Zadanie 2

Metody szyfrowania danych

Celem ćwiczenia jest zapoznanie się z podstawowymi metodami szyfrowania danych z użyciem kluczy symetrycznych i asymetrycznych. W skład ćwiczenia wchodzą następujące elementy:

1. Implementacja jednej z metod blokowych z kluczem symetrycznym zgodnie z przydzielonym wariantem:
   - (**S1**) Data Encryption Standard (DES),
   - (**S2**) Fast Data Encryption Algorithm (FEAL),
   - (**S3**) International Data Encryption Algorthm (IDEA),
   - (**S4**) Advanced Encryption Standard (AES),
2. Implementacja jednej z metod z kluczem asymetrycznym (tzw. kluczem publicznym) zgodnie z przydzielonym wariantem zadania:
   - (**A1**) szyfrowanie RSA,
   - (**A2**) metoda Rabin,
   - (**A3**) metoda ElGamal,
   - (**A4**) metoda plecakowa z kluczem publicznym.

Metody szyfrowania danych z podpunktu 1 powinny być zaimplementowane w postaci aplikacji umożliwiających szyfrowanie (i późniejsze deszyfrowanie) dowolnych plików. W danym przypadku dopuszczalne są następujące rozwiązania: dwa pliki, po jednym osobno dla programu szyfrującego i deszyfrującego, oraz jeden plik z opcją wyboru szyfrowanie/deszyfrowanie z poziomu menu lub poprzez parametr podawany z wiersza poleceń. Aplikacje powinny być uruchamiane z wiersza polecenia i przyjmować nazwy plików do odczytu/zapisu w postaci argumentów. Podstawowym kryterium poprawnej implementacji metody jest cykl:

- szyfrowanie pliku1 i zapis wyniku do pliku2,
- deszyfrowanie pliku2 i zapis wyniku do pliku3,
- porównanie pliku1 i pliku3 (muszą być identyczne).

Pliki wynikowe po operacji szyfrowania powinny zawierać wszystkie dodatkowe informacje, które są wymagane do poprawnego odszyfrowania danych wejściowych. Oczywiście uwaga ta nie dotyczy informacji poufnych (np. w postaci kluczy szyfrujących), które powinien znać wyłącznie adresat szyfrowanej wiadomości.

Przydzielony wariant metody szyfrowania z kluczem asymetrycznym należy krótko opracować teoretycznie.

Sprawozdanie z realizacji zadania powinno zawierać następujące elementy:

- instrukcję korzystania z aplikacji,

- opis implementacji ze szczególnym uwzględnieniem zastosowanych struktur danych,
- opis rozwiązania przypadków szczególnych (również: inicjalizacji, warunków stopu),
- prezentację własnych pomysłów i usprawnień.

Opis metod szyfrowania

Metody opisane w niniejszej instrukcji stanowią jedne z podstawowych metod kryptograficznych, które w przeciągu ostatnich kilku dziesięcioleci znalazły szerokie zastosowania praktyczne w mechanizmach ochrony i zapewnienia integralności danych, protokołach autoryzacyjnych, oraz w systemach elektronicznych podpisów cyfrowych. W pierwszej części instrukcji skupiono się na metodach blokowych symetrycznych (z tzw. kluczem prywatnym). Natomiast drugą część poświęcono metodom blokowym asymetrycznym o jawnym kluczu szyfrującym (tzw. kluczu publicznym).

1. Metody symetryczne (metody z kluczem prywatnym)

W metodach symetrycznych wykorzystuje się uzgodniony przez obie strony klucz (tzw. klucz prywatny), który wymagany jest zarówno w procesach szyfrowania jak i deszyfrowania danych, przy czym najczęściej obie strony występują w charakterze nadawcy i odbiorcy. W danym schemacie bezpieczeństwo przesyłanych wiadomości zależy w zasadniczym stopniu od tajności klucza prywatnego. Metody te posiadają następujące niedogodności:

1. Brak możliwości ustalenia po stronie odbiorcy od kogo pochodzi nadesłana wiadomość, co może stanowić poważną przeszkodę np. w systemach transakcji bankowych,
2. Konieczność zagwarantowania bezpiecznego kanału dla uzgadniania i wymiany kluczy pomiędzy stronami wymieniającymi wiadomości,
3. Wymagana duża liczba kluczy prywatnych i kanałów w przypadku dużej liczby stron zainteresowanych komunikacją.

Metody blokowe symetryczne operują na n-bitowych paczkach (blokach) danych tekstu jawnego, który przy pomocy sparametryzowanej kluczem prywatnym funkcji szyfrującej odwzorowywany jest na n-bitowy blok tekstu zaszyfrowanego (szyfrogram). W chwili, gdy długość tekstu jawnego jest większa od n, stosuje się kilka różnych trybów działania.

Najprostszy z nich ECB (z ang. $Electronic\ Code\ Book$, czyli elektroniczna książka kodowa - ze względu na niski poziom komplikacji tryb ten jest zalecany do realizacji w ramach niniejszego ćwiczenia) polega na niezależnym przetwarzaniu wszystkich n-bitowych bloków danych. Ze względu na niezależny sposób generowania przyległych bloków tekstu zaszyfrowanego podejście to nie ukrywa pewnych schematów, które mogą występować w szyfrowanych zbiorach danych - identyczne bloki tekstu jawnego implikują identyczne bloki tekstu zaszyfrowanego. Stąd tryb ECB nie jest zalecany dla tekstów dłuższych niż jeden blok.

Tryb CBC (z ang. $Cipher\ Block\ Chaining$, tryb wiązania bloków szyfrogramu) wymaga dodatkowego n-bitowego wektora, którym inicjalizuje się początkową ("zerową") wartość szyfrogramu. Postaci szyfrogramów dla poszczególnych bloków wiadomości otrzymuje się w wyniku przekształcania funkcją szyfrującą kolejnych bloków tekstu jawnego, które dodatkowo sumuje

się modulo 2 z blokiem szyfrogramu z kroku poprzedzającego. W ten sposób uzyskuje się szyfrogram zależny od całej historii sesji szyfrowania.

Pozostałe tryby oferują możliwość generowania danych wyjściowych o różnych długościach r mniejszych od n, często $r = 1, 8, 16$, czy też 32 bity. Są to tryby CFB (z ang. $Cipher\ Feedback$, tryb sprzężenia zwrotnego szyfrogramu), oraz OFB (z ang. $Output\ Feedback$, tryb wyjściowego sprzężenia zwrotnego), który dodatkowo cechuje brak propagacji ewentualnych błędów w jednym bloku szyfrogramu na bloki kolejne.

Data Encryption Standard (DES)

Algorytm DES (z ang. $Data\ Encryption\ Standard$) został opracowany w roku 1977 w Stanach Zjednoczonych na polecenie Narodowego Biura Standaryzacji (NSA) dla potrzeb szyfrowania ogólnych danych komputerowych (standard FIPS PUB 46). Dzięki stosunkowo niewielkim wymaganiom pamięciowym i obliczeniowym metoda ta zyskała ogromną popularność i znalazła szerokie zastosowania m.in. w bankowości dla potrzeb elektronicznego transferu walut, do szyfrowania cywilnej komunikacji satelitarnej, czy też do ochrony haseł w systemie Unix.

