

Laboratorium ochrony danych

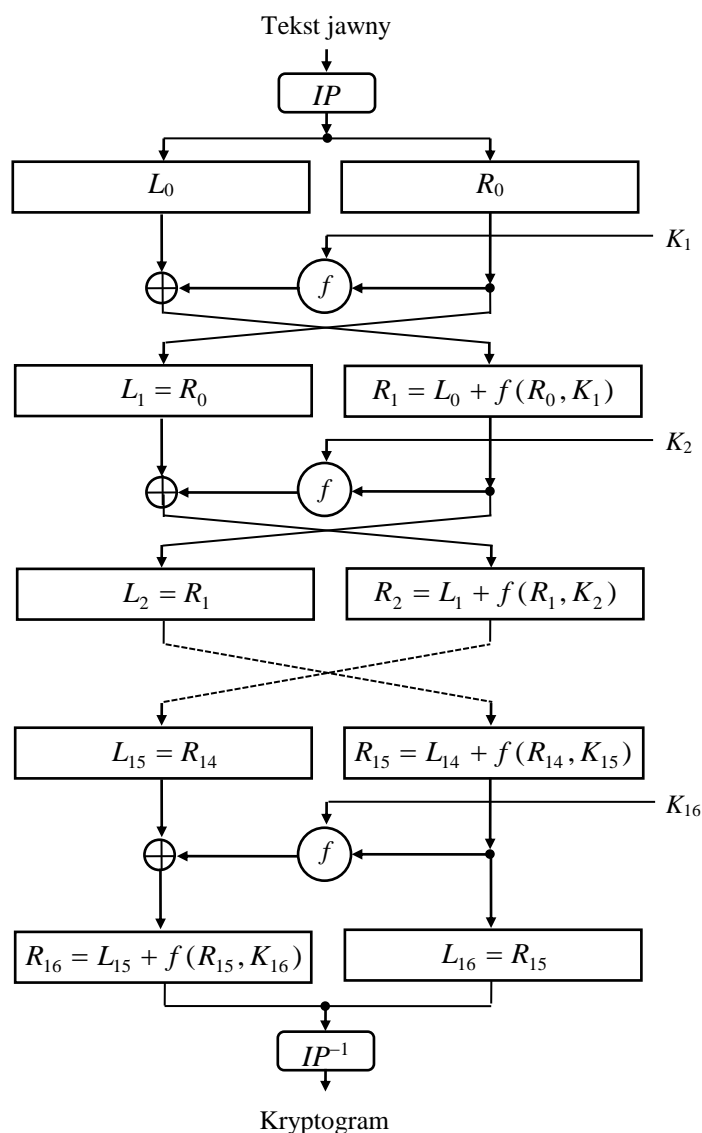
Ćwiczenie nr 4

Temat ćwiczenia: Algorytm kryptograficzny DES

Cel dydaktyczny: Poznanie metod szyfrowania i deszyfrowania informacji za pomocą algorytmu symetrycznego z kluczem tajnym na przykładzie algorytmu Data Encryption Standard (DES). Zastosowanie algorytmu DES do szyfrowania i deszyfrowania plików w trybie elektronicznej książki kodowej ECB oraz w trybie wiązania bloków CBC.

Wprowadzenie teoretyczne

Opis algorytmu DES



Rys. 1. Schemat blokowy algorytmu DES

Podstawą działania algorytmu DES oraz innych algorytmów symetrycznych jest struktura nazywana **siecią Feistela**, która została opublikowana przez pracownika IBM Horsta Feistela.

Sieć (schemat) Feistela pozwala na szyfrowanie i deszyfrowanie informacji tym samym algorytmem, mimo iż sama funkcja szyfrująca f nie jest odwracalna. Sieć Feistela generuje z tekstu jawnego szyfrogram, a z szyfrogramu tekst jawny. Jej zastosowanie znacznie uprościło konstruowanie algorytmów szyfrujących, gdyż nie trzeba się troszczyć o odwracalność funkcji szyfrującej f .

Tekst jawny dzieli się na dwa równe bloki L_i oraz R_i . Funkcja f jest właściwym algorytmem szyfrującym. Jako wynik otrzymuje się szyfrogram, który jest wykorzystywany w kolejnej rundzie. Numer kolejnej rundy oznaczany jest indeksem i , to oznacza iż wynik szyfrowania jest ponownie i -krotnie szyfrowany, co polepsza jakość szyfrowania. Algorytmy zbudowane na bazie sieci Feistela: DES, 3DES, Twofish, FEAL.

