

Na prawach rękopisu

**INSTYTUT CYBERNETYKI TECHNICZNEJ
POLITECHNIKI WROCŁAWSKIEJ**

Raport serii: PREPRINTY nr 100/98

**Metody konstruowania
generatorów permutacji
dla zastosowań kryptograficznych**

(rozprawa doktorska)

mgr inż. Leszek WOLANIUK

PROMOTOR:

**dr hab. inż. Czesław Kościelny,
prof. PZ**

Słowa kluczowe:

- kryptografia**
- ciała Galois**
- permutacje**

WROCŁAW 1998

Biblioteka Główna i OIMT
Politechniki Wrocławskiej



001108280



300 195 L/1

Akc. 503 p/99

SPIS TREŚCI

1. Wstęp	5
1.1 Generatory permutacji	6
1.2 Generowanie permutacji za pomocą układów do rachowania w ciałach Galois charakterystyki 2	8
1.3 Cel i zakres pracy	10
1.4 Organizacja pracy	11
2. Wprowadzenie matematyczne	13
2.1 Metoda realizacji działań w ciałach Galois charakterystyki 2	13
2.2 Permutacje	24
3. Metoda generowania par permutacji wzajemnie odwracalnych zbioru 2^m - elementowego z zastosowaniem przekształceń afinicznych przestrzeni liniowych na ciałami Galois charakterystyki 2	29
3.1 Przekształcenia afiniczne przestrzeni liniowych nad $GF(q)$ charakterystyki 2	29
3.2 Przekształcenia afiniczne przestrzeni liniowych nad $GF(2)$	36
3.3 Realizacja układów do generowania permutacji za pomocą przekształceń afinicznych ..	40
3.4 Własności permutacji wygenerowanych za pomocą przekształceń afinicznych	41
4. Metoda generowania samoodwracalnych permutacji zbioru 2^m - elementowego	45
4.1 Permutacje samoodwracalne	45
4.2 Inwersja w $GF(2^m)$ jako permutacja samoodwracalna zbioru 2^m - elementowego	47
4.3 Inwersja w $GF(q^c)$, $q = 2^k$, $kc = m$ jako permutacja samoodwracalna zbioru 2^m - elementowego	50
5. Układy do podnoszenia do kwadratu i pierwiastkowania kwadratowego w $GF(2^m)$ jako uniwersalne generatory permutacji o charakterze liniowym	58
5.1 Podnoszenie do kwadratu i pierwiastkowanie kwadratowe w ciałach Galois charakterystyki 2	58
5.2 Wzajemnie odwracalne permutacje zbioru 2^m - elementowego generowane za pomocą potęgowania w $GF(2^m)$ z wykładnikiem 2^k i 2^{-k}	60
5.3 Generowanie permutacji samoodwracalnych zbioru 2^m - elementowego za pomocą układów do wielokrotnego podnoszenia do kwadratu w $GF(2^m)$, m - liczba parzysta	66
6. Hybrydowa metoda generowania par permutacji wzajemnie odwracalnych i permutacji samoodwracalnych zbioru 2^n - elementowego	70
6.1 Założenia i definicje	71
6.2 Łączenie bloków permutacyjnych	74
6.2.1 Kolejność bloków permutacyjnych	74
6.2.2 Konfiguracje bloków permutacyjnych	75
6.2.3 Uwarunkowania strukturalne konstruowania wzajemnie odwracalnych i samoodwracalnych hybrydowych generatorów permutacji	79
6.3 Analiza własności permutacji generowanych metodą hybrydową	82
6.3.1 Założenia i oprogramowanie pomocnicze	82
6.3.2 Wyniki testów	82

7. Konstruowanie generatorów permutacji dla zastosowań kryptograficznych	87
7.1 Charakterystyka metod szyfrowania	87
7.2 Zacieranie charakterystycznych cech statystycznych tekstu jawnego w kryptogramie...	94
7.2.1 Kryteria weryfikacyjne s-skrzynek	95
7.3 Odporność na korelację.....	100
7.4 Generowanie ciągów losowych i pseudolosowych	104
8. Przykłady zastosowań - (symulacja)	108
8.1 Założenia	108
8.2 Kryptosystem.....	108
8.2.1 Budowa i działanie	108
8.2.2 S-skrzynki	111
8.2.3 Generator Podkluczy PSMR.....	115
8.3 Realizacja programowa	119
8.4 Wyniki testów	119
8.5 Ocena jakości zaprojektowanego systemu kryptograficznego	124
9. Zakończenie	125
9.1 Przebieg realizacji pracy	127
9.2 Perspektywy rozwoju tematyki	128
9.3 Podsumowanie i wnioski końcowe	129
Literatura.....	130
DODATEK A.....	135
DODATEK B.....	136
DODATEK C.....	138
DODATEK D	141

1. Wstęp

Kryptografia przez tysiące lat była sztuką zapewnienia poufności korespondencji realizowanej ogólnodostępnymi kanałami łączności. Jej rozwój w przeszłości był jednak zawsze zdeterminowany potrzebami nielicznej grupy społecznej, związanej przeważnie z dyplomacją lub wojskowością. Miało to swój wpływ na ograniczenie zakresu zastosowań systemów kryptograficznych i powodowało przez długie lata stagnację w rozwoju technik szyfrowania.

Wraz z rewolucją komputerową zmieniło się także społeczeństwo. Powstało duże zapotrzebowanie na przechowywanie obszernej ilości informacji osobistej, finansowej, komercyjnej, wojskowej i technologicznej w komputerowych bankach danych i przesyłanie jej między systemami komputerowymi.

Upowszechnienie się techniki mikrokomputerowej spowodowało jeszcze gwałtowniejszy wzrost znaczenia informacji jako towaru o wymiernej wartości, który znajdując się w powszechnym obiegu, wymaga także ochrony. Odzwierciedla się to w rosnącym zainteresowaniu technikami ograniczania dostępu do informacji poprzez jej szyfrowanie. Następstwem tego jest rozwój technik kryptograficznych, oferujących coraz to doskonalsze rozwiązania oparte na modelach matematycznych, stosujących systemy algebraiczne o takich cechach, które są łatwe do zastosowania w systemach komputerowych i jednocześnie posiadają właściwości gwarantujące zapewnienie poufności chronionej informacji.

Stosowanie szyfrowania w różnych mediach wykorzystujących technikę cyfrową spowodowało także zwiększenie wymagań co do szybkości działania systemów kryptograficznych. Pojawiło się wiele nowych obszarów zastosowań dla techniki komputerowej. Powszechne stały się kryptosystemy do szyfrowania obrazu i dźwięku, wykorzystywane w systemach audio-wizyjnych. Wiele standardów szyfrowania wprowadzonych w ostatnich latach na rynek należało zmodyfikować lub wręcz zaniechać ich używania, ponieważ przestały one odpowiadać coraz ostrzejszym normom technologicznym.

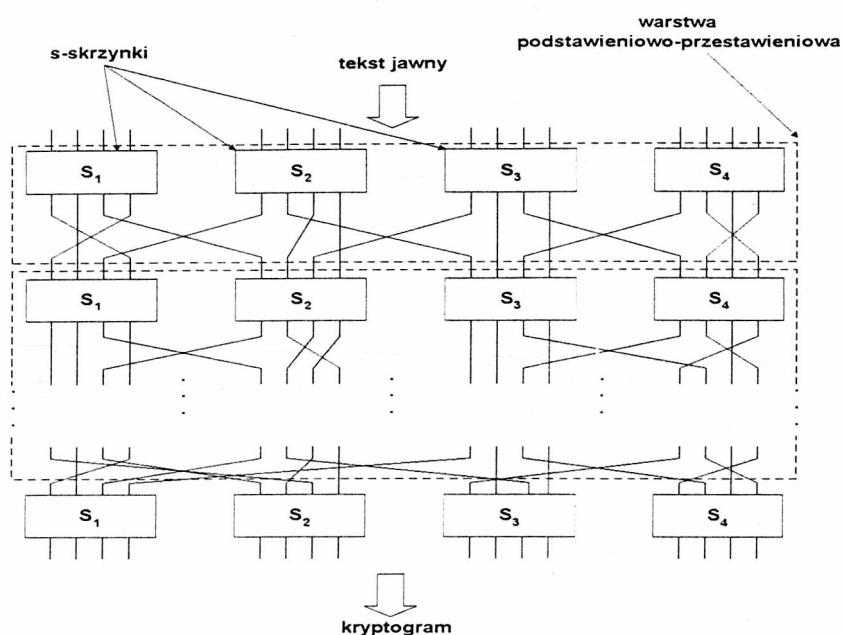
Bez względu na to, jak głębokie zmiany dokonały się w kryptografii, aktualna pozostała jednak sama idea użycia szyfru w procesach komunikacyjnych. Ma on zawsze za zadanie takie utajnienie wiadomości, aby tylko upoważniony adresat wiadomości mógł zrozumieć jej treść. Zastosowanie szyfrowania determinuje więc postać procesu przekazywania informacji, wprowadzając do niego elementy, które można przedstawić schematycznie za pomocą wzorów

$$f: P \rightarrow C; \quad f^{-1}: C \rightarrow P \quad (1.1)$$

Interpretując te wzory można powiedzieć, że tekst jawny P oraz kryptogram C w postaci słów nad pewnym alfabetem są przekształcane za pomocą funkcji szyfrującej f i funkcji deszyfrującej f^{-1} . Elementem weryfikującym upoważnienie do poprawnej deszyfracji jest w tym procesie klucz szyfrujący, który pozwala na zastosowanie poprawnej transformacji z postaci zaszyfrowanej w jawną. Każdy element tekstu jawnego i kryptogramu oraz klucza szyfrującego odwzorowany jest za pomocą obiektów matematycznych, które są argumentami i wartościami funkcji matematycznych, użytych w przekształceniach szyfrująco-deszyfrujących. Zbiór wartości tych obiektów jest skończony i ściśle określony. Szyfrowanie w najprostszej, a zarazem najczęściej używanej postaci polega zatem na wzajemnie jednoznacznym odwzorowaniu zbioru tych wartości na ten sam zbiór, a takie odwzorowanie nazywa się permutacją.

1.1 Generatory permutacji

Generatory permutacji stanowią podstawowy element wielu systemów kryptograficznych i stosowane są przede wszystkim w szyfrach blokowych do realizacji dwóch podstawowych technik zmniejszających redundancję tekstu jawnego: rozpraszania i mieszania. Technicznie realizuje się ten proces przez przetwarzanie bloku tekstu jawnego za pomocą szeregu warstw, zawierających tzw. skrzynki podstawieniowe (*ang. substitution boxes*) zwane dalej w pracy s-skrzynkami i przestawieniowe p-skrzynki (*ang. permutation boxes*), które razem tworzą tzw. sieć podstawieniowo - przestawieniową (*ang. SPN - Substitution-Permutation Network*).



Rys.1.1. Przykład sieci podstawieniowo-przestawieniowej

Szyfrowanie bloku tekstu jawnego odbywa się w takiej strukturze poprzez naprzemienne operacje podstawiania, realizowane przez s-skrzynki s_i i przestawiania fragmentów przetwarzanego bloku, które praktycznie wykonywane jest najczęściej poprzez krzyżowanie połączeń z kolejną warstwą. Kolejność występowania warstw podstawieniowo-przestawieniowych oraz ich struktura są tak zaprojektowane, aby cały układ realizował przekształcenie samoodwracalne tzn. aby szyfrowanie i deszyfrowanie odbywały się według tej samej kolejności warstw.

Na podobnej zasadzie pracuje stosowany od wielu lat system DES i wszystkie jego późniejsze modyfikacje (DES wielokrotny, uogólniony lub ze zmiennymi s-skrzynkami), oraz najnowsze proponowane aktualnie algorytmy kryptograficzne, takie jak m.in. NewDES, FEAL, IDEA, LOKI. Warto tu jednocześnie nadmienić, że współtwórcą tego ostatniego algorytmu jest uczony australijski polskiego pochodzenia, J. Pieprzyk.