Obecnie ze względu na niewielką długość klucza szyfrującego (64-bity, przy czym efektywna długość wynosi 56 bitów, gdyż osiem bitów to bity parzystości) algorytm DES jest zastępowany nowszymi wariantami: TripleDES, DES-X, lub innymi metodami np. AES, RC6, Serpent, MARS, czy też Twofish. Jednakże prywatni użytkownicy w najbliższym czasie najprawdopodobniej nie będą w stanie złamać tej metody. Dla przykładu na złamanie algorytmu DES w roku 2007 maszyna COPACOBANA o równoległej architekturze FPGA (cena około 10.000 dolarów) potrzebowała 6.4 dnia.

Szyfrowanie

W metodzie DES szyfrowaniu podlega $n = 64$-bitowy blok danych (8 bajtów). Na początku blok tekstu jawnego poddawany jest transpozycji IP (patrz Tabela S1.1), a następnie dzielony na dwa 32-bitowe bloki L_0 i R_0. Dalej wykonuje się 16 kroków (dla $i = 1, 2, ..., 16$) przeprowadzanych według poniższego schematu

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

gdzie K_i stanowi 48-bitowy i-ty klucz pośredni, który generowany jest z 64-bitowego klucza szyfrującego K zgodnie z Algorytmem S1.2. Funkcja f stanowi zasadniczą część algorytmu. Tutaj 32-bitowy blok R_i jest rozszerzany do bloku 48-bitowego (patrz Tabela S1.3) poprzez powielenie bitów. Następnie blok ten sumowany jest bitowo modulo 2 (operacja $\oplus$) z kluczem pośrednim K_i. Wynikowy blok 48-bitowy dzielony jest na osiem grup po 6 bitów każda. Grupy te stanowią dane wejściowe dla modułów podstawienia $S_1 ... S_8$ (tzw. S-boksów), które przeprowadzają podstawienia nieliniowe. Osiem 4-bitowych bloków wyjściowych z modułów S (po zakończeniu wszystkich kroków $i = 1, 2, ..., 16$) po konkatenacji poddaje się transpozycji końcowej IP^{-1} (patrz Tabela S1.2). Cały proces szyfrowania opisany został za pomocą Algorytmu S1.1.

Deszyfrowanie

Do deszyfrowania wykorzystuje się ten sam algorytm (patrz Algorytm S1.1), przy czym klucze pośrednie K_i dobiera się w odwrotnej kolejności.

Algorytm S1.1 Szyfrowanie/deszyfrowanie DES

WEJŚCIE: 64-bitowy blok tekstu jawnego: $M = m_1m_2...m_{64}$; 64-bitowy klucz $K = k_1k_2...k_{64}$.
WYJŚCIE: 64-bitowy blok tekstu zaszyfrowanego $C = c_1c_2...c_{64}$.

1. Wyznacz szesnaście 48-bitowych kluczy pośrednich K_i za pomocą Algorytmu S1.2.
2. $L_0R_0 \leftarrow IP(m_1m_2...m_{64})$. Operacja IP oznacza wstępną permutację bitów tekstu jawnego według Tabeli S1.1. Przemieszane bity tekstu jawnego należy rozdzielić na dwie 32-bitowe połowy: $L_0 = m_{58}m_{50}...m_8$, $R_0 = m_{57}m_{49}...m_7$.
3. Wykonaj szesnaście iteracji dla $i = 1, ..., 16$, obliczając L_i i R_i zgodnie ze wzorem (1), przy czym $f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$ wyznaczamy według schematu:
   (a) Rozszerz $R_{i-1} = r_1r_2...r_{32}$ z 32 na 48 bity przy użyciu operatora E (patrz Tabela S1.3), tj. $T \leftarrow E(R_{i-1})$. Stąd $T = r_{32}r_1r_2...r_{32}r_1$.
   (b) $T' \leftarrow T \oplus K_i$ (operacja $\oplus$ oznacza alternatywę wykluczającą na bitach). Przedstaw T' w postaci ośmiu 6-bitowych bloków: $B_1B_2...B_8 = T'$.
   (c) $T'' \leftarrow S_1(B_1)S_2(B_2)...S_8(B_8)$. Przez S_i oznaczamy podstawienie (z ang. S-box) w miejsce $B_i = b_1b_2...b_6$ pewnej 4-bitowej wartości z Tabeli S1.7. Dla każdego i nowa wartość identyfikowana jest poprzez numer wiersza: $n_w = 2 \cdot b_1 + b_6$ oraz numer kolumny liczony jako 4-bitowa liczba $b_2b_3b_4b_5$ w kodzie naturalnym. Przykład: $S_2(110010)$ wyznacza indeksy: $n_w = 2$, $n_k = 9$, co po odczytaniu z tablicy daje podstawianą wartość 1000.
   (d) $T''' \leftarrow P(T'')$. Wykonaj permutację 32-bitowego bloku T'' zgodnie z Tabelą S1.4. W rezultacie otrzymujemy: $T''' = t_{16}t_7...t_{25}$. Stąd $f(R_{i-1}, K_i) \leftarrow T'''$.
4. $b_1b_2....b_{64} \leftarrow R_{16}L_{16}$. Zamiana miejscami finalnych bloków R_{16} i L_{16}.
5. $C \leftarrow IP^{-1}(b_1b_2...b_{64})$. Odwrotna permutacja IP zgodnie z Tabelą S1.2. Stąd ostateczna postać tekstu zaszyfrowanego $C = b_{40}b_8...b_{25}$.

Wyznaczanie kluczy pośrednich

W metodzie DES do szyfrowania/deszyfrowania wykorzystuje się tzw. klucze pośrednie, które generowane są na podstawie klucza szyfrującego K. W tym celu najpierw usuwane są bity parzystości (patrz operacja $PC1$), a następnie 56-bitowy blok dzieli się na dwa moduły 28-bitowe. W kolejnych krokach bloki te przesuwa się cyklicznie w lewo o 1 lub 2 bity w zależności od indeksu przebiegu. Klucz pośredni o długości 48 bitów powstaje po przemieszaniu bloków 28-bitowych zgodnie z Tabelą S1.6. Sposób wyznaczania kluczy opisuje Algorytm S1.2.

Algorytm S1.2 Wyznaczanie kluczy pośrednich K_i (DES)

WEJŚCIE: 64-bitowy klucz $K = k_2k_2...k_{64}$.
WYJŚCIE: szesnaście 48-bitowych kluczy K_i dla $i = 1, 2, ..., 16$.

1. Przez v_i dla $i = 1, 2, ..., 16$ oznaczamy wartości przesunięć bitowych, przy czym $v_i = 1$ dla $i \in \{1, 2, 9, 16\}$ oraz $v_i = 2$ w przeciwnym wypadku.

2. $C_0D_0 \leftarrow PC1(k)$. Przedstaw klucz K w postaci dwóch 28-bitowych bloków C_0 i D_0, dla których bity klucza szyfrującego przydzielane są za pomocą operatora $PC1$ definiowanego zgodnie z Tabelą S1.5. Stąd $C_0 = k_{57}k_{49}...k_{36}$ oraz $D_0 = k_{63}k_{55}...k_4$.

