\x08\x02\x82\xd1?\xb2\xdb\xbb	...dalej losowe bajty

ECB - zmiana jednego bitu, a co za tym idzie całego bajtu w drugim bloku spowodowała, że nie udało się go poprawnie rozszyfrować. Tak drobna zmiana spowodowała propagacje błędu na cały blok. Pozostałe bloki zostały oczywiście odszyfrowane bez problemu

CBC - w tym przypadku wydawać by się mogło, że skoro drugi blok szyfrogramu został zmieniony (i przez to błędnie odszyfrowany), to CBC rozpropaguje owy błąd na blok następny. Tak właśnie się stało, ale w sposób bardzo subtelny. Zauważmy, że trzeci blok został odszyfrowany w sposób poprawny, z wyjątkiem trzeciej litery owego bloku. Zmieniła

się ona z małej litery na wielką. W CBC poprzedni blok szyfrogramu jest łączony z obecnym operacją XOR. XOR działa w sposób bitowy, błąd przeniósł się wyłącznie na 19 bit bloku, a więc odszyfrowana wiadomość została zmieniona tylko w tym miejscu. Tak się złożyło, że jest to trzeci bajt trzeciego bloku i to właśnie on różni się od oryginalnej wiadomości.

PBC - w przeciwieństwie do CBC, tu odszyfrowany (a nie jeszcze zaszyfrowany) blok łączony jest operacją XOR z następnym blokiem. Jeden błędny bit szyfrogramu po odszyfrowaniu rozpropagował się na cały blok. Co za tym idzie, blok następny który łączy się z blokiem poprzednim już po odszyfrowaniu spowodował, że całe dalsze odszyfrowywanie obarczone było błędem. Powstała klasyczna już kaskada, przez co nie da się poprawnie odczytać rozszyfrowanej wiadomości.

2.5 Zamiana bajtów miejscami w bloku

Zamieniony zostanie trzeci oraz czwarty bajt drugiego bloku, czyli 19. oraz 20. bajt szyfrogramu

<table>
<tr><td>Operacja:</td><td></td><td>Zamiana bajtów miejscami w bloku</td><td></td></tr>
<tr><td>Wiadomość:</td><td></td><td>Ala ma kota ale nie tylko bo posiada tez psa, a prolog to najlepszy jezyk programowania</td><td></td></tr>
<tr><td rowspan="3">Tryb pracy</td><td>ECB</td><td>Ala ma kota ale \x93\x01\xcd[\n\xbf\xfd\x97\xa2\xe9\xaa%\x91\xf6vriada tez psa, a prolog to najlepszy jezyk programowania</td><td></td></tr>
<tr><td>CBC</td><td>Ala ma kota ale k'\x95\x85S\xa2R\x82\xf7Q\x08\xef>\n\xdc^ia\xa5\xa0 tez psa, a prolog to najlepszy jezyk programowania</td><td></td></tr>
<tr><td>PBC</td><td>Ala ma kota ale \x97\x1c~\xf51\xb1u\x06\xbc\x1e%\xe2\xed\xe50\x0e\x90\x14\x7f\xb4e\xbc|\x17\xf3N4\xec\xe1\xb5>]\x89\x07t\xb9*\xaf9\</td><td>...dalej losowe bajty</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>Operacja:</td><td></td><td>Zamiana bajtów miejscami w bloku</td><td></td></tr>
<tr><td>Wiadomość:</td><td></td><td>aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa</td><td></td></tr>
<tr><td rowspan="3">Tryb pracy</td><td>ECB</td><td>aaaaaaaaaaaaaaaavB\xd9\xb92\xf8\x81)\x87\xff\xbc\x9d\x12\x9b:\xbcaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa</td><td></td></tr>
<tr><td>CBC</td><td>aaaaaaaaaaaaaaaag\x8e\xde\xb7\x9c\xc0\x12G\x94\x06W\xaef\x96\x04\xfaaa\x9b\x9baaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa</td><td></td></tr>
<tr><td>PBC</td><td>aaaaaaaaaaaaaaaa\xcc\xfa\xc0\x93$\n;_B\xbd\x13[\xf6\x83\xbd\xb5\xcc\xfa\xc0\x93$\n;_B\xbd\x13[\xf6\x83\xbd\xb5\xcc\xfa\xc0\x93$\n;_</td><td>...dalej losowe bajty</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>Operacja:</td><td></td><td>Zamiana bajtów miejscami w bloku</td><td></td></tr>
<tr><td>Wiadomość:</td><td></td><td>abcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz</td><td></td></tr>
<tr><td rowspan="3">Tryb pracy</td><td>ECB</td><td>abcdefghijklmnop\r\x1c`k[w\x84\xdd\x99\x99\xfd\x13S\xe2\x90\xb7hijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz</td><td></td></tr>
<tr><td>CBC</td><td>abcdefghijklmnop\xf7\x1bR[+t\xf3\xf3#(z\xa3\xda\xa7T\xd2hi\xa9\xa8lmnopqrstuvxyzabcdefghijklmnopqrstuvxyz</td><td></td></tr>
<tr><td>PBC</td><td>abcdefghijklmnop\x0bl<\x8dJp\x02\x92\x8d\xa8%\xe9\xe0f*\x86\x12w%\x92Sk\x14\x84\x87\xb85\xf9\xf0v:\x99\x03d.\x9b\\b\x1f\x8d\x90\x</td><td>...dalej losowe bajty</td></tr>
</table>

ECB - zamiana trzeciego i czwartego bajtu spowodowała nieprawidłowe odszyfrowanie bloku. Pozostałe bloki odszyfrowano bezbłędnie.