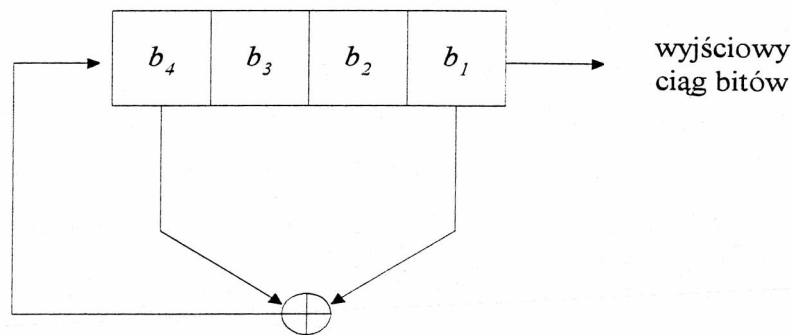
Przeglądając literaturę opisującą najnowsze proponowane szyfry blokowe można zaobserwować, że większość z nich objęta jest ochroną patentową, a informacje o zastosowanych w nich s-skrzynkach są bardzo skąpe i ograniczają się jedynie do opisu zasady ich działania oraz podania zestawu argumentów wraz z odpowiadającymi im wynikami zastosowanego przekształcenia [Den93][Sch95]. Stwarza to pewne utrudnienia inżynierom, zajmującym się ochroną danych w systemach i sieciach komputerowych, ponieważ muszą się oni liczyć z wydatkami na opłaty patentowe, gdyż nie dysponują narzędziem do projektowania podstawowego elementu szyfru jakim jest s-skrzynka.

Ostatnio pojawiło się kilka prac na ten temat [Bro93][Ada97][Mis96][Ser96] ale mają one raczej bardzo ogólny charakter i koncentrują się przede wszystkim na zasadach weryfikacji skrzynek podstawieniowych pod kątem ich własności mieszająco-rozpraszających, przez co na ich podstawie nie można zaprojektować odpowiedniego sprzętu. Dlatego też autor zajął się opracowaniem metod generowania permutacji, które można z powodzeniem zastosować do budowy szybkich urządzeń szyfrująco-deszyfrujących, opartych na zasadzie iloczynowego szyfru blokowego. Przez szyfr iloczynowy rozumie się tutaj szyfr przetwarzający blok danych za pomocą sekwencyjnego złożenia wielu różnych funkcji, działających na przetwarzanym bloku lub m jego fragmentach.

1.2 Generowanie permutacji za pomocą układów do rachowania w ciałach Galois charakterystyki 2.

Wybór przekształceń służących do konstruowania generatorów permutacji jest elementem decydującym o sposobie ich wykorzystania i przydatności do celów kryptograficznych. Niemal wszyscy autorzy prac dotyczących projektowania sieci podstawieniowo-przestawieniowych są zgodni, że do budowy s-skrzynek bardzo użyteczne są układy wykonujące działania w ciałach skończonych [Mat91][Rue86][Sch95], brak jednak w ogólnej dostępnej literaturze opisów konkretnych rozwiązań. W nielicznych pozycjach poświęconych tej tematyce [Mor97] autorzy zamieszczają jedynie proste przykłady, ogólnikowe opisy lub rozwiązania, na podstawie których nie można zaprojektować własnych, profesjonalnie działających, szybkich urządzeń kryptograficznych.

Układy wykonujące działania w ciałach Galois charakterystyki 2 można wykonywać za pomocą sieci standardowych bramek logicznych. Jednak ze względu na uwarunkowania technologiczne realizuje się je przede wszystkim z zastosowaniem urządzeń przetwarzających dane w sposób szeregowy, wykorzystując szeregowy rejestr przesuwający LFSR oraz zestawy bramek *exclusiv-OR* lub *AND*. Jest to naturalną konsekwencją zastosowania teorii kodów cyklicznych Bose-Chaudhuri-Hocquenghem'a (kod BCH) i Reeda-Solomona (kod RS). Do uzyskania wyniku działania przeprowadzanego na n - bitowym bloku informacyjnym potrzeba w tych układach co najmniej n cykli układu taktującego, czyli tyle ile wynosi długość rejestru. Generatory permutacji zrealizowane taką metodą wprowadzają więc znaczne opóźnienia w działaniu układów aplikacyjnych. Schemat przykładowego 4-bitowego generatora permutacji z wykorzystaniem rejestru LSFR zamieszczono na rys. 1.2.



Rys. 1.2. Przykład generatora permutacji opartego na wykorzystaniu rejestru LFSR.

W literaturze poświęconej konstruowaniu urządzeń do rachowania w ciałach skończonych zawęża się zakres rozwiązań do przypadków pozwalających na maksymalne uproszczenie układów [Mat91], co powoduje, że rozwiązania te nie nadają się do zastosowań kryptograficznych ze względu na niewielki zakres rozwiązań, które musiałby sprawdzić kryptoanalityk, aby złamać szyfr oparty na takich przekształceniach. Podobny problem występuje, gdy do konstruowania s-skrzynek rozważa się zastosowanie jedynie jednego lub dwóch przekształceń [Mor97].

Innym zagadnieniem dotyczącym projektowania układów do rachowania w ciałach Galois charakterystyki 2 jest ich wielkość mająca ścisły związek ze złożonością realizacji technicznych oraz odpornością na kryptoanalizę. W literaturze nie ma precyzyjnych opisów dotyczących projektowania s-skrzynek przetwarzających bloki o dużej liczbie bitów, co powoduje, że w praktyce stosuje się tylko małe s-skrzynki.

Ze względu na powyższe fakty autor niniejszej rozprawy opracował nowe metody służące do projektowania układów do rachowania w ciałach Galois charakterystyki 2 bez zastosowania rejestrów LSFR oraz ograniczeń wynikających z wymagań dotyczących prostoty rozwiązań. Proponowane metody można będzie wykorzystać do konstruowania s-skrzynek o praktycznie dowolnej wielkości i złożoności struktury realizujących liniowe i nieliniowe operacje matematyczne w ciałach $GF(q^c)$ charakterystyki 2 oraz w ciałach izomorficznych. Tak skonstruowane generatory permutacji, opisane precyzyjnie równaniami matematycznymi, łatwo będzie można poddać analizie odporności na metody kryptoanalityczne, a także projektować układy o innych pożądanых cechach np. zawierających ukryte tzw. „tajne przejścia”.

1.3 Cel i zakres pracy

Celem niniejszej rozprawy jest opracowanie takich metod generowania permutacji z zastosowaniem układów do rachowania w ciałach Galois charakterystyki 2, które będzie można wykorzystać do celów kryptograficznych w szybkich urządzeniach szyfrująco - deszyfrujących o architekturze równoległej. Aby to zrealizować należy rozwiązać następujące problemy:

- określić analityczne metody wyznaczania zależności opisujących działania arytmetyczne realizowane w $GF(q^m)$ charakterystyki 2 i w ciałach izomorficznych,
- przeanalizować właściwości permutacji wygenerowanych za pomocą działań arytmetycznych o charakterze liniowym i nieliniowym w ciałach Galois charakterystyki 2 i określić ich przydatność do generowania permutacji wzajemnie odwracalnych i samoodwracalnych,
- wyznaczyć sposoby łączenia różnych generatorów permutacji w celu generowania permutacji hybrydowych,
- zbadać przydatność zaprojektowanych generatorów permutacji do zastosowań kryptograficznych,
- zaprojektować szyfr iloczynowy z zastosowaniem opracowanych generatorów i wykonać badania symulacyjne tego szyfru.

Przyjęto następujący zakres prac służących do wypełnienia powyższych założeń :

- opracowanie metody generowania par permutacji wzajemnie odwracalnych zbioru 2^m -elementowego z zastosowaniem przekształceń afinicznych przestrzeni liniowych nad ciałami Galois charakterystyki 2,
- zastosowanie inwersji w $GF(2^m)$ i $GF(q^c)$, $kc = m$ do generowania permutacji samoodwracalnych,
- wykorzystanie układów do realizacji potęgowania w $GF(2^m)$ z wykładnikiem 2^k i 2^{-k} do generowania wzajemnie odwracalnych i samoodwracalnych permutacji zbioru 2^m -elementowego,
- sekwencyjne i równoległe łączenie generatorów permutacji w celu generowania permutacji hybrydowych,
- weryfikacja zaprojektowanych generatorów permutacji z zastosowaniem znanych kryteriów do badania przydatności s-skrzynek dla celów kryptograficznych,

- przeprowadzenie symulacji komputerowej samodzielnie zaprojektowanego blokowego szyfru iloczynowego z zastosowaniem hybrydowych generatorów permutacji, wykorzystujących działania w ciałach Galois charakterystyki 2.

1.4 Organizacja pracy

Niniejsza rozprawa zorganizowana jest w następujący sposób :

- rozdział pierwszy - wstęp - zawiera wprowadzenie do tematyki kryptograficznej oraz ciał Galois. Określono w nim także cel i założenia niniejszej pracy oraz podano układ niniejszej rozprawy,
- rozdział drugi - przedstawia matematyczne podstawy rachowania w ciałach Galois charakterystyki 2 oraz podstawowe wiadomości dotyczące permutacji,
- rozdział trzeci podaje metodę generowania par permutacji wzajemnie odwracalnych zbiorów 2^m -elementowych w oparciu o przekształcenia afiniczne przestrzeni liniowych nad ciałami Galois charakterystyki 2.
- rozdział czwarty - poświęcony jest zastosowaniu inwersji w $GF(2^m)$ do realizacji permutacji samoodwracalnych zbioru 2^m -elementowego. Przeanalizowano tu także możliwość realizacji takich układów dla $GF(q^c)$ gdzie $q = 2^k$, $kc = m$ oraz porównano właściwości tych dwóch rozwiązań,
- rozdział piąty omawia sposoby generowania permutacji samoodwracalnych i wzajemnie odwracalnych z zastosowaniem układów do podnoszenia do kwadratu i wyciągania pierwiastka kwadratowego w ciałach $GF(2^m)$,
- rozdział szósty prezentuje koncepcję hybrydowego generowania par permutacji wzajemnie odwracalnych i permutacji samoodwracalnych zbioru 2^m -elementowego, w oparciu o połączone układy zaprojektowane na podstawie metod opisanych w poprzednich rozdziałach,
- rozdział siódmy podaje opis sposobów stosowania generatorów permutacji w urządzeniach kryptograficznych oraz weryfikacji tych generatorów,
- rozdział ósmy - zawiera przykład zastosowania zaprojektowanych przez autora generatorów permutacji w rzeczywistym kryptosystemie o charakterze blokowego szyfru iloczynowego,
- rozdział dziewiąty - to podsumowanie osiągniętych wyników oraz wnioski końcowe,

Na ostatnich stronach niniejszej pracy podano wykaz literatury poświęconej tematyce związanej z przedmiotem pracy.

W dodatkach do pracy umieszczono tabelę oraz wydruki procedur, ilustrujące rozwiązania niektórych problemów projektowych zamieszczonych w rozprawie.

2. Wprowadzenie matematyczne

W rozdziale tym przedstawione zagadnienia, twierdzenia i definicje, które dotyczą bezpośrednio zakresu wiedzy niezbędnej w badaniach przeprowadzanych w ramach realizacji niniejszej pracy. Jest to zwięzłe określenie aparatu matematycznego zastosowanego do opisu przedstawionych w niniejszej rozprawie problemów.

Rozdział dotyczy metod realizacji działań w ciałach Galois oraz zawiera podstawowe twierdzenia i definicje dotyczące permutacji.