Algorytm Data Encryption Standard jest iloczynowym algorytmem kryptograficznym wprowadzonym w 1977 r. przez Narodowe Biuro Normalizacji w USA. Algorytm DES jest używany do szyfrowania i deszyfrowania danych w postaci bloków 64-bitowych. Schemat blokowy algorytmu pokazano na Rys. 1.

Blok wejściowy 64-bitowy jest poddany permutacji początkowej IP zgodnie z poniższą tabelą:

IP															
58	50	42	34	26	18	10	02	60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06	64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01	59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05	63	55	47	39	31	23	15	07

Po wykonaniu permutacji początkowej blok wejściowy jest dzielony na dwa równe, 32-bitowe bloki: lewy L_0 i prawy R_0 . Dalsze operacje są wykonywane oddzielnie na każdym z tych bloków. Blok R_0 przesuwany jest na lewo a blok L_0 przekształcany według funkcji szyfrującej f i umieszczany z prawej strony. Spełnione są więc zależności:

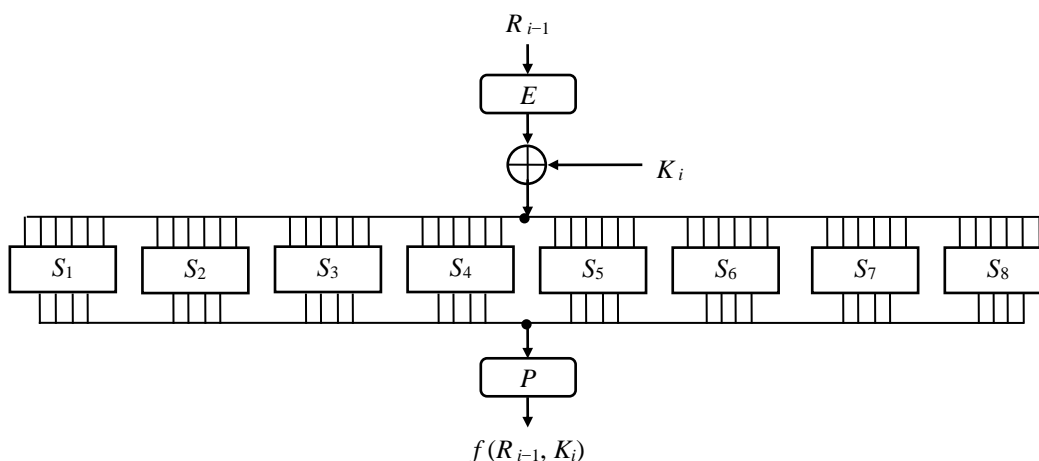
$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i).$$

Operacje te powtarzane są szesnaście razy. Po ostatniej iteracji nie zmienia się pozycji części lewej i prawej. Następnie łączy się część lewą i prawą i wykonuje się permutację końcową IP^{-1} zgodnie z poniższą tabelą:

IP^{-1}															
40	08	48	16	56	24	64	32	39	07	47	15	55	23	63	31
38	06	46	14	54	22	62	30	37	05	45	13	53	21	61	29
36	04	44	12	52	20	60	28	35	03	43	11	51	19	59	27
34	02	42	10	50	18	58	26	33	01	41	09	49	17	57	25

Funkcja szyfrująca $f(R_{i-1}, K_i)$



Rys. 2. Schemat blokowy obliczania funkcji szyfrującej $f(R_{i-1}, K_i)$

Funkcja szyfrująca zawiera szereg operacji pokazanych na Rys. 2. Danymi wejściowymi funkcji f są: 32-bitowy blok R_{i-1} i 48-bitowy klucz K_i . Na bloku 32-bitowym R_{i-1} wykonuje się permutację rozszerzającą E zgodnie z poniższą tabelą:

E															
32	01	02	03	04	05	04	05	06	07	08	09	08	09	10	11
12	13	12	13	14	15	16	17	16	17	18	19	20	21	20	21
22	23	24	25	24	25	26	27	28	29	28	29	30	31	32	01