CBC - w tym przypadku sytuacja wygląda podobnie do opisanej w punkcie 2.4, drugi blok został odszyfrowany całkowicie niepoprawnie, natomiast w przypadku bloku trzeciego błąd został rozpropagowany wyłącznie na zamienione bajty, tj. trzeci oraz czwarty bajt bloku.

PBC - również analogicznie od punktu 2.4. Po drugim bloku nastąpiła kaskada błędów uniemożliwiająca poprawne odszyfrowanie dalszej części szyfrogramu

2.6 Usunięcie fragmentu bloku

Z bloku szyfrogramu zostanie usunięty jeden bajt. Jest to dokładnie 12. bajt drugiego bloku. Z uwagi na to, że testowane tryby wymagają wyrównania szyfrogramu do pełnego bloku, na koniec wiadomości wstawiony zostanie bajt wyrównujący, który podczas interpretacji wyników zostanie po prostu zignorowany

Operacja:		**Usunięcie fragmentu bloku**	
Wiadomość:		**Ala ma kota ale nie tylko bo posiada tez psa, a prolog to najlepszy jezyk programowania**	
Tryb pracy	ECB	Ala ma kota ale \x1f\xa4\x96\x83=\x87\xebv\x17&`fB!\xc1\xa9~\xb62\xfa\xcdN\x9fo\x9e\xda\xc5\xe2\x83\xea\xd7u\x83\xcb\x90\x15\x97R\x18\x9c\xba\x9eH\x9f7\xc1\t\xc7	...dalej losowe bajty
	CBC	Ala ma kota ale \xb6\x98\x9fS\xc6\xa11 q\xfd\xf5 \x13\xe2x\xde/m\x10\xc1\xab\xe8\xf8\x17\xedp\x0c\xba\x9c}\xd9\xf5\xec\x01\xc4\xc3\xf6\xf8\x99\xbb\x0cC\x83d!~V\x02\	...dalej losowe bajty
	PBC	Ala ma kota ale)!\n=|\xe2}\x96\xff,-\xb5F\tL\xc8-q\xdc\xb8(q]\tHr\xcbe\xbeM\xe3\xeeS\xde$/\xac\x02\x99\xfefc\xec\xe0o\xa8\x1dz\xac\xf9U\xd2j0\xecy\xe1\xe0\xe4$\x00\	...dalej losowe bajty

Operacja:		**Usunięcie fragmentu bloku**	
Wiadomość:		**aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa**	
Tryb pracy	ECB	aaaaaaaaaaaaaaaa\xd0\xf3\xe2w6\xcfGc\x8a\xc2\x0f\xa3\xdb\xecl`\xb9\xcd\xf4\xf0\xfb\x1f(\x9aE\x9d\t\x90\xdb\x145\xc0_$iP\x8f\xc6i\x85/\xf1\x1ezhC\x16K[\x03L\x00\x1	...dalej losowe bajty
	CBC	aaaaaaaaaaaaaaaa\xf6R\xa8.U\xd2\x9a\xef\x0f\x9d\xb7\xba\xad\x97\xaf\xd8\x1e\xa6wh{\x18~\xe1\x01\xcd`\x1fY\xbe+u\x0b\x10\xcf\x86\x89\x14\xf8\x19\xbeG6w\xbe)dg	...dalej losowe bajty
	PBC	aaaaaaaaaaaaaaaa\xd1\x8c)N\x05\xb4M\x9dTo\xf1\x85\xeb\xb1\xfa'\x1f\xfd\xc0\xc6\xaef\x92\xac\xfb\x051n\xe1\xc1\xa2\xc5(\x94\xf0*!yL\xda\xff \xaf<uz\xa4\xd2/&\xc3e\	...dalej losowe bajty

Operacja:		**Usunięcie fragmentu bloku**	
Wiadomość:		**abcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyzabcdefghijklmnopqrstuvxyz**	
Tryb pracy	ECB	abcdefghijklmnop\xa1(\x8a\x82_\xab\xceu\xd5\xa9\x04H*Z\xff\xe0h\xfft\xb3\x08S\xc4\xd5_\x00c\x04\x165Y\xd7\xf3\xf1\xf6f\xb9rB\xb9&\xbcV%t\xcbh\x7f J\xb1\xe1\xbeG'	...dalej losowe bajty
	CBC	abcdefghijklmnop\x9d\xf8b"\x02\xb0+\xaf\xf7Q\xa3b\xf0&O@\xd5\xea\xbd\xf0\xa5\xfdS\x02\xed\xa8\x964\xb7\xd2\xc3\xe3)\xb0\xdd\x8f\x14\xbc\x01+\x17\xb1\x8cE]4X\xb	...dalej losowe bajty
	PBC	abcdefghijklmnop\xba\xbe\xc1j\xf0\xd3\x02\xc1zO\xf0:\xc0\x8b\xc1r\xcb\xa7\xea\x07D\xff7}YE\x83\x84\x11\x0f\xd9Y\x14\xc4\xbc\x82\xba\xc4\x1b\xd0\x02t\xc7\xf5\xca~#	...dalej losowe bajty