2.1 Metoda realizacji działań w ciałach Galois charakterystyki 2

Techniki rachowania w ciałach Galois charakterystyki 2 są dość dobrze opracowane, ale dla zastosowań do realizacji aparatury kodowania korekcyjnego [Mat91] [Men93] [Mar77] [Moc97]. Przyjęto tu zasadę, aby układy do realizacji działań były jak najmniej skomplikowane [Mce87] [Mat91] [Paa96]. Dlatego też do konstrukcji ciał stosuje się z reguły trójmiany pierwotne nad $GF(2)$ typu $x^n + x^k + 1$ (o ile takie istnieją), najlepiej jeśli $k=1$, lub pięciomiany pierwotne. Takich właśnie trójmianów lub pięciomianów, które umożliwiają realizację układów o minimalnej liczbie elementów jest bardzo mało. Mimo to, te same techniki rachowania w ciałach skończonych przyjęto dla zastosowań kryptograficznych [Lie86][Mor97]. Jest to poważny błąd metodologiczny, ponieważ bardzo ograniczona liczba układów do rachowania w ciele Galois ogranicza możliwość tworzenia wielu równoważnych pod względem odporności na złamanie systemów kryptograficznych. Dlatego też w zastosowaniach kryptograficznych powinno używać się technik rachowania w ciałach izomorficznych, co ogromnie zwiększa możliwość generowania kryptosystemów, a w kontekście tej pracy umożliwia wygenerowanie znacznie większej liczby permutacji. Wiadomo bowiem, że takich trójmianów lub pięciomianów stopnia m , które pozwalają na realizację działań w ciałach Galois za pomocą układów o minimalnej liczbie bramek jest dla każdego m co najwyżej kilka. Tymczasem funkcja $N_q(m)$, oznaczająca liczbę wielomianów nierozkładalnych stopnia m nad $GF(q)$ wyraża się wzorem

$$N_q(m) = \frac{1}{m} \sum_{d|m} \mu(d) q^{m/d} \quad (2.1)$$

gdzie

m - stopień wielomianu,

q - liczba elementów ciała,

d - dzielnik m ,

$\mu(d)$ - funkcja Möbiusa.

Funkcja $N_q(m)$ rośnie bardzo szybko ze wzrostem m , co ilustruje tabela 2.1.

Tabela 2.1.

m	$N_2(m)$	$N_4(m)$	$N_8(m)$	$N_{16}(m)$	$N_{32}(m)$	$N_{64}(m)$	$N_{128}(m)$	$N_{256}(m)$
2	1	6	28	120	496	2016	8128	32640
3	2	20	168	1360	10912	87360	699008	5592320
4	3	60	1008	16320	261888	4193280	67104768	$\approx 10^9$
5	6	204	6552	209712	6710880	214748352	6871947648	$\approx 2,2 \cdot 10^{11}$
6	9	670	43596	2795480	178951344	$\approx 1,1 \cdot 10^{10}$	$\approx 7,3 \cdot 10^{11}$	$\approx 4,7 \cdot 10^{13}$
7	18	2340	299592	38347920	4908534048	$\approx 6,3 \cdot 10^{11}$	$\approx 8 \cdot 10^{13}$	$\approx 10^{16}$
8	30	8160	20966640	536862720	$\approx 1,4 \cdot 10^{11}$	$\approx 3,5 \cdot 10^{13}$	$\approx 9 \cdot 10^{16}$	$\approx 2,3 \cdot 10^{18}$
9	56	29120	14913024	$\approx 7,6 \cdot 10^9$	$\approx 3,9 \cdot 10^{12}$	$\approx 2 \cdot 10^{15}$	$\approx 10^{18}$	$\approx 5,2 \cdot 10^{20}$
10	99	104754	107370900	$\approx 10^{11}$	$\approx 10^{14}$	$\approx 1,2 \cdot 10^{17}$	$\approx 1,2 \cdot 10^{20}$	$\approx 1,2 \cdot 10^{23}$
11	186	381300	780903144	$\approx 1,6 \cdot 10^{12}$	$\approx 3,3 \cdot 10^{15}$	$\approx 6,7 \cdot 10^{18}$	$\approx 1,4 \cdot 10^{22}$	$\approx 2,8 \cdot 10^{25}$
12	335	1397740	$\approx 5,7 \cdot 10^9$	$\approx 2,3 \cdot 10^{13}$	$\approx 9,6 \cdot 10^{16}$	$\approx 3,9 \cdot 10^{20}$	$\approx 1,6 \cdot 10^{24}$	$\approx 6,6 \cdot 10^{27}$

Istnieją też dość obszerne tablice wielomianów nieredukowalnych dla $q = 2$ [Pet72] [Gar70], niewielkie tablice dla $q = 2^m$, $m = 2, 3, 4$ [Gre74] oraz bardzo obszerne tablice (ponad 430 stron) dla $q = 2^m$, $m = 2, \dots, 10$ [Koś79].

Jeśli np. zachodzi potrzeba rachowania w ciele 64-elementowym, to można do jego konstrukcji zastosować 9 wielomianów stopnia 6 nad $GF(2)$, 20 wielomianów stopnia 3 nad $GF(4)$ lub 28 wielomianów stopnia 2 nad $GF(8)$ (patrz tabela 2.2.).

Tabela 2.2.

$GF(2^6)$	$GF(4^3)$		$GF(8^2)$		
$x^6 + x + 1$	$x^3 + 2$	$x^3 + 2x^2 + 1$	$x^2 + x + 1$	$x^2 + 3x + 6$	$x^2 + 6x + 1$
$x^6 + x^3 + 1$	$x^3 + 3$	$x^3 + 2x^2 + x + 3$	$x^2 + x + 3$	$x^2 + 3x + 7$	$x^2 + 6x + 2$
$x^6 + x^4 + x^2 + x + 1$	$x^3 + x + 1$	$x^3 + 2x^2 + 2x + 2$	$x^2 + x + 5$	$x^2 + 4x + 1$	$x^2 + 6x + 5$
$x^6 + x^4 + x^3 + x + 1$	$x^3 + 2x + 1$	$x^3 + 2x^2 + 3x + 2$	$x^2 + x + 7$	$x^2 + 4x + 3$	$x^2 + 6x + 6$
$x^6 + x^5 + 1$	$x^3 + 3x + 1$	$x^3 + 2x^2 + 3x + 3$	$x^2 + 2x + 1$	$x^2 + 4x + 4$	$x^2 + 7x + 2$
$x^6 + x^5 + x^2 + x + 1$	$x^3 + x^2 + 1$	$x^3 + 3x^2 + 1$	$x^2 + 2x + 2$	$x^2 + 4x + 6$	$x^2 + 7x + 3$
$x^6 + x^5 + x^3 + x^2 + 1$	$x^3 + x^2 + x + 2$	$x^3 + 3x^2 + x + 2$	$x^2 + 2x + 4$	$x^2 + 5x + 2$	$x^2 + 7x + 4$
$x^6 + x^5 + x^4 + x + 1$	$x^3 + x^2 + x + 3$	$x^3 + 3x^2 + 2x + 2$	$x^2 + 2x + 7$	$x^2 + 5x + 3$	$x^2 + 5x + 7$
$x^6 + x^5 + x^4 + x^2 + 1$	$x^3 + x^2 + 2x + 3$	$x^3 + 3x^2 + 2x + 3$	$x^2 + 3x + 4$	$x^2 + 5x + 6$	
	$x^3 + x^2 + 3x + 2$	$x^3 + 3x^2 + 3x + 3$	$x^2 + 3x + 5$	$x^2 + 7x + 5$	

Dla uproszczenia zapisu wielomianów w tabeli 2.2. zastosowano proste odwzorowanie zbioru kolejnych potęg elementu pierwotnego α ciała $GF(q) = GF(2^m)$ w zbiór liczb całko-

witych $\{0, 1, \dots, q-1\}$. Wiadomo, że $\alpha^i = [a_{i,0} \ a_{i,1} \ \dots \ a_{i,m-1}]$, gdzie $x^i \equiv a_{i,0} + a_{i,1}x + \dots + a_{i,m-1}x^{m-1} \pmod{p_m(x)}$, $p_m(x)$ - wielomian pierwotny stopnia m ponad $GF(2)$ zastosowany do konstrukcji ciała, oraz $i = 0, 1, \dots, 2^m - 2$. Dla uproszczenia tego zapisu przyjęto konwencję: $0 = [0 \ 0 \ \dots \ 0] \leftrightarrow 0$ i $\alpha^i = [a_{i,0} \ a_{i,1} \ \dots \ a_{i,m-1}] \leftrightarrow i + 1$.

Biorąc pod uwagę, że istnieją 2 wielomiany nierozkładalne stopnia 3 nad $GF(2)$, które można zastosować do konstrukcji ciała $GF(8)$, łatwo można utworzyć 56 izomorficznych ciał $GF(64)$. Ten właśnie prosty izomorfizm proponuje autor dla zwiększenia możliwości generowania permutacji.

Opisując ogólnie procedurę projektowania układów do realizacji działań w $GF(q^m)$ charakterystyki 2, należy przede wszystkim zająć się układami mnożenia, podnoszenia do kwadratu i obliczania odwrotności multiplikatywnej, ponieważ dodawanie jest trywialne.

Niech

$$P(x) = x^m + P_{m-1}x^{m-1} + \dots + P_1x + P_0, \quad (2.2)$$

oznacza wielomian nierozkładalny stopnia m nad $GF(q)$, $q = 2^k$, $k \geq 1$.

Wobec tego dwa dowolne elementy ciała $GF(q^m)$ można przedstawić za pomocą wielomianów nad $GF(q)$

$$\begin{aligned} A(x) &= A_0 + A_1x + \dots + A_{m-1}x^{m-1}, \\ B(x) &= B_0 + B_1x + \dots + B_{m-1}x^{m-1}. \end{aligned} \quad (2.3)$$

Iloczynem tych dwóch elementów $GF(q^m)$ jest wielomian

$$\begin{aligned} C(x) &= A(x) \cdot B(x) \pmod{P(x)} \equiv C_0 + C_1x + \dots + C_{m-1}x^{m-1} = \\ &= \left[\sum_{k=0}^{2m-2} x^k \left(\sum_{i+j=k} A_i B_j \right) \right] \pmod{P(x)} = \sum_{k=0}^{m-1} x^k \left(\sum_{i+j=k} A_i B_j \right) + \sum_{l=m}^{2m-2} [x^l \pmod{P(x)}] \left(\sum_{u+v=l} A_u B_v \right), \end{aligned} \quad (2.4)$$

Zakładając, że $A(x) = B(x)$, można wyznaczyć wielomian, który jest kwadratem $A(x)$

$$\begin{aligned} D(x) &= (A(x))^2 \pmod{P(x)} \equiv D_0 + D_1x + \dots + D_{m-1}x^{m-1} = \\ &= \left[\sum_{i=0}^{m-1} A_i^2 x^{2i} \right] \pmod{P(x)} = \sum_{j=0}^{\hat{m}} A_j^2 x^{2j} + \sum_{k=\hat{m}}^{m-1} A_k^2 [x^{2k} \pmod{P(x)}], \end{aligned} \quad (2.5)$$

gdzie

$$\hat{m} = \left\lceil \frac{m}{2} \right\rceil - \lceil x \rceil \text{ oznacza najmniejszą liczbę całkowitą } \geq x.$$

Za pomocą równań (2.4) i (2.5), w których występują wielomiany o współczynnikach $GF(q)$, można wyznaczyć równanie, opisujące układ do realizacji operacji podnoszenia do dowolnej potęgi za pomocą kombinacji operacji podnoszenia do kwadratu i mnożenia.

Aby obliczyć odwrotność elementu $A(x)$ należy wyznaczyć wielomian

$$I(x) = I_0 + I_1 x + \dots + I_{m-1} x^{m-1} \equiv (1 / A(x)) \pmod{P(x)}, \quad (2.6)$$

taki, że

$$A(x)I(x) \equiv 1 \pmod{P(x)}, \quad (2.7)$$

skąd po pomnożeniu i pogrupowaniu wyrazów względem kolejnych potęg x oraz współczynników występujących w $I(x)$ otrzyma się układ równań liniowych postaci

$$\begin{array}{ccccccc} f_{0,0}I_0 & + & f_{0,1}I_1 & + & \dots & + & f_{0,m-1}I_{m-1} & = & 1 \\ f_{1,0}I_0 & + & f_{1,1}I_1 & + & \dots & + & f_{1,m-1}I_{m-1} & = & 0 \\ \vdots & & & & \vdots & & \vdots & & \\ f_{m-1,0} & + & f_{m-1,1}I_1 & + & \dots & + & f_{m-1,m-1}I_{m-1} & = & 0 \end{array}, \quad (2.8)$$

gdzie

$$f_{i,j} = \sum_{\substack{\text{dla niektórych } k \\ k \in \{0,1,\dots,m-1\}}} A_k.$$

Wartości współczynników $f_{i,j}$ są zależne od postaci wielomianu $P(x)$. Powyższy układ równań można rozwiązać np. metodą wyznacznikową.