3. Dla $i = 1, 2, ..., 16$ wyznacz postaci kolejnych kluczy pośrednich K_i według następującego schematu: $C_i \leftarrow (C_{i-1} \hookleftarrow v_i)$, $D_i \leftarrow (D_{i-1} \hookleftarrow v_i)$ oraz $K_i \leftarrow PC2(C_i, D_i)$. Operacja $\hookleftarrow$ oznacza cykliczną rotację bitową. Postać klucza K_i wyznacza się jako konkatenację przemieszanych zgodnie z Tablicą S1.6 bitów $b_1b_2...b_{56}$ bloków C_i i D_i, tzn. jeżeli $C_i = b_1b_2...b_{28}$ oraz $D_i = b_{29}b_{30}...b_{56}$, to $K_i = b_{14}b_{17}...b_{32}$.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Tabela S1.1 Tablica permutacji IP

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Tabela S1.2 Tablica permutacji IP^{-1}

32	1	2	3	4	5
4	5	6	7	8	9
8	9	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	1

Tabela S1.3 Operator rozszerzenia E

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Tabela S1.4 Permutacja P

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Tabela S1.5 Operator wyboru bitów $PC1$

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Tabela S1.6 Operator wyboru $PC2$

wiersz	kolumna															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1																
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2																
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3																
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4																
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5																
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6																
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7																
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8																
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Tabela S1.7 Podstawienia nieliniowe dla S-boksów S_i

Bezpieczeństwo metody

Długość klucza szyfrującego K w metodzie DES wynosi 64 bity, przy czym 8 z nich to bity parzystości. Zatem efektywna długość klucza wynosi 56 bitów, co daje zbiór Z wszystkich

możliwych kluczy o mocy $|Z| = 2^{56} \approx 72 \cdot 10^{15}$. Taka ilość testów wymagana jest podczas ataku wyczerpującego*. Istnieją jednak metody jej redukcji nawet do poziomu 2^{43} (patrz [1]).

Fast Data Encipherment (FEAL)

Metoda FEAL została po raz pierwszy opublikowana w roku 1987 w Japonii. Głównym założeniem projektantów metody było opracowanie algorytmu cechującego się dużą szybkością działania w implementacjach programowych, w szczególności z przeznaczeniem dla procesorów 8-bitowych i kart chipowych. Stąd algorytm ten jest zorientowany na operacje bajtowe: 8-bitowe dodawania modulo 256, oraz rotacje i operacje XOR na blokach 8-bitowych.

Z założenia algorytm FEAL miał stanowić szybszą alternatywę dla algorytmu DES, lecz okazał się od niego znacząco mniej bezpieczny. Jego bezpieczeństwo można wzmocnić stosując wiekszą ilość rund, np. wariant FEAL-32 (patrz [2]).

Opisany w niniejszej instrukcji wariant to FEAL-8 (o ośmiu rundach), który operuje na 64-bitowych blokach tekstu jawnego ($n = 64$) i wymaga klucza o długości 64 bitów.

Szyfrowanie
W procesie szyfrowania wymaganych jest szesnaście 16-bitowych kluczy pośrednich K_i dla parametru $i = 0, 1, ..., 15$, które generowane są na podstawie klucza szyfrującego K za pomocą Algorytmu S2.2. Na wstępie 64-bitowy blok tekstu jawnego dzielony jest na dwa moduły L_0 i R_0 po 32 bity każdy. Dla modułów tych wykonuje się bitowo operację alternatywy wykluczającej (XOR) z kluczami pośrednimi $K_8K_9K_{10}K_{11}$, których konkatenacja tworzy blok 64-bitowy. Dalej przeprowadza się osiem iteracji (tzw. rund) według poniższego schematu:

$$L_i \leftarrow R_{i-1}, \quad R_i \leftarrow L_{i-1} \oplus f(R_{i-1}, K_{i-1}),$$

gdzie funkcja f przeprowadza bajtowo zorientowane operacje podstawień S_0 i S_1 (S-boksy). Postać funkcji f definiuje Tablica S2.1, w której

$$S_d(x, y) = \mathrm{ROT2}((x + y + d) \bmod 256),$$

gdzie ROT2 oznacza operację rotacji w lewo o dwa bity. W ostatniej fazie realizuje się operację XOR dla bloków R_8 i L_8 z kluczami pośrednimi $K_{12}K_{13}K_{14}K_{15}$. Należy zwrócić uwagę na to, że bloki L_8 i R_8 zostały zamienione miejscami.

Deszyfrowanie
Podczas deszyfrowania wykorzystuje się ten sam Algorytm S2.1, przy czym odwraca się kolejność kluczy pośrednich w kroku 5 algorytmu, tzn. klucze podaje się w kolejności od K_7 do K_0, oraz w krokach 3 i 7 używa się następujących konkatenacji kluczy pośrednich: $K_{12}K_{13}K_{14}K_{15}$ (krok 3) oraz $K_8K_9K_{10}K_{11}$ (krok 7).

Wyznaczanie kluczy pośrednich
Do wyznaczania kluczy pośrednich K_i dla $i = 0, 1, ..., 15$ służy Algorytm S2.2. W algorytmie tym wykorzystuje się podobną do f funkcje f_K, której postać opisuje Tabela S2.1.

*Atak wyczerpujący (z ang. *brute force attack*) polega na testowaniu kolejno wszystkich możliwych kombinacji kluczy w celu odczytania zaszyfrowanej wiadomości.

	$U \leftarrow f(A, Y)$	$U \leftarrow f_K(A, B)$
$t_1 =$	$(A_0 \oplus A_1) \oplus Y_0$	$A_0 \oplus A_1$
$t_2 =$	$(A_2 \oplus A_3) \oplus Y_1$	$A_2 \oplus A_3$
$U_1 =$	$S_1(t_1, t_2)$	$S_1(t_1, t_2 \oplus B_0)$
$U_2 =$	$S_0(t_2, U_1)$	$S_0(t_2, U_1 \oplus B_1)$
$U_0 =$	$S_0(A_0, U_1)$	$S_0(A_0, U_1 \oplus B_2)$
$U_3 =$	$S_1(A_3, U_2)$	$S_1(A_3, U_2 \oplus B_3)$

Tabela S2.1 Definicje funkcji f i f_K o wyjściach w postaci konkatenacji $U = U_0U_1U_2U_3$.

Algorytm S2.1 Algorytm szyfrowania FEAL-8

WEJŚCIE: 64-bitowy blok tekstu jawnego: $M = m_1m_2...m_{64}$; 64-bitowy klucz $K = k_1k_2...k_{64}$.
WYJŚCIE: 64-bitowy blok tekstu zaszyfrowanego $C = c_1c_2...c_{64}$.