W wyniku tej permutacji tworzy się blok 48-bitowy. Następnie na elementach tego bloku i klucza K_i wykonuje się operację XOR a blok wyjściowy dzieli się na osiem ciągów 6-bitowych. Każdy z tych ciągów jest poddany operacji podstawienia według tablic $S_1 \div S_8$ w następujący sposób. Pierwszy i ostatni bit ciągu 6-bitowego określa numer wiersza tablicy S_i od 0 do 3, a cztery bity środkowe numer jej kolumny od 0 do 15. Tablice podstawień dla kolejnych iteracji są następujące:

S_1															
14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

S_2															
15	01	08	14	06	11	03	04	09	07	02	13	12	00	05	10
03	13	04	07	15	02	08	14	12	00	01	10	06	09	11	05
00	14	07	11	10	04	13	01	05	08	12	06	09	03	02	15
13	08	10	01	03	15	04	02	11	06	07	12	00	05	14	09

S_3															
10	00	09	14	06	03	15	05	01	13	12	07	11	04	02	08
13	07	00	09	03	04	06	10	02	08	05	14	12	11	15	01
13	06	04	09	08	15	03	00	11	01	02	12	05	10	14	07
01	10	13	00	06	09	08	07	04	15	14	03	11	05	02	12

S_4

07 13 14 03 00 06 09 10 01 02 08 05 11 12 04 15
13 08 11 05 06 15 00 03 04 07 02 12 01 10 14 09
10 06 09 00 12 11 07 13 15 01 03 14 05 02 08 04
03 15 00 06 10 01 13 08 09 04 05 11 12 07 02 14

S_5

02 12 04 01 07 10 11 06 08 05 03 15 13 00 14 09
14 11 02 12 04 07 13 01 05 00 15 10 03 09 08 06
04 02 01 11 10 13 07 08 15 09 12 05 06 03 00 14
11 08 12 07 01 14 02 13 06 15 00 09 10 04 05 03

S_6

12 01 10 15 09 02 06 08 00 13 03 04 14 07 05 11
10 15 04 02 07 12 09 05 06 01 13 14 00 11 03 08
09 14 15 05 02 08 12 03 07 00 04 10 01 13 11 06
04 03 02 12 09 05 15 10 11 14 01 07 06 00 08 13

S_7

04 11 02 14 15 00 08 13 03 12 09 07 05 10 06 01
13 00 11 07 04 09 01 10 14 03 05 12 02 15 08 06
01 04 11 13 12 03 07 14 10 15 06 08 00 05 09 02
06 11 13 08 01 04 10 07 09 05 00 15 14 02 03 12

S_8

13 02 08 04 06 15 11 01 10 09 03 14 05 00 12 07
01 15 13 08 10 03 07 04 12 05 06 11 00 14 09 02
07 11 04 01 09 12 14 02 00 06 10 13 15 03 05 08
02 01 14 07 04 10 08 13 15 12 09 00 03 05 06 11

Wynikiem podstawień są liczby od 0 do 15. Zamienia się je na liczby binarne i łączy razem w jeden ciąg 32-bitowy. Na ciągu tym wykonuje się permutację P zgodnie z poniższą tabelą:

P

16 07 20 21 29 12 28 17 01 15 23 26 05 18 31 10
02 08 24 14 32 27 03 09 19 13 30 06 22 11 04 25

Uzyskany w ten sposób ciąg binarny jest dodawany modulo 2 do L_{i-1} (XOR), w rezultacie czego powstaje blok R_i , jak to pokazano na Rys. 1.

Generowanie kluczy

Klucz pierwotny 64-bitowy wprowadzony do systemu kryptograficznego służy do generowania 16 kluczy wtórnych używanych w procesie szyfrowania i deszyfrowania. W procesie deszyfrowania używa się kluczy w odwrotnej kolejności. Schemat blokowy algorytmu generowania kluczy pokazano na Rys. 3.