Odwrotność elementu $A(x)$ można też określić na podstawie równania

$$I(x) \equiv A(x)^{q^m-2} \pmod{P(x)} \quad (2.9)$$

W przypadku, gdy $q = 2^k$ inwersję łatwo wyznaczyć korzystając z innej zależności. Ponieważ każdą n -bitową $n = k \cdot m$, liczbę binarną postaci 1111 .. 11 można przedstawić jako sumę $\sum_{i=0}^{n-1} 2^i$, czyli $2^n - 2 = \sum_{i=1}^{n-1} 2^i$. Pozwala to na przekształcenie wzoru (2.9) do postaci

$$(A(x))^{-1} = (A(x))^{2^n-2} \pmod{P(x)} = (A(x))^2 (A(x))^{2^2} \dots (A(x))^{2^{n-1}} \pmod{P(x)} = \prod_{i=1}^{n-1} A(x)^{2^i} \pmod{P(x)}, \quad (2.10)$$

gdzie

$$n = k \cdot m.$$

Warto w tym miejscu zauważyć, że operacje dodawania i mnożenia we wzorach (2.4)-(2.10) wykonuje się w ciele $GF(q)$.

PRZYKŁAD 2.1 Przykład dotyczy działań w ciele $GF(2^6) = GF(64)$, utworzonym za pomocą wielomianu nieredukowalnego $P(x) = x^6 + x^3 + 1$ nad $GF(2)$, który nie jest wielomianem pierwotnym. Dwa dowolne elementy tego ciała są wielomianami nad $GF(2)$ stopnia co najwyżej 5:

$$\begin{aligned} A(x) &= A_0 + A_1x + A_2x^2 + A_3x^3 + A_4x^4 + A_5x^5, \\ B(x) &= B_0 + B_1x + B_2x^2 + B_3x^3 + B_4x^4 + B_5x^5. \end{aligned} \quad (2.11)$$

Stosując przedstawione poprzednio wzory na iloczyn (2.4), kwadrat (2.5) i inwersję (2.7) otrzyma się

$$\begin{aligned} A(x)B(x) \pmod{x^6 + x^3 + 1} &= C(x) \equiv C_0 + C_1x + C_2x^2 + C_3x^3 + C_4x^4 + C_5x^5, \\ (A(x))^2 \pmod{x^6 + x^3 + 1} &= D(x) \equiv D_0 + D_1x + D_2x^2 + D_3x^3 + D_4x^4 + D_5x^5, \\ (A(x))^{-1} \pmod{x^6 + x^3 + 1} &= I(x) \equiv I_0 + I_1x + I_2x^2 + I_3x^3 + I_4x^4 + I_5x^5, \end{aligned} \quad (2.12)$$

czyli

$$\begin{aligned} C(x) \equiv A(x)B(x) \pmod{x^6 + x^3 + 1} &= A_0B_0 + (A_1B_0 + A_0B_1)x + (A_2B_0 + A_1B_1 + A_0B_2)x^2 + (A_3B_0 + \\ &+ A_2B_1 + A_1B_2 + A_0B_3)x^3 + (A_4B_0 + A_3B_1 + A_2B_2 + A_1B_3 + A_0B_4)x^4 + (A_5B_0 + A_4B_1 + A_3B_2 + \\ &+ A_2B_3 + A_1B_4 + A_0B_5)x^5 + (A_5B_1 + A_4B_2 + A_3B_3 + A_2B_4 + A_1B_5)x^6 + (A_5B_2 + A_4B_3 + A_3B_4 + \\ &+ A_2B_5)x^7 + (A_5B_3 + A_4B_4 + A_3B_5)x^8 + (A_5B_4 + A_4B_5)x^9 + A_5B_5x^{10} \pmod{x^6 + x^3 + 1}. \end{aligned}$$

Redukując wielomian $C(x)$ modulo $x^6 + x^3 + 1$ otrzyma się

$$\begin{aligned} A(x)B(x) &= A_0B_0 + A_5B_1 + A_4B_2 + A_3B_3 + A_2B_4 + A_5B_4 + A_1B_5 + A_4B_5 + (A_1B_0 + A_0B_1 + A_5B_2 + \\ &+ A_4B_3 + A_3B_4 + A_2B_5 + A_5B_5)x + (A_2B_0 + A_1B_1 + A_0B_2 + A_5B_3 + A_4B_4 + A_3B_5)x^2 + (A_3B_0 + \\ &+ A_2B_1 + A_5B_1 + A_1B_2 + A_4B_2 + A_0B_3 + A_3B_3 + A_2B_4 + A_1B_5)x^3 + (A_4B_0 + A_3B_1 + A_2B_2 + A_5B_2 + \\ &+ A_1B_3 + A_4B_3 + A_0B_4 + A_3B_4 + A_2B_5)x^4 + (A_5B_0 + A_4B_1 + A_3B_2 + A_2B_3 + A_5B_3 + A_1B_4 + A_4B_4 + \\ &+ A_0B_5 + A_3B_5)x^5. \end{aligned}$$

Ostatecznie składniki iloczynu dwóch elementów mają postać

$$\begin{aligned} C_0 &= A_0B_0 + A_5B_1 + A_4B_2 + A_3B_3 + A_2B_4 + A_5B_4 + A_1B_5 + A_4B_5, \\ C_1 &= A_1B_0 + A_0B_1 + A_5B_2 + A_4B_3 + A_3B_4 + A_2B_5 + A_5B_5, \\ C_2 &= A_2B_0 + A_1B_1 + A_0B_2 + A_5B_3 + A_4B_4 + A_3B_5, \\ C_3 &= A_3B_0 + A_2B_1 + A_5B_1 + A_1B_2 + A_4B_2 + A_0B_3 + A_3B_3 + A_2B_4 + A_1B_5, \\ C_4 &= A_4B_0 + A_3B_1 + A_2B_2 + A_5B_2 + A_1B_3 + A_4B_3 + A_0B_4 + A_3B_4 + A_2B_5, \\ C_5 &= A_5B_0 + A_4B_1 + A_3B_2 + A_2B_3 + A_5B_3 + A_1B_4 + A_4B_4 + A_0B_5 + A_3B_5. \end{aligned} \quad (2.13)$$

Aby obliczyć kwadrat elementu $A(x)$, we wzorach (2.13) należy podstawić, że $B_i = A_i$. W ciele $GF(q)$ charakterystyki 2 zachodzi $A_i^q = A_i$, a ponieważ $q = 2$, wobec tego

$$\begin{aligned}
D_0 &= A_0 + A_3, \\
D_1 &= A_5, \\
D_2 &= A_1 + A_4, \\
D_3 &= A_3, \\
D_4 &= A_2, \\
D_5 &= A_4,
\end{aligned} \tag{2.14}$$

gdzie $D_0, D_1, D_2, D_3, D_4, D_5$ - współczynniki wielomianu $D(x)$ występującego we wzorach (2.12).

Aby obliczyć $A(x)^{-1}$ należy doprowadzić do równań postaci (2.8). W tym celu należy obliczyć iloczyn $A(x)I(x) = 1 \pmod{x^6 + x^3 + 1}$ wykorzystując zależności (2.13)

$$\begin{aligned}
A(x)I(x) \pmod{x^6 + x^3 + 1} &= A_0I_0 + A_5I_1 + A_4I_2 + A_3I_3 + A_2I_4 + A_5I_4 + A_1I_5 + A_4I_5 + (A_1I_0 + \\
&+ A_0I_1 + A_5I_2 + A_4I_3 + A_3I_4 + A_2I_5 + A_5I_5)x + (A_2I_0 + A_1I_1 + A_0I_2 + A_5I_3 + A_4I_4 + A_3I_5)x^2 + (A_3I_0 \\
&+ A_2I_1 + A_5I_1 + A_1I_2 + A_4I_2 + A_0I_3 + A_3I_3 + A_2I_4 + A_1I_5)x^3 + (A_4I_0 + A_3I_1 + A_2I_2 + A_5I_2 + A_1I_3 + \\
&+ A_4I_3 + A_0I_4 + A_3I_4 + A_2I_5)x^4 + (A_5I_0 + A_4I_1 + A_3I_2 + A_2I_3 + A_5I_3 + A_1I_4 + A_4I_4 + A_0I_5 + A_3I_5)x^5 \\
&= 1
\end{aligned}$$

Ostatecznie obliczenie inwersji sprowadza się do odwrócenia macierzy symbolicznej układu równań (2.15)

$$\begin{bmatrix} A_0 & A_5 & A_4 & A_3 & A_2 + A_5 & A_1 + A_4 \\ A_1 & A_0 & A_5 & A_4 & A_3 & A_2 + A_5 \\ A_2 & A_1 & A_0 & A_5 & A_4 & A_3 \\ A_3 & A_2 + A_5 & A_1 + A_4 & A_0 + A_3 & A_2 & A_1 \\ A_4 & A_3 & A_2 + A_5 & A_1 + A_4 & A_0 + A_3 & A_2 \\ A_5 & A_4 & A_3 & A_2 + A_5 & A_1 + A_4 & A_0 + A_3 \end{bmatrix} \begin{bmatrix} I_0 \\ I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{2.15}$$

który rozwiązując metodą wyznacznikową¹ (obliczenia upraszcza fakt, że warunkiem istnienia rozwiązania jest aby $DET \neq 0$ czyli dla arytmetyki w $GF(2)$ $DET = 1$, gdzie DET - wyznacznik główny) otrzymuje się zależność (2.16) na inwersję.

$$\begin{aligned}
I_0 &= A_0 + A_0A_1 + A_3 + A_2A_3 + A_0A_4 + A_0A_1A_4 + A_2A_4 + A_0A_1A_2A_4 + A_2A_3A_4 + A_0A_2A_3A_4 + \\
&+ A_1A_2A_3A_4 + A_1A_5 + A_0A_1A_2A_5 + A_3A_5 + A_1A_3A_5 + A_0A_1A_3A_5 + A_2A_3A_5 + A_1A_2A_3A_5 + A_4A_5 \\
&+ A_0A_4A_5 + A_0A_1A_4A_5 + A_0A_2A_4A_5 + A_0A_1A_2A_4A_5 + A_0A_3A_4A_5 + A_1A_3A_4A_5 + A_2A_3A_4A_5 \\
&+ A_1A_2A_3A_4A_5,
\end{aligned}$$

¹ obliczenia te można łatwo wykonać za pomocą jednego z programowych pakietów specjalistycznych np. MATHEMATICA lub MAPLE.