1. Oblicz szesnaście kluczy pośrednich K_i dla $i = 0, 1, ..., 15$ za pomocą Algorytmu S2.2.
2. Niech $M_L = m_1...m_{32}$ i $M_R = m_{33}...m_{64}$.
3. Oblicz $(L_0R_0) \leftarrow (M_LM_R) \oplus (K_8K_9K_{10}K_{11})$. Operacja XOR wykonywana jest dla bloku tekstu jawnego i konkatenacji kluczy pośrednich K_8, K_9, K_{10} i K_{11}.
4. Wykonaj podstawienie $R_0 \leftarrow R_0 \oplus L_0$.
5. Wykonaj osiem rund dla $i = 1, ..., 8$ postaci: $L_i \leftarrow R_{i-1}$, oraz $R_i \leftarrow L_{i-1} \oplus f(R_{i-1}, K_{i-1})$, gdzie $A = R_{i-1} = A_0A_1A_2A_3$ i $Y = K_{i-1} = Y_0Y_1$. Bloki A_i oraz Y_i są 8-bitowe.
6. Wykonaj podstawienie $L_8 \leftarrow L_8 \oplus R_8$.
7. Oblicz $(R_8L_8) \leftarrow (R_8L_8) \oplus (K_{12}K_{13}K_{14}K_{15})$.
8. Wykonaj podstawienie: $C \leftarrow (R_8L_8)$.

Algorytm S2.2 Algorytm wyznaczania kluczy pośrednich (FEAL)

WEJŚCIE: 64-bitowy klucz szyfrujący $K = k_1k_2...k_{64}$.
WYJŚCIE: szesnaście 16-bitowych kluczy pośrednich K_i dla $i = 0, 1, ..., 15$.

1. Zainicjalizuj: $U^{(-2)} \leftarrow 0$, $U^{(-1)} \leftarrow k_1...k_{32}$, oraz $U^{(0)} \leftarrow k_{33}...k_{64}$.
2. Niech $U = U_0U_1U_2U_3$. Wówczas oblicz $K_0, ..., K_{15}$, przy $i = 1, ..., 8$, według poniższego schematu:
   (a) $U \leftarrow f_K(U^{(i-2)}, U^{(i-1)} \oplus U^{(i-3)})$. Funkcja f_K zdefiniowana jest przy pomocy Tabeli S2.1, gdzie symbole A i B oznaczają 4-bajtowe wektory $A = A_0A_1A_2A_3$ i $B = B_0B_1B_2B_3$.

(b) Wyznacz klucze pośrednie jako: $K_{2i-2} = U_0U_1$ oraz $K_{2i-1} = U_2U_3$. Wykonaj następujące podstawienie $U^{(i)} \leftarrow U$.

Bezpieczeństwo metody
Algorytm FEAL posiada 64-bitowy klucz szyfrujący. Stąd liczba kombinacji kluczy (złożoność kombinatoryczna) wymagana podczas ataku metodą wyczerpującą (patrz strona 7) będzie równa $2^{64} \approx 18 \cdot 10^{18}$. Stosując inne metody (metodę różnicową lub liniową) złożoność tę można zredukować nawet do poziomu 2^{15}.

International Data Encryption Algorithm (IDEA)

W algorytmie IDEA szyfrowaniu podlegają 64-bitowe ($n = 64$) bloki tekstu jawnego do postaci 64-bitowych bloków szyfrogramu, przy czym w procesie kodowania wykorzystuje się klucze szyfrujące K o długości 128 bitów. Algorytm powstał w roku 1991 jako alternatywa dla metody DES. Znalazł on zastosowanie między innymi w takich narzędziach kryptograficznych jak PGP (z ang. $Pretty\ Good\ Privacy$), czy też OpenPGP.

Głównym założeniem projektowym dla metody IDEA było połączenie operacji charakterystycznych dla trzech różnym grup algebraicznych nad 16-bitowymi blokami danych. Wspomniane operacje to: alternatywa wykluczająca (XOR) oznaczana jako $\oplus$, dodawanie modulo 2^{16} oznaczane jako $\boxplus$ (można realizować według wzoru: $a \boxplus b = (a + b)$ AND 0FFFFh), oraz zmodyfikowane mnożenie modulo $2^{16} + 1$ oznaczane jako $a \odot b$.

Operację zmodyfikowanego mnożenia $a \odot b$ można zaimplementować w sposób następujący. Niech $c = ab = c_0 \cdot 2^{32} + c_H \cdot 2^{16} + c_L$, gdzie $0 \leqslant c_H, c_L < 2^{16}$ oraz $c_0 \in \{0, 1\}$. Składowe c_H i c_L można wyznaczyć podczas standardowego mnożenia 32-bitowego, natomiast $c_0 = 1$ tylko wtedy, gdy $a = b = 2^{16}$. Wówczas jako wynik mnożenia $c' = c \bmod (2^{16} + 1)$ przyjmuje się $c' = 1$. W przeciwnym wypadku: jeżeli $c_L \geqslant c_H$, jako wynik przyjmujemy $c' = c_L - c_H$, natomiast dla $c_L < c_H$ w wyniku otrzymujemy odpowiednio $c' = c_L - c_H + (2^{16} + 1)$.

Szyfrowanie
W procesie szyfrowania wykorzystuje się pięćdziesiąt dwa 16-bitowe klucze pośrednie $K_i^{(r)}$ dla parametrów: $i = 1, ..., 6$ oraz $r = 1, ..., 9$, które generuje się za pomocą Algorytmu S3.2 na podstawie 128-bitowego klucza szyfrującego K. Sam proces szyfrowania przebiega w ośmiu rundach, których trzon stanowią opisane wcześniej operacje XOR ($\oplus$), dodawanie modulo 2^{16} ($\boxplus$) oraz zmodyfikowane mnożenie modulo $(2^{16} + 1)$ ($\odot$). Algorytm S3.1 opisuje proces szyfrowania charakterystyczny dla metody IDEA.

Deszyfrowanie
W metodzie IDEA podczas deszyfrowania szyfrogramu wykorzystuje się Algorytm S3.1, przy czym klucze pośrednie K_i zastępowane są kluczami K_i' wyznaczanymi na podstawie K_i zgodnie z Tabelą S3.1. Operacja $-K_i$ oznacza tutaj addytywną inwersję mod 2^{16}, tj. $-K_i$ oblicza się jako $-K_i = (2^{16} - K_i)$ AND 0FFFFh. Natomiast operacja K_i^{-1} wskazuje inwersję multiplikatywną modulo $2^{16} + 1$ obliczaną przy pomocy rozszerzonego algorytmu Euklidesa (patrz Algorytmu S3.3). Algorytm ten zwraca w wyniku dwie liczby całkowite x i y takie, że $ax + by = NWD(a, b)$. Wówczas podstawiając za $a = 2^{16} + 1$ i za $b = K_i$, to największy wspólny dzielnik będzie równy 1 poza jednym przypadkiem, gdy $K_i = 0$. Stąd jako K_i^{-1}

wystarczy wziąć $K_i^{-1} = y$, a w przypadku, gdy $K_i = 0$ podstawić $K_i^{-1} = 0$.

runda r	$K_1'^{(r)}$	$K_2'^{(r)}$	$K_3'^{(r)}$	$K_4'^{(r)}$	$K_5'^{(r)}$	$K_6'^{(r)}$
$r = 1$	$(K_1^{(10-r)})^{-1}$	$-K_2^{(10-r)}$	$-K_3^{(10-r)}$	$(K_4^{(10-r)})^{-1}$	$K_5^{(9-r)}$	$K_6^{(9-r)}$
$2 \leqslant r \leqslant 8$	$(K_1^{(10-r)})^{-1}$	$-K_3^{(10-r)}$	$-K_2^{(10-r)}$	$(K_4^{(10-r)})^{-1}$	$K_5^{(9-r)}$	$K_6^{(9-r)}$
$r = 9$	$(K_1^{(10-r)})^{-1}$	$-K_2^{(10-r)}$	$-K_3^{(10-r)}$	$(K_4^{(10-r)})^{-1}$	———	———

Tabela S3.1 Sposób wyznaczania kluczy pośrednich $K_i'^{(r)}$ dla potrzeb deszyfrowania IDEA.