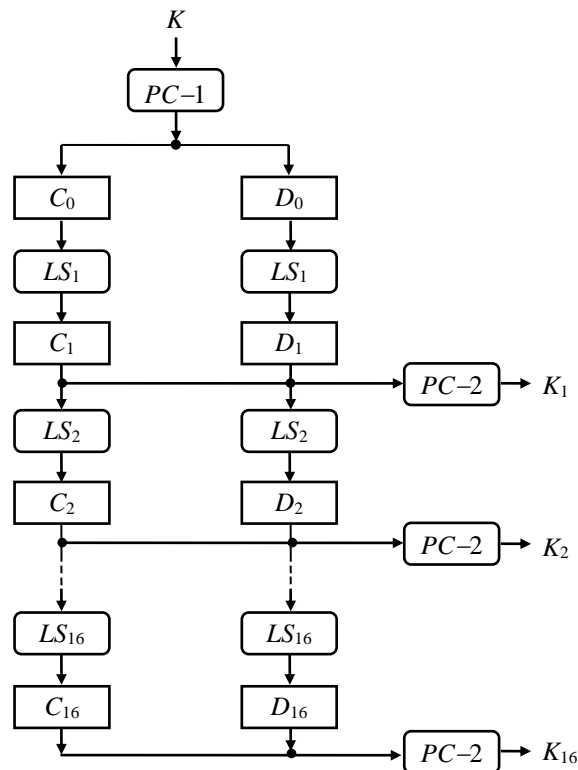
Klucz pierwotny po odrzuceniu bitów parzystości (8,16,24,32,40,48,56,64) jest poddany permutacji PC-1 zgodnie z tabelą:

PC-1															
57	49	41	33	25	17	09	01	58	50	42	34	26	18		
10	02	59	51	43	35	27	19	11	03	60	52	44	36		
63	55	47	39	31	23	15	07	62	54	46	38	30	22		
14	06	61	53	45	37	29	21	13	05	28	20	12	04		

Następnie blok klucza 56-bitowy dzieli się na dwa bloki 28-bitowe C_0 i D_0 . Bloki te są przesuwane cyklicznie w lewo. Ilość przesunięć w lewo dla kolejnych szesnastu iteracji są następujące: 1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1. Po przesunięciu bloków C_i i D_i łączy się je razem i poddaje permutacji z kompresją PC-2 zgodnie z poniższą tabelą:

PC-2															
14	17	11	24	01	05	03	28	15	06	21	10				
23	19	12	04	26	08	16	07	27	20	13	02				
41	52	31	37	47	55	30	40	51	45	33	48				
44	49	39	56	34	53	46	42	50	36	29	32				

Otrzymany w ten sposób 48-bitowy klucz jest kluczem używanym w procesie szyfrowania lub deszyfrowania.



Rys. 3. Schemat blokowy generatora kluczy

Szyfry blokowe

Szyfr blokowy dzieli wiadomość M na bloki M_1, M_2, \dots i każdy z nich szyfruje, używając tego samego klucza K .

$$E_K(M) = E_K(M_1)E_K(M_2)\dots$$

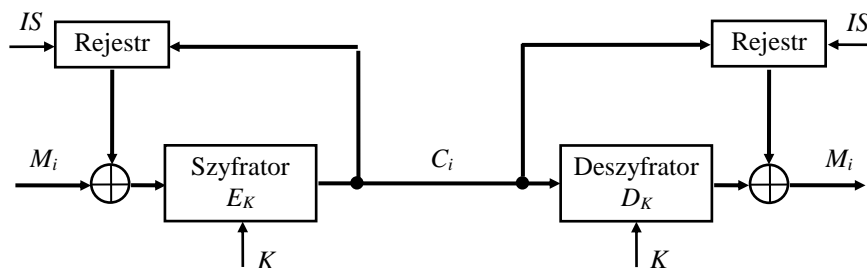
Do algorytmów blokowych należą: szyfry przestawieniowe, Lucifer, DES i szyfry z kluczem jawnym. W szyfrach blokowych następuje propagacja błędów transmisyjnych o zakresie jednego bloku, ale błędy te nie mają wpływu na inne bloki.

Najprostszą metodą szyfrowania blokowego jest przekształcenie każdego bloku tekstu jawnego w blok szyfrogramu niezależnie od innych bloków. Taka metoda pracy nazywa się *trybem elektronicznej książki kodowej* (electronic codebook – ECB).

Tryb ECB jest najłatwiejszy do implementacji. Brak związków między blokami ułatwia szyfrowanie w bazach danych i transmisji pracującej w trybie datagramowym. W trybie datagramowym pakiety wiadomości są przekazywane niezależnie, a stacja odbiorcza układa je w odpowiedniej kolejności.