$$I_1 = A_2 + A_1A_2 + A_1A_3 + A_1A_2A_3 + A_0A_1A_2A_3 + A_0A_4 + A_2A_4 + A_0A_2A_4 + A_1A_2A_4 + A_0A_1A_2A_4 + \\ + A_3A_4 + A_0A_2A_3A_4 + A_1A_2A_3A_4 + A_0A_1A_2A_3A_4 + A_0A_5 + A_1A_5 + A_3A_5 + A_0A_3A_5 + A_1A_2A_3A_5 \\ + A_4A_5 + A_1A_4A_5 + A_0A_2A_4A_5 + A_2A_3A_4A_5 ,$$

$$I_2 = A_1 + A_0A_1 + A_0A_2 + A_0A_1A_2 + A_1A_3 + A_0A_1A_3 + A_2A_3 + A_0A_1A_2A_3 + A_0A_4 + A_2A_4 + \\ + A_0A_1A_2A_4 + A_3A_4 + A_0A_3A_4 + A_1A_2A_3A_4 + A_0A_5 + A_0A_1A_2A_5 + A_0A_1A_3A_5 + A_1A_2A_3A_5 + \\ + A_0A_1A_2A_3A_5 + A_4A_5 + A_0A_1A_4A_5 + A_2A_4A_5 ,$$

$$I_3 = A_0A_1 + A_0A_2 + A_1A_2 + A_3 + A_1A_3 + A_0A_1A_2A_3 + A_0A_4 + A_0A_1A_4 + A_2A_4 + A_3A_4 + A_1A_3A_4 + \\ + A_2A_3A_4 + A_0A_2A_3A_4 + A_1A_2A_3A_4 + A_0A_5 + A_1A_5 + A_0A_2A_5 + A_1A_3A_5 + A_0A_1A_3A_5 + \\ + A_1A_2A_3A_5 + A_1A_3A_4A_5 + A_2A_3A_4A_5 + A_1A_2A_3A_4A_5 ,$$

$$I_4 = A_0A_1 + A_2 + A_0A_2 + A_1A_3 + A_2A_3 + A_0A_2A_3 + A_1A_2A_3 + A_0A_1A_2A_3 + A_0A_4 + A_0A_2A_4 + \\ + A_0A_1A_2A_4 + A_0A_2A_3A_4 + A_1A_2A_3A_4 + A_0A_1A_2A_3A_4 + A_5 + A_1A_5 + A_0A_1A_5 + A_0A_1A_2A_5 + \\ + A_0A_1A_3A_5 + A_1A_2A_3A_5 + A_4A_5 + A_1A_4A_5 + A_0A_1A_4A_5 + A_0A_2A_4A_5 + A_0A_3A_4A_5 + A_1A_3A_4A_5 \\ + A_0A_1A_3A_4A_5 ,$$

$$I_5 = A_1 + A_0A_2 + A_1A_2 + A_0A_1A_2 + A_0A_1A_2A_3 + A_4 + A_0A_4 + A_0A_1A_2A_4 + A_3A_4 + A_0A_3A_4 + \\ + A_0A_2A_3A_4 + A_1A_5 + A_1A_2A_5 + A_0A_1A_2A_5 + A_3A_5 + A_0A_1A_3A_5 + A_1A_2A_3A_5 + A_0A_1A_2A_3A_5 + \\ + A_0A_2A_4A_5 + A_3A_4A_5 + A_0A_3A_4A_5 + A_1A_3A_4A_5 + A_2A_3A_4A_5 + A_0A_2A_3A_4A_5 .$$

Wszystkie działania na wielomianach (2.11) wykonuje się z zastosowaniem operacji w $GF(2)$.

PRZYKŁAD 2.2 Dane jest ciało $GF(4)$ oraz wielomian nieredukowalny $P(x) = x^3 + 2$ o współczynnikach z tego ciała [Koś97:1][Koś98:1]. Dodawanie i mnożenie w $GF(4)$ wykonuje się zgodnie z tabliczkami:

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

·	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Można więc skonstruować ciało $GF(4^3) = GF(64)$. Będzie to ciało izomorficzne z ciałem z przykładu 2.1. Teraz elementami ciała $GF(64)$ będą wielomiany stopnia ≤ 2 nad $GF(4)$ $A(x) = A_0 + A_1x + A_2x^2$, $B(x) = B_0 + B_1x + B_2x^2$.

Podobnie jak w poprzednim przykładzie, podstawowe zależności dla tego ciała są następujące:

$$\begin{aligned}
A(x)B(x)(\bmod x^3 + 2) &= C(x) \equiv C_0 + C_1x + C_2x^2, \\
(A(x))^2 (\bmod x^3 + 2) &= D(x) \equiv D_0 + D_1x + D_2x^2, \\
(A(x))^{-1} (\bmod x^3 + 2) &= I(x) \equiv I_0 + I_1x + I_2x^2,
\end{aligned} \tag{2.17}$$

$$\begin{aligned}
C(x) = A(x)B(x) (\bmod x^3 + 2) &= A_0B_0 + (A_1B_0 + A_0B_1)x + (A_2B_0 + A_1B_1 + A_0B_2)x^2 + \\
&+ (A_2B_1 + A_1B_2)x^3 + A_2B_2x^4 (\bmod x^3 + 2).
\end{aligned}$$

Po zredukowaniu $(\bmod x^3 + 2)$ tego wielomianu otrzyma się

$$A(x)B(x) (\bmod x^3 + 2) = A_0B_0 + 2(A_2B_1 + A_1B_2) + (A_1B_0 + A_0B_1 + 2A_2B_2)x + (A_2B_0 + A_1B_1 + A_0B_2)x^2$$

wobec czego

$$\begin{aligned}
C_0 &= A_0B_0 + 2(A_2B_1 + A_1B_2), \\
C_1 &= A_1B_0 + A_0B_1 + 2A_2B_2, \\
C_2 &= A_2B_0 + A_1B_1 + A_0B_2.
\end{aligned} \tag{2.18}$$

W podobny sposób otrzyma się

$$\begin{aligned}
D_0 &= A_0^2, \\
D_1 &= 2A_2^2, \\
D_2 &= A_1^2.
\end{aligned} \tag{2.19}$$

Aby wyznaczyć wzory do obliczania inwersji wykorzystana będzie zależność (2.10), która w przypadku $GF(4^3)$ przyjmie postać

$$(A(x))^{-1}(\bmod x^3 + 2) = (A(x))^{62}(\bmod x^3 + 2) = (A(x))^2(A(x))^4(A(x))^8(A(x))^{16}(A(x))^{32}(\bmod x^3 + 2). \tag{2.20}$$

Podstawiając dane do wzorów na kwadrat

$$\begin{aligned}
D(x) &\equiv (A(x))^2(\bmod x^3 + 2) = A_0^2 + 2A_2^2x + A_1^2x^2, \\
(D(x))^2(\bmod x^3 + 2) &\equiv (A(x))^4(\bmod x^3 + 2) = A_0 + 2A_1x + 3A_2x^2, \\
((D(x))^2)^2(\bmod x^3 + 2) &\equiv (A(x))^8(\bmod x^3 + 2) = A_0^2 + 3A_2^2x + 3A_1^2x^2, \\
(((D(x))^2)^2)^2(\bmod x^3 + 2) &\equiv (A(x))^{16}(\bmod x^3 + 2) = A_0 + 3A_1x + 2A_2x^2, \\
((((D(x))^2)^2)^2)^2(\bmod x^3 + 2) &\equiv (A(x))^{32}(\bmod x^3 + 2) = A_0^2 + A_2^2x + 2A_1^2x^2.
\end{aligned} \tag{2.21}$$

Przyjmując teraz oznaczenia iloczynów częściowych jako

$$\begin{aligned}
DI1(x) &= (A(x))^2(A(x))^4(\bmod x^3 + 2) \equiv DI1_0 + DI1_1x + DI1_2x^2, \\
DI2(x) &= DI1(x)(A(x))^8(\bmod x^3 + 2) \equiv DI2_0 + DI2_1x + DI2_2x^2, \\
DI3(x) &= DI2(x)(A(x))^{16}(\bmod x^3 + 2) \equiv DI3_0 + DI3_1x + DI3_2x^2, \\
DI4(x) &\equiv I(x) = DI3(x)(A(x))^{32}(\bmod x^3 + 2) \equiv DI4_1 + DI4_1x + DI4_2x^2,
\end{aligned} \tag{2.22}$$

oraz podstawiając do wzorów na iloczyn dwóch elementów², otrzymuje się zależności

$$\begin{aligned}
 DI1(x) &= A_0^2 A_0 + 2(A_1^2 2A_1 + 2A_2^2 3A_2) + (2A_2^2 A_0 + A_0^2 2A_1 + 2A_1^2 3A_2)x + \\
 &\quad + (A_1^2 A_0 + 2A_2^2 2A_1 + A_0^2 3A_2)x^2, \\
 DI2(x) &= DI1_0 A_0^2 + 2(DI1_2 3A_2^2 + DI1_1 3A_1^2) + (DI1_1 A_0^2 + DI1_0 3A_2^2 + 2DI1_2 3A_1^2)x + \\
 &\quad + (DI1_2 A_0^2 + DI1_1 3A_2^2 + DI1_0 3A_1^2)x^2, \\
 DI3(x) &= DI2_0 A_0 + 2(DI2_2 3A_1 + DI2_1 2A_2) + (DI2_1 A_0 + DI2_0 3A_1 + 2DI2_2 2A_2)x + \\
 &\quad + (DI2_2 A_0 + DI2_1 3A_1 + DI2_0 2A_2)x^2, \\
 DI4(x) &= DI1_0 A_0^2 + 2(DI1_2 A_2^2 + DI1_1 2A_1^2) + (DI1_1 A_0^2 + DI1_0 A_2^2 + 2DI1_2 2A_1^2)x + \\
 &\quad + (DI1_2 A_0^2 + DI1_1 A_2^2 + DI1_0 2A_1^2)x^2,
 \end{aligned}$$

co po pomnożeniu i zredukowaniu daje

$$\begin{aligned}
 I_0 &\equiv DI4_0 = A_0^2 + 3A_0 A_1^2 A_2^2, \\
 I_1 &\equiv DI4_1 = 3A_2^2 + A_0^2 A_1^2 A_2, \\
 I_2 &\equiv DI4_2 = 3A_1^2 + 3A_0^2 A_1 A_2^2.
 \end{aligned} \tag{2.23}$$

PRZYKŁAD 2.3 Izomorficzne ciało $GF(64)$ można też utworzyć za pomocą wielomianu nieredukowalnego $P(x) = x^2 + x + 1$ nad $GF(8)$ o następujących tabliczkach dodawania i mnożenia

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	3	2	5	4	7	6
2	2	3	0	1	6	7	4	5
3	3	2	1	0	7	6	5	4
4	4	5	6	7	0	1	2	3
5	5	4	7	6	1	0	3	2
6	6	7	4	5	2	3	0	1
7	7	6	5	4	3	2	1	0

·	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	3	1	7	5
3	0	3	6	5	7	4	1	2
4	0	4	3	7	6	2	5	1
5	0	5	1	4	2	7	3	6
6	0	6	7	1	5	3	2	4
7	0	7	5	2	1	6	4	3

Elementami $GF(64)$ będą więc wielomiany stopnia ≤ 1 nad $GF(8)$ $A(x) = A_0 + A_1 x$, $B(x) = B_0 + B_1 x$. Rozważane zależności mają dla tego ciała postać:

$$\begin{aligned}
 A(x)B(x)(\text{mod } x^2 + x + 1) &= C(x) \equiv C_0 + C_1 x, \\
 (A(x))^2(\text{mod } x^2 + x + 1) &= D(x) \equiv D_0 + D_1 x, \\
 (A(x))^{-1}(\text{mod } x^2 + x + 1) &= I(x) \equiv I_0 + I_1 x,
 \end{aligned} \tag{2.24}$$

co daje zależność

$$C(x) \equiv A(x)B(x)(\text{mod } x^2 + x + 1) = A_0 B_0 + (A_1 B_0 + A_0 B_1)x + (A_2 B_0 + A_1 B_1 + A_0 B_2)x^2,$$

zaś po zredukowaniu

² redukcję wyrażeń upraszcza fakt, że w przypadku potęgowania dowolnego niezerowego elementu A w $GF(4)$ zachodzi $A^k = A^{k \bmod 3}$

$$A(x)B(x) \pmod{x^2 + x + 1} = A_0B_0 + A_1B_1 + (A_0B_1 + A_1B_1 + A_1B_0)x,$$

czyli

$$\begin{aligned} C_0 &= A_0B_0 + A_1B_1, \\ C_1 &= A_0B_1 + A_1B_1 + A_1B_0, \end{aligned} \quad (2.25)$$

oraz wzory na kwadrat

$$\begin{aligned} D_0 &= A_0^2 + A_1^2, \\ D_1 &= A_1^2, \end{aligned} \quad (2.26)$$

Podobnie jak w przykładzie 2.2

$$I(x) = A(x)^{62} \pmod{x^2 + x + 1} = (A(x)^2)(A(x)^4)(A(x)^8)(A(x)^{16})(A(x)^{32}) \pmod{x^2 + x + 1}. \quad (2.27)$$