Wyznaczanie kluczy pośrednich

W danej metodzie do wyznaczania kluczy pośrednich K_i służy Algorytm S3.2. Wszystkie 52 klucze pośrednie wyznaczane są na podstawie 128-bitowego klucza szyfrującego K w procesie iteracyjnym bazującym na operacjach cyklicznych przesunięć bitowych w lewo.

Algorytm S3.1 Algorytm szyfrowania/deszyfrowania IDEA

WEJŚCIE: 64-bitowy blok tekstu jawnego: $m_1 m_2 ... m_{64}$; 128-bitowy klucz $K = k_1 k_2 ... k_{128}$.
WYJŚCIE: 64-bitowy blok tekstu zaszyfrowanego $C = C_1 C_2 C_3 C_4$.

1. Oblicz za pomocą Algorytmu S3.2 16-bitowe klucze pośrednie $K_1^{(r)}, ..., K_6^{(r)}$ dla rund $r = 1, ..., 8$, a także klucze pośrednie $K_1^{(9)}, ..., K_4^{(9)}$ dla ostatniego kroku wyjściowego algorytmu.
2. Podstaw $X_1 X_2 X_3 X_4 \leftarrow m_1 ... m_{16} m_{17} ... m_{32} m_{33} ... m_{48} m_{49} ... m_{64}$, gdzie X_i oznacza blok o długości 16 bitów.
3. Dla rund $r = 1, ..., 8$ wykonaj następujące kroki obliczeniowe:
   (a) $X_1 \leftarrow X_1 \odot K_1^{(r)}$, $X_4 \leftarrow X_4 \odot K_4^{(r)}$, $X_2 \leftarrow X_2 \boxplus K_2^{(r)}$, $X_3 \leftarrow X_3 \boxplus K_3^{(r)}$.
   (b) $t_0 \leftarrow K_5^{(r)} \odot (X_1 \oplus X_3)$, $t_1 \leftarrow K_6^{(r)} \odot (t_0 \boxplus (X_2 \oplus X_4))$, $t_2 \leftarrow t_0 \boxplus t_1$.
   (c) $X_1 \leftarrow X_1 \oplus t_1$, $X_4 \leftarrow X_4 \oplus t_2$, $a \leftarrow X_2 \oplus t_2$, $X_2 \leftarrow X_3 \oplus t_1$, $X_3 \leftarrow a$.
4. Wykonaj przekształcenia końcowe nst. postaci: $C_1 \leftarrow X_1 \odot K_1^{(9)}$, $C_4 \leftarrow X_4 \odot K_4^{(9)}$, $C_2 \leftarrow X_3 \boxplus K_2^{(9)}$, $C_3 \leftarrow K_2 \boxplus K_3^{(9)}$.

Algorytm S3.2 Algorytm wyznaczania kluczy pośrednich (IDEA)

WEJŚCIE: 128-bitowy klucz szyfrujący $K = k_1 k_2 ... k_{128}$.
WYJŚCIE: pięćdziesiąt dwa 16-bitowe klucze pośrednie $K_i^{(r)}$ dla $i = 1, ..., 6$ i $r = 1, ..., 8$ oraz $K_i^{(9)}$ dla parametru $i = 1, ..., 4$.

1. Uszereguj klucze pośrednie w kolejności: $K_1^{(1)} ... K_6^{(1)} K_1^{(2)} ... K_6^{(2)} K_6^{(1)} ... K_6^{(8)} K_1^{(9)} ... K_4^{(9)}$.

2. Podziel klucz K na bloki 16-bitowe. Przypisz wprost pierwszym ośmiu kluczom pośrednim osiem pierwszych 16-bitowych bloków klucza szyfrującego K.

3. Wykonuj następujące operacje dopóki wszystkie 52 klucze pośrednie nie zostaną zapisane: cyklicznie przesuń w lewo klucz K o 25 bitów, podstaw pierwszych osiem bloków do kolejnych (dotychczas nie przypisanych) kluczy pośrednich.

Algorytm S3.3 Rozszerzony algorytm Euklidesa

WEJŚCIE: dwie dodatnie liczby całkowite a i b, przy czy, $a \geqslant b$
WYJŚCIE: $d = NWD(a, b)$ i dwie liczby x i y spełniające nst. równanie: $ax + by = d$.

1. Jeżeli $b = 0$ to ustaw $d \leftarrow a$, $x \leftarrow 1$, $y \leftarrow 0$. Koniec.

2. Ustaw $x_2 \leftarrow 1$, $x_1 \leftarrow 0$, $y_2 \leftarrow 0$ i $y_1 \leftarrow 1$.

3. Dopóki $b > 0$ wykonuj następujące kroki:

   (a) $q \leftarrow [a/b]^{\dagger}$, $r \leftarrow a - qb$, $x \leftarrow x_2 - qx_1$, $y \leftarrow y_2 - qy_1$.

   (b) $a \leftarrow b$, $b \leftarrow r$, $x_2 \leftarrow x_1$, $x_1 \leftarrow x$, $y_2 \leftarrow y_1$ oraz $y_1 \leftarrow y$.

Bezpieczeństwo metody

Metoda szyfrowania danych IDEA jest uważana za jedną z najbezpieczniejszych metod o specyfikacji udostępnionej publicznie. W przypadku ataku wyczerpującego złożoność kombinatoryczna wynosi $2^{128} \approx 34 \cdot 10^{37}$. Metoda ta skutecznie opiera się również kryptoanalizie różnicowej i liniowej.

Advanced Encryption Standard (AES)

Metoda szyfrowania danych AES (zwana również Rijndael) operuje na 128-bitowych blokach danych, generując w kilku rundach 128-bitowe bloki tekstu zaszyfrowanego. W procesie kodowania wykorzystuje się klucze o długościach: 128, 192 lub 256 bitów. W zależności od długości klucza zmianie podlega wyłącznie liczba rund, która dla kluczy 128 bitowych wynosi 10, dla kluczy 192 bitowych 12, oraz 14 dla kluczy o długości 256 bitów.

Algorytm AES został opracowany w roku 2001 przez Narodowy Instytut Standardów i Technologii USA jako następca popularnej metody szyfrowania DES. Prace nad rozwojem metody prowadzone były przez dwóch belgijskich kryptografów: J. Daeman, V. Rijmen. Alternatywna nazwa metody, tzn. Rijndael (czyt. Reindal), powstała jako zbitek fragmentów nazwisk obu naukowców. Algorytm AES został zarejestrowany jako standard o numerze kodowym FIPS PUB 197.