ECB - pierwszy blok został rozszyfrowany prawidłowo, natomiast cała dalsza wiadomość rozpoczynając od bloku drugiego, czyli bloku w którym usunięto bajt nie nadaje się do odczytania. Powodem nie występującej wcześniej w ECB kaskady błędów jest to, że usunięcie jednego bajtu ze środka spowodowało przesunięcie wszystkich dalszych bajtów o jeden do przodu. Zmieniły się "granice" bloków przez co szyfr ni ejest w stanie poprawnie rozszyfrować żadnego z bloków.

CBC, PBC - w tym przypadku, oba tryby zachowują się identycznie. Z uwagi na to, że opierają się one na trybie ECB, ich wyniki również są niepoprawne.

3. Wnioski końcowe

Przetestowane powyżej tryby pracy, mimo podobieństw w konstrukcji zachowywały się z różny sposób. Warto na sam początek wspomnieć o najprostszym trybie jaki pojawił się w tym badaniu - trybie ECB.

ECB posiada bardzo istotna cechę, która znacząco odróżnia go od pozostałych dwóch - kolejne bloki szyfrogramu nie są ze sobą w żaden sposób związane. Dzięki tej własności byliśmy w stanie zobaczyć wszystkie przeprowadzone manipulacje "jak na dłoni". Jest to bardzo mało bezpieczne rozwiązanie. Sami doświadczyliśmy, jak łatwo chociażby zmienić lub usunąć bloki szyfrogramu bez powodowania rażących błędów, które mogły by świadczyć o tym, że szyfrogram został zmanipulowany. Tryb ten nie powinien być wykorzystywany do szyfrowania wiadomości i zdecydowanie lepiej jest użyć innego trybu, na przykład któregoś z pozostałych jakie znalazły się w badaniu.

Kolejnym trybem jaki został wzięty pod lupę jest tryb CBC. Wykorzystuje on wektor inicjujący do zaszyfrowania pierwszego bloku. W przypadku kolejnych bloków wykorzystany jest szyfrogram bloku poprzedniego. Wymusza to odpowiednią kolejność bloków i powoduje, że ciężej odczytać zmanipulowaną wiadomość. Bazowanie na szyfrogramie poprzednika (a nie odszyfrowanej wiadomości) powoduje, że on jest mniej skłonny do tworzenia kaskady błędów niż tryb PBC. Zduplikowanie bloku spowoduje niepoprawny odczyt tylko tego bloku, dalsze bloki zostaną odszyfrowane bez zmian.

Tryb PBC posiada chyba najciekawsze właściwości ze wspomnianej trójki badanych trybów. Zdawać by się mogło, że ponieważ poprzedni blok bazuje nie na szyfrogramie, a na odszyfrowanej wiadomości to zapewnia dodatkową warstwę obrony. Na podstawie przeprowadzonych badań da się zauważyć, że ma on rzeczywiście skłonności do

wywoływania kaskady błędów nie pozwalających na odczytanie zmanipulowanej wiadomości. Jest to bardzo dobra cecha z perspektywy właściciela pliku. Jeżeli ktoś będzie w sposób nieumiejętny próbował manipulować szyfrogramem, to nawet przy odgadnięciu klucza po odszyfrowaniu spotka go jedynie ciąg losowych bajtów. Niestety, jest jeszcze druga strona medalu. Podczas badań udało mi się zauważyć kilka właściwości, które mogą zostać wykorzystane do złamania szyfru. Jedną z nich jest to, że jeżeli poprzedni blok szyfruje tą samą wiadomość (na przykład same litery “a”, tak jak w naszym badaniu), to niektóre manipulacje szyfrogramu produkują wynik identyczny do wyniku np. trybu ECB. Możemy manipulować blokiem danych w taki sposób, aby nie nastąpiła kaskada błędów - w ten sposób wiemy, jaka wiadomość szyfrowana jest przez blok poprzedni. Anomalia ta pokazana jest w punkcie 2.1, gdzie pomimo usunięcia całego bloku, przy pewnych okolicznościach nie zaobserwowano propagacji błędu na dalsze bloki.

Szyfr AES posiada wiele więcej trybów niż te, które przedstawiłem powyżej. Każdy tryb posiada inną wydajność oraz jest odporny na inne manipulacje i próby złamania szyfru. Warto mieć świadomość ogromu możliwości w dobraniu odpowiednich narzędzi, które najlepiej zaspokoją nasze potrzeby.