Słabą stroną ECB jest możliwość zgromadzenia przez kryptoanalityka takiej ilości materiałów, która umożliwia złamanie szyfru, gdy nie znamy klucza. W tym celu kryptoanalitycy wykorzystują powtarzające się fragmenty dokumentów, np. w poczcie elektronicznej. Szyfry blokowe można uodpornić na kryptoanalizę, stosując technikę wiązania blokowego.

W *trybie wiązanie bloków* wykorzystano technikę sprzężenia zwrotnego, w której blok poprzedni jest służący do modyfikacji szyfrowania bloku następnego. W ten sposób każdy blok szyfrogramu zależy nie tylko od bieżącego bloku tekstu jawnego, ale również od wszystkich poprzednich bloków. Istnieją różne tryby wiązania bloków, lecz najczęściej korzysta się z wiązania bloków zaszyfrowanych (cipher block chaining – CBC) i jego modyfikacji.



Rys. 4. Szyfr blokowy z wiązaniem bloków zaszyfrowanych

Schemat szyfrowania z wiązaniem bloków zaszyfrowanych pokazano na Rys. 4. Grube linie oznaczają połączenia dla bloków wiadomości. W układzie sprzężenia zwrotnego znajdują się rejestry przesuwne umożliwiające zapamiętanie bloku kryptogramu. Kolejny blok wiadomości jest sumowany

modulo dwa (XOR) z poprzednim blokiem kryptogramu, a następnie szyfrowany za pomocą klucza K w szyfratorze. Szyfrowanie wyraża zależność:

$$C_i = E_K(M_i + C_{i-1}).$$

Deszyfrowanie wykonuje się obliczając:

$$D_K(C_i) + C_{i-1} = D_K(E_K(M_i + C_{i-1})) + C_{i-1} = (M_i + C_{i-1}) + C_{i-1} = M_i.$$

W celu rozpoczęcia pracy systemu stosuje się wektor początkowy C_0 , który jest zwykle ciągiem losowym.

Ponieważ każdy zaszyfrowany blok ma powiązanie z blokiem poprzednim, przeto jeden błąd transmisyjny wpływa najwyżej na dwa bloki wyjściowe. System ten jest wrażliwy na błędy synchronizacji (IS – impulsy synchronizujące), gdyż przesunięcie szyfrogramu nawet o jeden bit powoduje błędną pracę systemu. Jednocześnie każdy zaszyfrowany blok zależy od wszystkich poprzednich bloków zaszyfrowanych, więc cechy statystyczne tekstu jawnego rozprzestrzeniają się na cały zaszyfrowany tekst, co znacznie utrudnia kryptoanalizę.

Tryb wiązania bloków zaszyfrowanych opracowano w IBM i zastosowano w Systemie Ochrony Informacji (Information Protection System – IPS). System ten pracuje z algorytmem DES. Tryb wiązania bloków zaszyfrowanych jest najlepszy do szyfrowania plików. Do szyfrowania baz danych wygodniejszy jest tryb bezpośredniego szyfrowania bloków. Dla baz danych opracowano też inne rodzaje szyfrów blokowych jak np. szyfr blokowy z podkluczami.

W szyfrach blokowych stosuje się również techniki szyfrowania wielokrotnego. Szyfrowanie wielokrotne ma miejsce wtedy, kiedy ten sam blok tekstu jawnego jest szyfrowany wiele razy. W literaturze opisano metody szyfrowania dwukrotnego lub trzykrotnego z różnymi kluczami, np. 3DES.

Opis oprogramowania

Głównym programem ćwiczenia jest CIPH.CPP, który szyfruje jeden blok tekstu wprowadzonego z klawiatury w postaci szesnastu znaków kodu szesnastkowego. System szesnastkowy ułatwia posługiwanie się programem.

Program CIPH używa następujących funkcji:

- HEXTOBIN - funkcja zamieniająca cyfry szesnastkowe na dwójkowe;
- BINTOHEX – funkcja zamieniająca ciągi czterobitowe na cyfry szesnastkowe;
- *des* - funkcja realizująca algorytm szyfrowania DES - blok 64-bitowy i klucz 64-bitowy, podawane w postaci znaków hex.

Funkcja *des* korzysta z pliku danych DESINP.DAT oraz z następujących funkcji:

- *ks* - funkcja służąca do generowania szesnastu wtórnych kluczy kryptograficznych; funkcja ta korzysta z pliku danych KSINPU.DAT;
- *cyfun* – funkcja realizuje funkcję szyfrującą; *cyfun* korzysta z pliku danych CYFUNL.DAT.