Metodę AES cechuje relatywnie niskie zapotrzebowania na pamięć operacyjną oraz prostota implementacyjna zarówno sprzętowa jak i programowa. Algorytm ten jest obecnie wdrażany na szeroką skalę wszędzie tam, gdzie swoje zastosowanie znajdują blokowe szyfry symetryczne.

[†]Operator $[a/b]$ zwraca część całkowitą z dzielenia a przez b.

Szyfrowanie

Proces szyfrowania wiadomości zgodnie z algorytmem AES przebiega w kilku rundach. Dla klucza o długości 256 bitów liczba rund wynosi 14. W każdej rundzie przeprowadza się pewne operacje zorientowane bajtowo. Są to kolejno: podstawienia bajtowe opisane za pomocą Tablic S4.1 i S4.2 (tzw. S-boksy), operacje cyklicznych przesunięć 4-bajtowych bloków danych, operacja przestawiania 4-bajtowych bloków danych, oraz sumowanie modulo 2 bloków danych z kluczami pośrednimi generowanymi zgodnie z Algorytmem S4.2. Proces szyfrowania opisany został za pomocą Algorytmu S4.1.

Deszyfrowanie

Proces deszyfrowania wiadomości na podstawie tekstu zakodowanego można przeprowadzić za pomocą algorytmu o bardzo zbliżonej strukturze do algorytmu szyfrującego (patrz Algorytm S4.1). Zmiany dotyczą następujących elementów: klucze pośrednie muszą być podawane w odwrotnej kolejności, operacje obrotów cyklicznych w lewo zamieniane są na operacje obrotów cyklicznych w prawo, podstawienia bajtów (S-boksy) w macierzy stanów wykonuje się według Tablicy S4.2, w procesie mieszania kolumn macierzy stanu wykorzystuje się macierz współczynników innej postaci, oraz kolejność stosowania poszczególnych operacji ulega zmianie. Algorytm S4.3 opisuje proces deszyfrowania wiadomości.

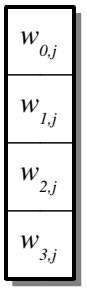
Wyznaczanie kluczy pośrednich

W procesie szyfrowania/deszyfrowania danych według metody AES dla liczby rund równej 14 potrzeba 60 kluczy pośrednich o długości 32-bitów każdy. Klucze te wyznacza się na podstawie 256-bitowego klucza szyfrującego K. Proces generowania kluczy pośrednich korzysta z następujących operacji: rotacja cykliczna słowa 32-bitowego w lewo o 8 bitów, podstawienia (S-boksów) bajtów w słowach 32-bitowych według Tablicy S4.1, oraz operacji sum bitowych modulo 2 (XOR). Sposób wyznaczania kluczy pośrednich dla tej metody szyfrowania opisuje Algorytm S4.2.

Algorytm S4.1 Algorytm szyfrowania AES z kluczem 256-bitowym

WEJŚCIE: 128-bitowy blok tekstu jawnego: $M = m_1...m_{128}$; 256-bitowy klucz $K = k_1...k_{256}$.
WYJŚCIE: 128-bitowy blok tekstu zaszyfrowanego $C = c_1c_2...c_{128}$.

1. Za pomocą Algorytmu S4.2 oblicz na podstawie klucza szyfrującego K sześćdziesiąt 32-bitowych kluczy pośrednich postaci



dla $j = 0, 1, ..., 59$.

2. Dokonaj podziału tekstu jawnego M na 1-bajtowe bloki, tzn. $M = M_1 M_2 ... M_{16}$. Przedstaw tekst jawny w postaci macierzy stanu S o wymiarze 4 na 4 elementy według

$S=$

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

następującego schematu: $s_{0,0} = M_1$, $s_{1,0} = M_2$, $s_{2,0} = M_3$, $s_{3,0} = M_4$, $s_{0,1} = M_5$, $s_{1,1} = M_6$, ... , $s_{0,2} = M_9$, ... , $s_{3,3} = M_{16}$.

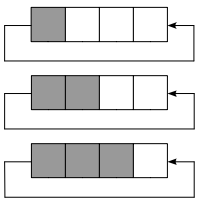
3. Dodaj do poszczególnych kolumn macierzy stanu S cztery klucze pośrednie o indeksach $j = 0, 1, 3$ i 4. Dodawanie należy wykonać jako bitowe dodawanie modulo 2 (operacja bitowa XOR).

4. Wykonaj 13 rund dla $r = 1, ..., 13$ obejmujących następujące operacje:

   (a) Podstawienia bajtów w macierzy stanów (S-boksy). Dla każdego elementu macierzy $s_{i,j}$ dokonaj jego podziału na dwa bloki 4-bitowe n_w i n_k, przy czym n_w będzie liczbą całkowitą bez znaku w kodzie naturalnym zbudowaną ze starszych 4 bitów $s_{i,j}$, tzn. $n_w = (s_{i,j} >> 4)$, natomiast n_k będzie liczbą całkowitą bez znaku w kodzie naturalnym zbudowaną z młodszych 4 bitów $s_{i,j}$, tzn. $n_k = (s_{i,j} \text{ AND } 0\text{Fh})$. Następnie używając n_w i n_k odpowiednio jako numery wierszy i kolumn w Tablicy S4.1 odczytaj z niej nowe wartości dla $s_{i,j}$.

   (b) Obroty cykliczne w lewo wierszy macierzy stanu S. Dla każdego wiersza o indeksie i (oprócz wiersza pierwszego, tzn. wiersza o indeksie $i = 0$) wykonaj cykliczną rotację jego elementów w lewo o i pozycji.

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$



$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,1}$	$s_{1,2}$	$s_{1,3}$	$s_{1,0}$
$s_{2,2}$	$s_{2,3}$	$s_{2,0}$	$s_{2,1}$
$s_{3,3}$	$s_{3,0}$	$s_{3,1}$	$s_{3,2}$

   (c) Przemieszanie kolumn macierzy stanu. Dla każdej kolumny macierzy S o indeksie j oblicz jej nową postać wykonując następujące mnożenie macierzowe:

$$\begin{bmatrix} s'_{0,j} \\ s'_{1,j} \\ s'_{2,j} \\ s'_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{bmatrix},$$

przy czym mnożenie ($\bullet$) jest realizowane w grupie wielomianów modulo 8, a operacja dodawania ($\oplus$) obliczana bitowo w sensie modulo 2 (XOR). Stąd wynik mnożenia kolumny macierzy stanu przez powyższą macierz można w przypadku ogólnym zapisać jako:

$$s'_{0,j} = (\{02\} \bullet s_{0,j}) \oplus (\{03\} \bullet s_{1,j}) \oplus s_{2,j} \oplus s_{3,j},$$

$$s'_{1,j} = s_{0,j} \oplus (\{02\} \bullet s_{1,j}) \oplus (\{03\} \bullet s_{2,j}) \oplus s_{3,j},$$

$$s'_{2,j} = s_{0,j} \oplus s_{1,j} \oplus (\{02\} \bullet s_{2,j}) \oplus (\{03\} \bullet s_{3,j}),$$

$$s'_{3,j} = (\{03\} \bullet s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (\{02\} \bullet s_{3,j}).$$