Zaś z wzoru na kwadrat i iloczyn elementu w tym ciele wiadomo także, iż

$$(A(x))^K \pmod{P(x)} = A_0^K + A_1^K x, \text{ dla } K = 2^{2^i}, i = 1, 2, \dots, \quad (2.28)$$

oraz

$$(A(x))^K \pmod{P(x)} = (A_0^K + A_1^K) + A_1^K x, \text{ dla } K = 2^{2^{k-1}}, i = 1, 2, \dots, \quad (2.29)$$

wobec tego

$$\begin{aligned} (A(x))^2 &= (A_0^2 + A_1^2) + A_1^2 x, \\ (A(x))^4 &= A_0^4 + A_1^4 x, \\ (A(x))^8 &= A_0^8 + A_1^8 x, \\ (A(x))^{16} &= A_0^{16} + A_1^{16} x, \\ (A(x))^{32} &= A_0^{32} + A_1^{32} x. \end{aligned} \quad (2.30)$$

Oznaczając teraz kolejne iloczyny częściowe występujące we wzorze (2.27) jako

$$\begin{aligned} DI1(x) &\equiv (A(x))^2 (A(x))^4 \pmod{x^2 + x + 1} = DI1_0 + DI1_1 x = (A_0^6 + A_1^6) + A_1^6 x, \\ DI2(x) &\equiv DI1(x)(A(x))^8 \pmod{x^2 + x + 1} = DI2_0 + DI2_1 x, \\ DI3(x) &\equiv DI2(x)(A(x))^{16} \pmod{x^2 + x + 1} = DI3_0 + DI3_1 x, \\ DI4(x) &\equiv DI3(x)(A(x))^{32} \pmod{x^2 + x + 1} = DI3_0 + DI3_1 x, \end{aligned} \quad (2.31)$$

otrzymuje się zależności, których postać można zapisać ogólnie

$$\begin{aligned} DIi(x) &= (DI[i-1]_0 A_0^{2^{i+1}} + DI[i-1]_1 A_1^{2^{i+1}}) + (DI[C-1]_0 A_1^{2^{i-1}} + \\ &\quad + DI[i-1]_1 A_1^{2^{i-1}} + DI[i-1]_1 A_0^{2^{i+1}}) x, \end{aligned} \quad (2.32)$$

gdzie

i – numer kolejnego iloczynu częściowego, $i = 2, 3, 4$.

Ponieważ dla każdego dowolnego elementu $C \neq 0$ ciała $GF(8)$ o zdefiniowanej powyżej tabliczce mnożenia zachodzi $C^K = C^{K \bmod 7}$, $C^7 = 1$, to obliczając kolejne iloczyny częściowe ostatecznie można otrzymać wzory na inwersję elementu w tym ciele

$$\begin{aligned} I_0 &= A_0^6 + A_0^5 A_1 + A_0^4 A_1^2 + A_0^2 A_1^4 + A_0 A_1^5 + A_1^6 + A_0^7 A_1^6 + A_0^6 A_1^7, \\ I_1 &= A_0^5 A_1 + A_0^3 A_1^3 + A_0^2 A_1^4 + A_1^6 + A_0^6 A_1^7. \end{aligned} \quad (2.33)$$

Należy dla ścisłości dodać, że w tym przypadku działania we wzorach (2.24)-(2.33) oznaczają operacje dodawania i mnożenia w $GF(8)$.

2.2 Permutacje

Występujące w literaturze pojęcia związane z analizą i sposobem prezentowania permutacji są w wielu przypadkach niejednolite, wobec tego autor niniejszej rozprawy, na podstawie [Kli93][Bir83][Den93][Koś94:1][Top95] określił podstawowy zakres zagadnień, definicji i twierdzeń związanych z tą tematyką.

DEFINICJA 2.1 Niech X będzie zbiorem o n elementach $\{x_0, x_1, \dots, x_{n-1}\}$. Funkcja $g: X \rightarrow X$ nazywana jest **permutacją** zbioru X jeżeli g jest odwzorowaniem wzajemnie jednoznacznym zbioru X na X .

Wobec wzajemnej jednoznaczności odwzorowania permutacyjnego nie ma potrzeby rozważać permutacji danego zbioru X o n elementach, bo każdą taką permutację $g: X \rightarrow X$ można zapisać jako

$$g(x_i) = x_{g(i)} \quad (2.34)$$

gdzie

i - indeks elementu zbioru X ,

co automatycznie można sprowadzić do permutacji

$$i \rightarrow g(i) \quad (2.35)$$

gdzie

i - indeks elementu zbioru ,

czyli permutacji zbioru $N = \{0, 1, 2, 3, \dots, n-1\}$ indeksów elementów zbioru X .

Na ogół jako N przyjmuje się zbiór początkowych liczb naturalnych tzn. $N = \{1, 2, 3, \dots, n\}$, czasem również przyjmuje się $N = \{0, 1, 2, 3, \dots, n-1\}$, co zastosowano w niniejszej rozprawie.

W pracy rozważa się również uproszczony wariant zbioru X , którego elementami są dodatnie liczby całkowite, czyli $\forall x_i \in X \ x_i = i$.

Gdy $g(x) = x \ \forall x \in N$ permutację nazywa się **identycznościową** i oznacza jako e .

TWIERDZENIE 2.1. Liczba wszystkich permutacji zbioru n elementowego jest równa $n!$.

TWIERDZENIE 2.2. Zbiór S wszystkich permutacji zbioru n elementowego wraz ze operacją mnożenia permutacji tworzy grupę, czyli system algebraiczny

$$\langle S, \circ \rangle \quad (2.36)$$

o własnościach

$$1^\circ \forall g, h \in S_n \ g \circ h \in S;$$

$$2^\circ \forall f, g, h \in S \quad (f \circ g) \circ h = f \circ (g \circ h); \quad (2.37)$$

$$3^\circ \forall g \in S \exists e \in S : g \circ e = e \circ g = g;$$

$$4^\circ \forall g \in S \exists g' \in S : g \circ g' = g' \circ g = e.$$

Grupę tę oznacza się jako S_n i nazywa grupą symetryczną stopnia n .

Pojedynczą permutację przedstawia się często w publikacjach jako dwuwiersz o postaci

$$g = \begin{pmatrix} 0 & 1 & \dots & n-1 \\ g(0) & g(1) & \dots & g(n-1) \end{pmatrix}. \quad (2.38)$$

W każdym z wierszy umieszczonych jeden pod drugim, przedstawione są ciągi n różnych elementów zbioru N . W górnym umieszczone są elementy N w porządku rosnącym w drugim natomiast pod każdym elementem wiersza pierwszego obraz tego elementu uzyskany w wyniku działania danej permutacji g .

PRZYKŁAD 2.4 Dany jest zbiór $N = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$. Permutację g jego elementów można zapisać jako

$$g = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 3 & 5 & 2 & 6 & 4 & 12 & 10 & 1 & 9 & 8 & 0 & 11 & 13 & 14 & 7 & 15 \end{pmatrix}. \quad (2.39)$$

Często w publikacjach związanych z kryptografią spotkać można także uproszczony sposób takiego zapisu permutacji polegający na pominięciu górnego wiersza. Przedstawia się tylko wiersz dolny zakładając znaną uporządkowaną kolejność przedstawiania elementów odwzorowania (od najmniejszego do największego). W takim przypadku permutacja g wygląda następująco

$$g = 3 \ 5 \ 2 \ 6 \ 4 \ 12 \ 10 \ 1 \ 9 \ 8 \ 0 \ 11 \ 13 \ 14 \ 7 \ 15. \quad (2.40)$$

Powyżej przedstawione sposoby prezentacji permutacji są jednak dość kłopotliwe, gdy permutowany zbiór zawiera dużo elementów. Poza tym trudno na ich podstawie wyciągnąć wnioski dotyczące niektórych cech zastosowanego przekształcenia.

TWIERDZENIE 2.3. Niech g i h będą permutacjami tego samego zbioru N . **Iloczynem** (złożeniem, kompozycją) $g \circ h$ (lub gh), tych permutacji nazywamy permutację f zbioru N określoną za pomocą warunku

$$x^f = (x^g)^h \quad \forall x \in N \quad (2.41)$$

Mnożenie permutacji nie jest przemienne. Niezmiennikiem operacji mnożenia permutacji jest permutacja identycznościowa e .

PRZYKŁAD 2.5 Dany jest zbiór $N = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$ oraz permutacja g tego zbioru

$$g = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 3 & 5 & 2 & 6 & 4 & 12 & 10 & 1 & 9 & 8 & 0 & 11 & 13 & 14 & 7 & 15 \end{pmatrix}, \quad (2.42)$$

a także permutacja h

$$h = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 9 & 10 & 15 & 6 & 5 & 4 & 13 & 8 & 0 & 12 & 14 & 11 & 1 & 2 & 3 & 7 \end{pmatrix}. \quad (2.43)$$

Iloczyn $f = g \circ h$ tych permutacji jest permutacją o postaci

$$f = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 6 & 4 & 15 & 13 & 5 & 1 & 14 & 10 & 12 & 0 & 9 & 11 & 2 & 3 & 8 & 7 \end{pmatrix}, \quad (2.44)$$

zaś iloczyn $k = h \circ g$ permutacją

$$k = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 8 & 0 & 15 & 10 & 12 & 4 & 14 & 9 & 3 & 13 & 7 & 11 & 5 & 2 & 6 & 1 \end{pmatrix}, \quad (2.45)$$

przy czym wyraźnie widać, że $f \neq k$.

Permutacja odwrotna do g oznaczana jako g^{-1} ma postać

$$g^{-1} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 10 & 7 & 2 & 0 & 4 & 1 & 3 & 14 & 9 & 8 & 6 & 11 & 5 & 12 & 13 & 15 \end{pmatrix}. \quad (2.46)$$

DEFINICJA 2.2. Rzędem r elementu $g \in S_n$ nazywa się liczbę naturalną taką, że $g^r = e$ oraz $g^k \neq e$ dla każdego $k < r$.

DEFINICJA 2.3. Dla danego zbioru N o liczbie elementów n permutacja $t \in S_n$ nazywana jest **cyklem długości k** , jeżeli istnieją elementy $a_0, a_1, \dots, a_{k-1} \in N$ takie, że

$$t(a_0) = a_1, t(a_1) = a_2, \dots, t(a_{k-2}) = a_{k-1}, t(a_{k-1}) = a_0 \quad (2.47)$$

i

$$t(x) = x, \quad \forall x \in N : x \neq a_i, \quad i = 0, 1, \dots, k-1. \quad (2.48)$$

Zapisuje się to jako $t = (a_0, a_1, \dots, a_{k-1})$ lub $t = (a_1, a_2, \dots, a_{k-1}, a_0)$ albo $t = (a_2, a_3, \dots, a_0, a_1)$, co oznacza, że notacja dla cykli długości k może być realizowana na k różnych sposobów zależnie od punktu startowego prezentacji.

DEFINICJA 2.4. Niech $t = (a_0, a_1, \dots, a_{k-1})$ i $\tau = (b_0, b_1, \dots, b_{m-1})$ będą cyklami w S_n . Jeżeli $a_i \neq b_j$ dla wszystkich i, j , to t i τ nazywają się **cyklami rozłącznymi**.

TWIERDZENIE 2.4. Każda permutacja może być zapisana jako iloczyn rozłącznych cykli. Cykle występujące w iloczynie są niepowtarzalne w ramach tej permutacji. Taki zapis nazywa się rozkładem permutacji na cykle.

Rozkład na cykle pozwala na zapisanie w jednym wierszu wyników działania przekształcenia permutacyjnego g . Rozłączne cykle składowe rozdzielone są w tym zapisie nawiasami. Każdy cykl rozpoczyna się od najmniejszego elementu nie występującego w poprzednich cyklach. Wewnątrz każdego cyklu elementy występują sekwencyjnie w kolejności odwzorowań od najmniejszego do największego. Każdy element występuje w dokładnie jednym cyklu.

Oto rozkład na cykle permutacji z przykładu 2.1

$$g = (0, 3, 6, 10) (1, 5, 12, 13, 14, 7) (2) (4) (8, 9) (11) (15). \quad (2.49)$$

Działanie permutacji g można łatwo odczytać z jej rozkładu na cykle. Wystarczy dla każdego elementu $x \in N$ znaleźć cykl t_i , w którym on występuje a obrazem $g(x)$ jest element bezpośrednio po nim występujący. Jeżeli x jest elementem stojącym na końcu cyklu, to $g(x)$ jest pierwszym elementem tego cyklu.