Do skojarzenia zmiennej plikowej z plikiem danych i otwarcia pliku do odczytu używa się w programach funkcji *glopen*. Plik HEX.DAT zawiera definicje binarne liczb heksadecymalnych 0...F wykorzystywane podczas konwersji HEXTOBIN.

Do deszyfrowania służy program DECIPH.CPP, który używa analogicznych funkcji. Jedynie do deszyfrowania wykorzystuje funkcję:

- *d_des* – funkcja realizująca algorytm deszyfrowania DES – blok 64-bitowy i klucz 64-bitowy, podawane w postaci znaków hex (realizuje szyfrowanie kluczami branymi w odwrotnej kolejności).

Przebieg ćwiczenia

Przed ćwiczeniem należy zapoznać się z opisem algorytmu i oprogramowaniem ćwiczenia.

1. Uruchomić i przeanalizować pracę programu CIPH. Przeanalizować również procedury *des*, *ks*, *cyfun*. Zszyfrować kilka 8-bajtowych bloków danych podawanych w postaci 16 znaków hex (blok 64-bitowy) stosując różne klucze 64-bitowe podane w postaci hex. Odszyfrować kryptogramy za pomocą programu DECIPH. Alternatywnie można opracować własny program w wybranym języku programowania, który realizuje procedury szyfrowania i deszyfrowania 8-bajtowych bloków danych podawanych w postaci 16 znaków hex (blok 64-bitowy) z wykorzystaniem wybranego algorytmu kryptografii symetrycznej (np. DES, 3DES, AES).
2. Przekształcić program CIPH.CPP (lub własny program) i odpowiednie procedury tak aby było możliwe wykonanie operacji szyfrowania bloków zawierających 8 znaków ASCII z wykorzystaniem klucza podawanego w postaci 8 znaków ASCII. Wprowadzić opcję deszyfrowania bloku złożonego z 8 bajtów danych podawanych w postaci 16 znaków hex z wykorzystaniem klucza podawanego w postaci 8 znaków ASCII. Opcje szyfrowania i deszyfrowania powinny być niezależne, tj. powinna istnieć możliwość wprowadzenia tekstu jawnego ASCII oraz kryptogramu hex z klawiatury, lub w oparciu o kopiowanie, a następnie wykonanie operacji szyfrowania lub deszyfrowania z udziałem podanego klucza ASCII (zobacz działanie programu N_CIPH oraz N_DECIPH).
3. Dostosować opracowany program do szyfrowania i deszyfrowania dowolnego pliku w blokach po 8 bajtów w trybie DES ECB 64 bity (lub inny algorytm, gdy opracowano własny program). W przypadku ostatniego, niepełnego bloku pliku, w którym może występować od 0 do 7 bajtów uzupełnić blok do 8 bajtów losowymi liczbami z przedziału od 0 do 255. Na końcu

zaszyfrowanego pliku dodać zaszyfrowany blok, w którym na pierwszej pozycji znajduje się liczba równa długości ostatniego bloku pliku, a pozostałe bajty są losowymi liczbami z przedziału od 0 do 255. Zweryfikować poprawność działania operacji szyfrowania i deszyfrowania plików (np. wykorzystać pliki tekstowe, proste mapy bitowe; sprawdzić, czy plik zdeszyfrowany jest identyczny jak plik wejściowy – zawartość oraz długość plików w bajtach powinny być takie same). Alternatywnie można zaproponować własną metodę rozwiązania problemu niepełnego, ostatniego bloku pliku.

4. Zrealizować analogiczny program do szyfrowania i deszyfrowania dowolnych plików w trybie DES CBC 64 bity (lub inny algorytm i tryb szyfrowania, jeśli opracowano własny program, np. AES CBC). Zweryfikować poprawność działania operacji szyfrowania i deszyfrowania plików (np. z udziałem przykładowych plików tekstowych, prostych map bitowych; plik zdeszyfrowany i wejściowy powinny być identyczne).

Przykłady

	Kod szesnastkowy	Kod szesnastkowy	Kod ASCII
Tekst:	0000000000000000	abc0000000000000	s s s s s s s s
Klucz:	0000000000000000	1230000000000000	4 4 4 4 4 4 4 4
Szyfr:	8ca64de9c1b123a7	d2b42378f52ec5ac	2a5ea167aff22a36