Poniższy przykład prezentuje sposób realizacji mnożeń $\bullet$. Przyjmijmy, że wymnażane są dwie liczby $\{57\mathrm{h}\}$ i $\{83\mathrm{h}\}$. Liczby te można zapisać w postaci wielomianów przyjmując za współczynniki wielomianu bity ich rozwinięcia w postaci kodu naturalnego, a za potęgi x numery bitów (bity numerujemy od 7 do 0). Stąd możemy zapisać

$$\{53\mathrm{h}\} \;=\; (01010111)_{(2)} = x^6 + x^4 + x^2 + x + 1,$$

$$\{83\mathrm{h}\} \;=\; (10000011)_{(2)} = x^7 + x + 1.$$

Wielomiany te należy wymnożyć przez siebie, przy czym suma współczynników przy x o tej samej potędze jest liczona modulo 2 (XOR). W naszym przypadku otrzymujemy

$$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) =$$

$$= \; x^{13} + x^{11} + x^9 + x^8 + \underbrace{x^7 + x^7}_{1\oplus 1=0} + x^6 + x^5 + x^4 + x^3 + \underbrace{x^2 + x^2}_{1\oplus 1=0} + \underbrace{x + x}_{1\oplus 1=0} +$$

$$+ \; 1 = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1.$$

Tak otrzymany wynik jest następnie dzielony modulo $(x^8 + x^4 + x^3 + x + 1)$. Dzielenie to wykonuje się jak zwykłe dzielenie wielomianów z tą różnicą, że operacje dodawania (odejmowania) są realizowane modulo 2. Jako wynik bierzemy resztę z dzielenia. Stąd dla naszego przykładu mamy

$$\begin{array}{l} \qquad\qquad\qquad\quad x^5 + x^3 \\ \overline{(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1)} : (x^8 + x^4 + x^3 + x + 1) \\ \underline{-x^{13} - x^9 - x^8 - x^6 - x^5} \\ \qquad\qquad x^{11} + x^4 + x^3 + 1 \\ \qquad\qquad \underline{-x^{11} - x^7 - x^6 - x^4 - x^3} \\ \qquad\qquad\qquad\qquad\quad x^7 + x^6 + 1. \end{array}$$

W wyniku jako resztę z dzielenia otrzymujemy wielomian x^7+x^6+1. Wielomian ten przekształcamy na postać binarną uzyskując wartość $\{\mathrm{c1h}\}$. Stąd wynik mnożenia

dla liczb $\{57\mathrm{h}\}$ i $\{83\mathrm{h}\}$ wynosi $\{57\mathrm{h}\} \bullet \{83\mathrm{h}\} = \{\mathrm{c1h}\}$. Operację mnożenia $\bullet$ można opisać w postaci następujących kroków:
Niech a i b będą liczbami 8-bitowymi, z będzie liczbą 16-bitową, obliczamy $(a \bullet b)$, gdzie z reprezentuje wielomian przez który dzielimy. Dla danego przypadku mamy $z = 11\mathrm{bh}$ (ze względu na dzielenie modulo $x^8 + x^4 + x^3 + x + 1$).

i. Podstaw $u \leftarrow 0$, n równa się najwyższemu numerowi bitu o wartości jeden w liczbie z (bity numerujemy od 7 do 0).
ii. Dla i zmieniającego się od 7 do 0 testuj i-ty bit liczby y i jeżeli jest on równy jedności to podstaw $u \leftarrow u \oplus (x << i)$, gdzie "<<" oznacza przesunięcie bitowe w lewo.
iii. Wykonuj operacje w punktach iv i v dopóki l jest różne od zera.
iv. Podstaw $l \leftarrow 0$.
v. Dla i zmieniającego się od 15 do n testuj i-te bity liczby u. Jeżeli bit na i-tej pozycji jest równy jeden to: podstaw $u \leftarrow u \oplus (z << (i - n))$ oraz $l \leftarrow 1$.
vi. Otrzymana wartość u jest szukanym wynikiem mnożenia.

(d) Operacja dodawania klucza pośredniego. Dodaj do poszczególnych kolumn macierzy stanu S cztery klucze pośrednie o indeksach $j = r, r + 1, r + 3$ i $r + 4$. Dodawanie należy wykonać jako bitowe dodawanie modulo 2 (XOR).

5. Wykonaj operację podstawienia bajtów macierzy stanu tak jak w punkcie 4.a.

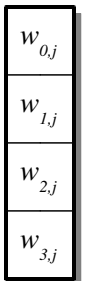
6. Obróć cyklicznie wektory macierzy stanu jak w punkcie 4.b.

7. Dodaj do poszczególnych kolumn macierzy stanu klucze pośrednie o następujących indeksach $j = 56, 57, 58$ i 59.

8. Niech $C = C_1C_2...C_{16}$ oznacza tekst zaszyfrowany podzielony na bloki długości jednego bajta. Przepisz do C wynik procesu szyfrowania według nst. schematu: $C_1 = s_{0,0}$, $C_2 = s_{1,0}$, $C_3 = s_{2,0}$, $C_4 = s_{3,0}$, , $C_5 = s_{0,1}$, $C_6 = s_{1,1}$, ... , $C_9 = s_{0,2}$, ... , $C_{16} = s_{3,3}$.

Algorytm S4.2 Algorytm wyznaczania kluczy pośrednich (AES)

WEJŚCIE: 256-bitowy klucz szyfrujący $K = k_1k_2...k_{256}$.
WYJŚCIE: sześćdziesiąt 32-bitowych kluczy pośrednich postaci



dla $j = 0, 1, ..., 59$.

1. Kluczom pośrednim o indeksach $j = 0, ..., 7$ przypisz: $w_{0,j}...w_{3,j} = k_{(1+32*j)}...k_{(32+32*j)}$.

2. Dla $j = 8, 9, ..., 59$ wykonuj następujące operacje:

   (a) Przedstaw klucz o indeksie $(j-1)$ jako liczbę 32-bitową $w = w_{0,(j-1)}...w_{3,(j-1)}$.

   (b) Jeżeli reszta z dzielenia j przez 8 jest równa zeru, to:

      i. Obróć w cyklicznie w lewo o 8 bitów, tzn. $w \leftarrow w \hookleftarrow 8$.

      ii. Dla każdego elementu składowego w o długości jednego bajta, tzn. dla elementów $w_{0,(j-1)}, w_{1,(j-1)}, w_{2,(j-1)}$ i $w_{3,(j-1)}$, odczytaj ich nowe wartości w Tablicy S4.1, przy czym numer wiersza n_w i numer kolumny n_k oblicz jako: $n_w = (w_{i,(j-1)} >> 4)$ (starsze 4 bity) oraz $n_k = (w_{i,(j-1)}$ AND 0Fh) (młodsze cztery bity), gdzie parametr $i = 0, 1, 2$ i 3.

      iii. Wykonaj operację XOR wartości w ze stałą $010000\text{h} << ([j/8] - 1)$, gdzie operacja $[\cdot]$ oznacza część całkowitą z dzielenia. Zatem operację tę można zapisać jako $w \leftarrow w$ XOR $(01000000\text{h} << ([j/8] - 1))$.