Dopuszczalny jest także tzw. skrócony rozkład na cykle, w którym opuszczane są wszystkie cykle długości 1.

Mnożenie cykli rozłącznych, w odróżnieniu od mnożenia permutacji, jest przemienne, wobec tego porządek cykli rozłącznych, w jakim przedstawiamy permutacje jest dowolny. Dla większej czytelności przykładów w pracy przyjęto jednak zasadę jednoznacznego uporządkowania cykli, choć powszechnie nie występuje ona w zapisie permutacji.

Jeżeli permutacja $g \in S_n$ zapisana jest w postaci iloczynu rozłącznych cykli, to (zgodnie z definicją 2.2) jej rząd r jest najmniejszym wspólną wielokrotnością iloczynu długości jej cykli.

DEFINICJA 2.5. Cykl (a_i, a_j) długości dwa nazywa się **transpozycją**.

Każdy cykl długości k jest iloczynem $k-1$ rozłącznych transpozycji, co można zapisać jako

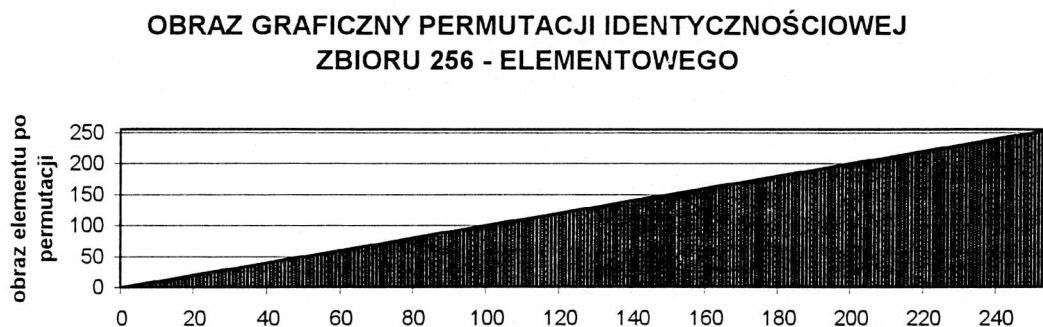
$$(a_0, a_1, \dots, a_{k-2}, a_{k-1}) = (a_0, a_{k-1})(a_0, a_{k-2}) \dots (a_0, a_2)(a_0, a_1). \quad (2.50)$$

W przykładach rozpatrywanych w niniejszej rozprawie wielokrotnie występują podzbiory zbioru S_n . Z punktu widzenia kryptografii istotne jest, aby przekształcenia permutacyjno - podstawieniowe użyte w procesach szyfrująco - deszyfrujących nie tworzyły grupy [Sch95].

W celu ułatwienia analizy permutacji zbiorów o dużej ilości elementów w pracy zastosowano także inny, odmienny od powszechnie przyjętych, sposób prezentacji permutacji - za pomocą **histogramów**. Metoda ta, poprzez zastosowanie grafiki dwuwymiarowej, pozwala na pełniejszą, intuicyjnie jednoznaczną identyfikację cech analizowanego przekształcenia, wyni-

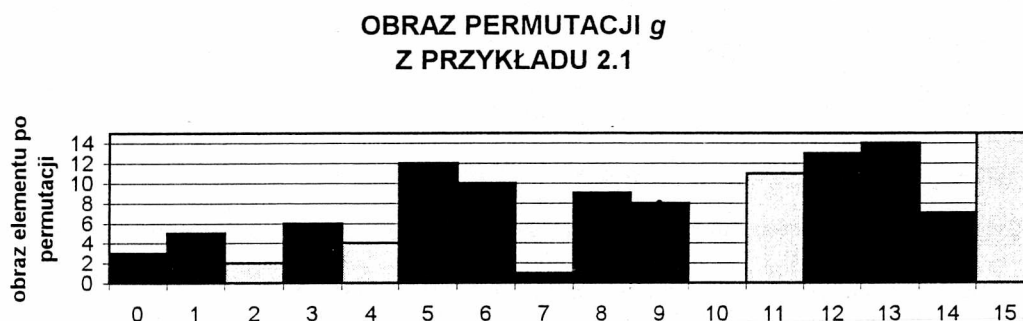
kającą z możliwości bezpośredniej obserwacji całego spektrum wartości zbioru permutowanego oraz ich obrazów w postaci kompozycji elementów graficznych.

Na rys .2.1. zaprezentowano przykład graficznego przedstawienia permutacji identycznościowej zbioru 256 - elementowego.



Rys.2.1 Histogram permutacji identycznościowej

Oto zaś histogram permutacji g z (2.38).



Rys.2.2 Histogram permutacji z przykładu 2.4

W takim sposobie prezentowania permutacji na osiach umieszczone są uporządkowane rosnąco elementy zbioru permutowanego N , zaś wysokości słupków nad osią odciętych określają wartości obrazów tych elementów w przekształceniu.

W przedstawionych powyżej histogramach łatwo zauważyć pewien rzucający się w oczy element - regularności rozmieszczenia elementów graficznych. W niniejszej pracy zwrócono uwagę przede wszystkim na te regularności, które mają związek z występowaniem zależności liniowych w przekształceniach permutacyjnych (takie regularności określane są w rozprawie mianem trendów liniowych), a także na inne mające wyraźnie symetryczne cechy graficzne. Kolorem jaśniejszym wyróżniono niezmienniki przekształcenia (elementy odwzorowane w permutacji same na siebie).

3. Metoda generowania par permutacji wzajemnie odwracalnych zbioru 2^m -elementowego z zastosowaniem przekształceń afinicznych przestrzeni liniowych na ciałach Galois charakterystyki 2.

W rozdziale niniejszym przedstawiono metodę generowania wzajemnie odwracalnych par permutacji zbioru q^m -elementowego (q - potęga liczby 2, m - liczba naturalna ≥ 1) z zastosowaniem przekształceń afinicznych m - wymiarowej przestrzeni liniowej nad $GF(q)$. Permutacje uzyskane za pomocą tej metody mają opinię „słabych” pod względem kryptograficznym, co jest spowodowane liniowym charakterem użytych w niej przekształceń. Wydaje się, że pełna znajomość metody pozwala jednak na właściwe określenie jej użyteczności w technikach kryptograficznych.

Najpierw rozpatrzono bardziej ogólny przypadek przekształceń afinicznych nad $GF(q)$, a następnie omówiono szczegółowo tematykę stosowania przekształceń afinicznych przestrzeni liniowej nad $GF(2)$.

3.1 Przekształcenia afiniczne przestrzeni liniowych nad $GF(q)$ charakterystyki 2

System algebraiczny

$$\langle V, +, \cdot \rangle \quad (3.1)$$

gdzie

$V - q^m$ - elementowy zbiór (q - potęga liczby 2, m - liczba naturalna ≥ 1)

$+, \cdot$ - operacje dodawania i mnożenia przez element $GF(q)$, zdefiniowane

w zbiorze V ,

jest przestrzenią liniową nad $GF(q)$, gdy spełnione są następujące warunki

1. System algebraiczny $\langle V, + \rangle$ jest addytywną grupą przemenną.
2. $\forall s \in GF(q) \quad \forall v \in V \quad s \cdot v = v \cdot s \in V$.

Niech elementami V będą m - wymiarowe wektory nad $GF(q)$

$$v = [v_0 \ v_1 \ \dots \ v_{m-1}], \quad w = [w_0 \ w_1 \ \dots \ w_{m-1}], \quad v_i, w_i \in GF(q). \quad (3.2)$$

Dodawanie zdefiniowane jest tutaj następująco

$$v + w = [v_0 + w_0 \quad v_1 + w_1 \quad \dots \quad v_{m-1} + w_{m-1}]. \quad (3.3)$$

Mnożenie przez skalar wykonuje się następująco:

$$s \cdot v = [s \cdot v_0 \quad s \cdot v_1 \quad \dots \quad s \cdot v_{m-1}]. \quad (3.4)$$

Przekształcenie afiniczne przestrzeni liniowej V na siebie

$$\pi : V \rightarrow V, \quad (3.5)$$

określone jest funkcją

$$\pi(x) = x \cdot S + c = y, \quad (3.6)$$

gdzie

$$S = \begin{bmatrix} s_{0,0} & s_{0,1} & \dots & s_{0,m-1} \\ s_{1,0} & s_{1,1} & \dots & s_{1,m-1} \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ s_{m-1,0} & s_{m-1,1} & \dots & s_{m-1,m-1} \end{bmatrix}, \quad (3.7)$$

jest macierzą nieosobliwą $m \times m$ nad $GF(q)$, zaś x, y, c to elementy przestrzeni V (m - wymiarowe wektory, których składowe są elementami $GF(q)$, przy czym c jest wektorem stałym).

Wartości funkcji (3.6) wyznaczają permutację zbioru V stowarzyszoną z przekształceniem π

Przekształcenie afiniczne jest izomorfizmem, wobec czego istnieje przekształcenie do niego odwrotne określone funkcją

$$\pi^{-1}(y) = (y + c) \cdot S^{-1} = x. \quad (3.8)$$

Liczba wszystkich permutacji zbioru V , możliwych do wygenerowania za pomocą tych przekształceń wynosi

$$N_T = q^m \cdot |GL(m, q)|, \quad (3.9)$$

gdzie $GL(m, q)$ oznacza grupę liniową, czyli grupę wszystkich nieosobliwych macierzy $m \times m$ nad $GF(q)$ z mnożeniem macierzy jako operacją grupową. Po rozwinięciu tego elementu i uproszczeniu wzoru otrzymuje się

$$N_T = q^{m+(m-1)\cdot m/2} \cdot \prod_{i=1}^m (q^i - 1). \quad (3.10)$$

Zależność liczby przekształceń afinicznych od wartości wymiaru przestrzeni m dla $q = 2$ (przypadek istotny dla realizacji sprzętowych w technice cyfrowej) pokazano w tabeli zamieszczonej w dodatku na str. 135.

Przykład ilustrujący projektowanie układu generującego permutacje w oparciu o wyżej opisane zależności zaprezentowano niżej.

PRZYKŁAD 3.1 Oznaczając elementy ciała $GF(8)$, utworzonego za pomocą wielomianu pierwotnego $P(x) = x^3 + x + 1$ jako $0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6$, można zdefiniować tablice dodawania i mnożenia w tym ciele

+	0	1	α	α^2	α^3	α^4	α^5	α^6
0	0	1	α	α^2	α^3	α^4	α^5	α^6
1	1	0	α^3	α^6	α	α^5	α^4	α^2
α	α	α^3	0	α^4	1	α^2	α^6	α^5
α^2	α^2	α^6	α^4	0	α^5	α	α^3	1
α^3	α^3	α	1	α^5	0	α^6	α^2	α^4
α^4	α^4	α^5	α^2	α	α^6	0	1	α^3
α^5	α^5	α^4	α^6	α^3	α^2	1	0	α
α^6	α^6	α^2	α^5	1	α^4	α^3	α	0

·	0	1	α	α^2	α^3	α^4	α^5	α^6
0	0	0	0	0	0	0	0	0
1	0	1	α	α^2	α^3	α^4	α^5	α^6
α	0	α	α^2	α^3	α^4	α^5	α^6	1
α^2	0	α^2	α^3	α^4	α^5	α^6	1	α
α^3	0	α^3	α^4	α^5	α^6	1	α	α^2
α^4	0	α^4	α^5	α^6	1	α	α^2	α^3
α^5	0	α^5	α^6	1	α	α^2	α^3	α^4
α^6	0	α^6	1	α	α^2	α^3	α^4	α^5

Dla prostoty rozważań zakłada się, że V jest taką przestrzenią liniową nad $GF(8)$, w której każdy element jest dwuwymiarowym wektorem o składowych z ciała $GF(8)$. Wobec tego

$$x = [x_0 \ x_1], \quad y = [y_0 \ y_1], \quad c = [c_0 \ c_1], \quad (3.11)$$

Niech dana będzie macierz nieosobliwa S o elementach w $GF(8)$ i postaci

$$S = \begin{bmatrix} 0 & 1 \\ \alpha & \alpha^2 \end{bmatrix}. \quad (3.12)$$

Łatwo stwierdzić, że macierz S jest macierzą nieosobliwą ponieważ $DET(S) = \alpha$, gdzie $DET(S)$ – wyznacznik macierzy S .