   (c) W przeciwnym wypadku, jeżeli reszta z dzielenia j przez 8 będzie równa 4, wykonaj operację z punktu b.ii.

   (d) Kluczowi o indeksie j przypisz nową wartość w postaci: wynik operacji XOR dla klucza o indeksie $j - 8$ i obliczonej wartości w.

		n_k															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
n_w	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Tabela S4.1 Podstawienia wartości macierzy stanu dla procesu szyfrowania (wszystkie liczby podano w formacie heksadecymalnym).

Algorytm S4.3 Algorytm deszyfrowania AES z kluczem 256-bitowym

WEJŚCIE: 128-bitowy blok tekstu zaszyfrowanego: $C = c_1c_2...c_{128}$; 256-bitowy klucz $K = k_1...k_{256}$.
WYJŚCIE: 128-bitowy blok tekstu jawnego: $M = m_1...m_{128}$.

1. Za pomocą Algorytmu S4.2 oblicz na podstawie klucza szyfrującego K sześćdziesiąt 32-bitowych kluczy pośrednich postaci

$w_{0,j}$
$w_{1,j}$
$w_{2,j}$
$w_{3,j}$

dla $j = 0, 1, ..., 59$.

2. Dokonaj podziału tekstu zaszyfrowanego C na 1-bajtowe bloki, tzn. $C = C_1C_2...C_{16}$. Przedstaw tekst jawny w postaci macierzy stanu S o wymiarze 4 na 4 elementy według

$S=$

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

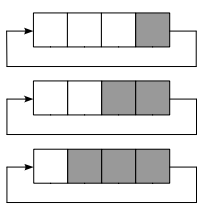
następującego schematu: $s_{0,0} = C_1$, $s_{1,0} = C_2$, $s_{2,0} = C_3$, $s_{3,0} = C_4$, $s_{0,1} = C_5$, $s_{1,1} = C_6$, ... , $s_{0,2} = C_9$, ... , $s_{3,3} = C_{16}$.

3. Dodaj do poszczególnych kolumn macierzy stanu S cztery klucze pośrednie o indeksach $j = 56, 57, 58$ i 59. Dodawanie należy wykonać jako bitowe dodawanie modulo 2 (operacja bitowa XOR).

4. Wykonaj 13 rund dla $r = 13, 12, ..., 1$ obejmujących następujące operacje:

   (a) Obroty cykliczne w prawo wierszy macierzy stanu S. Dla każdego wiersza o indeksie i (oprócz wiersza pierwszego, tzn. wiersza o indeksie $i = 0$) wykonaj cykliczną rotację jego elementów w prawo o i pozycji.

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$



$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,1}$	$s_{1,2}$	$s_{1,3}$	$s_{1,0}$
$s_{2,2}$	$s_{2,3}$	$s_{2,0}$	$s_{2,1}$
$s_{3,3}$	$s_{3,0}$	$s_{3,1}$	$s_{3,2}$

(b) Podstawienia bajtów w macierzy stanów (S-boksy). Dla każdego elementu macierzy $s_{i,j}$ dokonaj jego podziału na dwa bloki 4-bitowe n_w i n_k, przy czym n_w będzie liczbą całkowitą bez znaku w kodzie naturalnym zbudowaną ze starszych 4 bitów $s_{i,j}$, tzn. $n_k = (s_{i,j} >> 4)$, natomiast n_k będzie liczbą całkowitą bez znaku w kodzie naturalnym zbudowaną z młodszych 4 bitów $s_{i,j}$, tzn. $n_k = (s_{i,j}\ \mathrm{AND}\ 0\mathrm{Fh})$. Następnie używając n_w i n_k odpowiednio jako numery wierszy i kolumn w Tablicy S4.2 odczytaj z niej nowe wartości dla $s_{i,j}$.

(c) Operacja dodawania klucza pośredniego. Dodaj do poszczególnych kolumn macierzy stanu S cztery klucze pośrednie o indeksach $j = r, r+1, r+3$ i $r+4$. Dodawanie należy wykonać jako bitowe dodawanie modulo 2 (operacja XOR).

(d) Odwrotne przemieszanie kolumn macierzy stanu. Dla każdej kolumny macierzy S o indeksie j oblicz jej nową postać wykonując następujące mnożenie macierzowe:

$$\begin{bmatrix} s'_{0,j} \\ s'_{1,j} \\ s'_{2,j} \\ s'_{3,j} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \cdot \begin{bmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{bmatrix},$$

przy czym mnożenie ($\bullet$) jest realizowane w grupie wielomianów modulo 8, a operacja dodawania ($\oplus$) obliczana bitowo w sensie modulo 2 (XOR) (patrz Algorytm S4.1, punkt 4.c).

5. Obróć cyklicznie wektory macierzy stanu jak w punkcie 4.a.

6. Wykonaj operację podstawienia bajtów macierzy stanu tak jak w punkcie 4.b.

7. Dodaj do poszczególnych kolumn macierzy stanu klucze pośrednie o następujących indeksach $j = 0, 1, 2$ i 3.

8. Niech $M = M_1 M_2 ... M_{16}$ oznacza tekst jawny podzielony na bloki długości jednego bajta. Przepisz do M wynik procesu deszyfrowania według nst. schematu: $M_1 = s_{0,0}$, $M_2 = s_{1,0}$, $M_3 = s_{2,0}$, $M_4 = s_{3,0}$, , $M_5 = s_{0,1}$, $M_6 = s_{1,1}$, ... , $M_9 = s_{0,2}$, ... , $M_{16} = s_{3,3}$.

Bezpieczeństwo metody

Algorytm AES jest algorytmem o bardzo wysokim poziomie bezpieczeństwa. Zgodnie ze wskazaniami Narodowej Agencji Bezpieczeństwa (NSA) Stanów Zjednoczonych klucze o długości 128-bitów mogą być wykorzystywane do ochrony danych utajnionych, natomiast dane ściśle tajne powinny być szyfrowane za pomocą kluczy 192 lub 256-bitowych. Ponadto algorytm AES został zatwierdzony przez NSA jako narzędzie do ochrony narodowych systemów bezpieczeństwa.

Obecnie nie stwierdzono udanych ataków na metodę AES. Pewne przypuszczenia i oszacowania złożoności takich ataków (dot. złożoności mniejszych niż dla ataku wyczerpującego) mają charakter spekulatywny.

		n_k															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
n_w	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Tabela S4.2 Podstawienia wartości macierzy stanu dla procesu deszyfrowania (wszystkie liczby podano w formacie heksadecymalnym).

Literatura

1. Bauer F. L., *Sekrety kryptografii*, Wydawnictwo Helion, Gliwice 2002.
2. Menezes A. J., van Oorschot P. C., Vanstone S. A., *Kryptografia stosowana*, Wydawnictwo WNT, Warszawa 2005.
3. *Announcing the Advanced Encryption Standard (AES)*, Federal Information Processing Standards Publication 197, 2001.