Macierz odwrotną do S wyznacza się z równania $S \cdot S^{-1} = E$, gdzie E oznacza macierz jednostkową, czyli

$$\begin{bmatrix} 0 & 1 \\ \alpha & \alpha^2 \end{bmatrix} \cdot \begin{bmatrix} s_{00} & s_{01} \\ s_{10} & s_{11} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \quad (3.13)$$

Oczywiście wszystkie operacje muszą być wykonywane zgodnie z regułami podanymi w tablicach dodawania i mnożenia dla ciała $GF(8)$. Ostatecznie wyznaczona macierz odwrotna ma postać

$$S^{-1} = \begin{bmatrix} \alpha & \alpha^6 \\ 1 & 0 \end{bmatrix}. \quad (3.14)$$

A oto postać przekształcenia afinicznego

$$\pi(x) = y = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}^T \begin{bmatrix} 0 & 1 \\ \alpha & \alpha^2 \end{bmatrix} + \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}^T, \quad (3.15)$$

gdzie

T oznacza operator transpozycji,

co po uproszczeniu można przedstawić jako

$$\pi(x) = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}^T = \begin{bmatrix} x_1 \alpha + c_0 \\ x_0 + x_1 \alpha^2 + c_1 \end{bmatrix}^T, \quad (3.16)$$

czyli

$$\begin{aligned} y_0 &= x_1 \alpha + c_0, \\ y_1 &= x_0 + x_1 \alpha^2 + c_1. \end{aligned} \quad (3.17)$$

Przekształcenie odwrotne do powyższego wyraża zależność

$$\pi^{-1}(y) = \left(\begin{bmatrix} y_0 \\ y_1 \end{bmatrix}^T + \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}^T \right) \begin{bmatrix} \alpha & \alpha^6 \\ 1 & 0 \end{bmatrix}, \quad (3.18)$$

co ostatecznie można przedstawić jako

$$\pi^{-1}(y) = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}^T = \begin{bmatrix} \alpha(y_0 + c_0) + y_1 + c_1 \\ \alpha^6(y_0 + c_0) \end{bmatrix}^T, \quad (3.19)$$

czyli

$$\begin{aligned} x_0 &= \alpha(y_0 + c_0) + y_1 + c_1, \\ x_1 &= \alpha^6(y_0 + c_0). \end{aligned} \quad (3.20)$$

Zakładając teraz, że $c = [\alpha^6 \ \alpha]$ i $x = [\alpha \ 1]$ i pamiętając o zgodności obliczeń z przytoczoną powyżej tabliczką dodawania i mnożenia w rozpatrywanym ciele $GF(8)$ charakterystyki 2, można obliczyć za pomocą wzoru (3.17), że $y = [\alpha^5 \ \alpha^2]$. Jednocześnie poprawność otrzymanego wyniku można sprawdzić podstawiając obliczoną wartość y do wzoru (3.20). W ten sposób otrzyma się założoną na początku obliczeń wartość x .

Za pomocą wzoru (3.20) można obliczyć całą permutację zbioru V , zawierającego 64 elementy.

Z punktu widzenia zastosowań praktycznych elementy zbioru V powinno się przedstawiać w postaci liczb binarnych.

Każdy element ciała $GF(8)$, skonstruowanego za pomocą wielomianu nieredukowalnego $P(x) = x^3 + x + 1$, można wyrazić za pomocą wielomianu stopnia ≤ 2 o współczynnikach z $GF(2)$. Wobec tego elementy $GF(8)$ mają postać wielomianów, zapisywanych za pomocą wektorów

$$\begin{array}{ll} 0 & \leftrightarrow [0 \ 0 \ 0] \\ 1 & \leftrightarrow [1 \ 0 \ 0] \\ \alpha & \leftrightarrow [0 \ 1 \ 0] \\ \alpha^2 & \leftrightarrow [0 \ 0 \ 1] \end{array} \quad \begin{array}{ll} \alpha^3 & \leftrightarrow [1 \ 1 \ 0] \\ \alpha^4 & \leftrightarrow [0 \ 1 \ 1] \\ \alpha^5 & \leftrightarrow [1 \ 1 \ 1] \\ \alpha^6 & \leftrightarrow [1 \ 0 \ 1] \end{array}$$

Mnożenie dwóch elementów ciała $GF(8)$ jest więc równoważne mnożeniu dwóch wielomianów modulo $x^3 + x + 1$

$$C(x) = A(x) B(x) \pmod{x^3 + x + 1}. \quad (3.21)$$

Na przykład

$$\begin{aligned} A(x) &= a_0 + a_1 x + a_2 x^2, \\ B(x) &= b_0 + b_1 x + b_2 x^2, \end{aligned} \quad (3.22)$$

to współczynniki wielomianu $C(x) = c_0 + c_1 x + c_2 x^2$ można obliczyć z następujących wzorów

$$\begin{aligned} c_0 &= a_0 b_0 + a_2 b_1 + a_1 b_2, \\ c_1 &= a_1 b_0 + a_0 b_1 + a_2 b_1 + a_1 b_2 + a_2 b_2, \\ c_2 &= a_2 b_0 + a_1 b_1 + a_0 b_2 + a_2 b_2. \end{aligned} \quad (3.23)$$

Zgodnie z powyżej określoną zasadą występujące w przykładzie wektory x, y, c można teraz przedstawić następująco

$$x = [x_{00} x_{01} x_{02} \ x_{10} x_{11} x_{12}], \quad y = [y_{00} y_{01} y_{02} \ y_{10} y_{11} y_{12}], \quad c = [c_{00} c_{01} c_{02} \ c_{10} c_{11} c_{12}],$$

gdzie wzory $x_{i,k}, y_{i,k}, c_{i,k}$ - elementy $GF(2)$.

$$\pi(x) = y = \begin{bmatrix} y_{00} y_{01} y_{02} \\ y_{10} y_{11} y_{12} \end{bmatrix}^T = \begin{bmatrix} x_{00} x_{01} x_{02} \\ x_{10} x_{11} x_{12} \end{bmatrix}^T \begin{bmatrix} 000 & 100 \\ 010 & 001 \end{bmatrix} + \begin{bmatrix} c_{00} c_{01} c_{02} \\ c_{10} c_{11} c_{12} \end{bmatrix}^T, \quad (3.24)$$

$$\pi^{-1}(y) = x = \begin{bmatrix} x_{00} & x_{01} & x_{02} \\ x_{10} & x_{11} & x_{12} \end{bmatrix}^T = \left(\begin{bmatrix} y_{00} & y_{01} & y_{02} \\ y_{10} & y_{11} & y_{12} \end{bmatrix}^T + \begin{bmatrix} c_{00} & c_{01} & c_{02} \\ c_{10} & c_{11} & c_{12} \end{bmatrix}^T \right) \begin{bmatrix} 010 & 101 \\ 100 & 000 \end{bmatrix}. \quad (3.25)$$

Biorąc pod uwagę układ równań (3.23) otrzymuje się

$$\begin{aligned} y_{00} &= x_{12} + c_{00}, \\ y_{01} &= x_{10} + x_{12} + c_{01}, \\ y_{02} &= x_{11} + c_{02}, \\ y_{10} &= x_{11} + x_{00} + c_{10}, \\ y_{11} &= x_{11} + x_{12} + x_{01} + c_{11}, \\ y_{12} &= x_{02} + x_{10} + x_{12} + c_{12}, \end{aligned} \quad (3.26)$$

oraz

$$\begin{aligned} x_{00} &= y_{02} + y_{10} + c_{02} + c_{10}, \\ x_{01} &= y_{00} + y_{02} + y_{11} + c_{00} + c_{02} + c_{11}, \\ x_{02} &= y_{01} + y_{12} + c_{01} + c_{12}, \\ x_{10} &= y_{00} + y_{01} + c_{00} + c_{01}, \\ x_{11} &= y_{02} + c_{02}, \\ x_{12} &= y_{00} + c_{00}. \end{aligned} \quad (3.27)$$

Operacje sumowania występujące we wzorach (3.26) i (3.25) mogą być zrealizowane za pomocą układów XOR.

Podstawiając teraz przyjęte w przykładzie wartości liczbowe do wzorów (3.26) można łatwo sprawdzić, że dla $x = [010 \ 100]$, $c = [101 \ 010]$ wartość $y = [111 \ 001]$. Zaś po podstawieniu otrzymanej wartości y do wzoru (3.27) otrzymamy założoną wartość x .

Zakładając, że wektory x, y, c reprezentują następujące liczby binarne

$$x_N = \sum_{i=0}^1 \sum_{j=0}^2 x_{ij} \cdot 2^{3i+j} \quad y_N = \sum_{i=0}^1 \sum_{j=0}^2 y_{ij} \cdot 2^{3i+j} \quad c_N = \sum_{i=0}^1 \sum_{j=0}^2 c_{ij} \cdot 2^{3i+j}, \quad (3.28)$$

można się przekonać, że dla założonej wyżej wartości $c_N = 21$ układ równań (3.26) może posłużyć do wygenerowania permutacji

(0 21 33 46 52 26 59) (1 29 3 13 31 19 17) (2 5 61 48 58 8 55) (4 53 18 25 35 62 40) (6 37 14 7 45 44 36) (9 63 32 38 22 57 16) (10 39 30 27 51 34 54) (11 47 60 56 24 43 28) (12 23 49 50 42 20 41) (15)

Rys. 3.1. Permutacja opisana równaniami (3.26)

Jednocześnie równania (3.27) generują permutację

(0 59 26 52 46 33 21) (1 17 19 31 13 3 29) (2 55 8 58 48 61 5) (4 40 62 35 25 18 53) (6 36 44 45 7 14 37) (9 16 57 22 38 32 63) (10 54 34 51 27 30 39) (11 28 43 24 56 60 47) (12 41 20 42 50 49 23) (15)

Rys. 3.2. Permutacja opisana równaniami (3.27)

Jest to permutacja odwrotna do poprzednio wygenerowanej permutacji.

Przytoczony w niniejszym podrozdziale sposób projektowania układów do generowania permutacji w ciele $GF(q)$ charakterystyki 2 dla $q > 2$ mimo niewątpliwych zalet związanych z prostotą i stosunkowo małą liczbą układów służących do ich realizacji, ma także wadę związaną ze złożonością obliczeń na etapie doboru przekształcenia wiążącego operacje w $GF(q)$ z działaniami w $GF(2)$. Jednocześnie metoda ta wydaje się bardzo użyteczna z kryptograficznego punktu widzenia. Pozwala ona na znaczne rozszerzenie liczby możliwych do realizacji generatorów permutacji.

Problem konstruowania układów w oparciu o przekształcenia afiniczne przestrzeni liniowej nad $GF(2)$ prezentuje następny rozdział 3.2.

3.2 Przekształcenia afiniczne przestrzeni liniowych nad $GF(2)$

Niech teraz

$$\langle V, +, \cdot \rangle \quad (3.29)$$

oznacza przestrzeń liniową nad $GF(2)$ gdzie

$V - 2^m$ - elementowy zbiór wektorów nad $GF(2)$ (m - liczba naturalna ≥ 1)

$+, \cdot$ - operacje dodawania i mnożenia przez skalar,

w której operacje dodawania i mnożenia przez skalar wykonuje się podobnie jak w poprzednim rozdziale. Wtedy wzory (3.6)-(3.8) mają taką samą postać, z tym, że działania trzeba teraz wykonywać w ciele $GF(2)$. Wszystkie obliczenia, dotyczące wyznaczania macierzy nieosobliwej S i macierzy do niej odwrotnej można zatem wykonać za pomocą dostępnego oprogramowania wspomagającego obliczenia matematyczne np. MATHEMATICA, MAPLE i in.

Oto przykład takiego postępowania :

PRZYKŁAD 3.2 Sposób generowania permutacji zbioru 256-elementowego w postaci zbioru 8-wymiarowych wektorów przestrzeni liniowej nad $GF(2)$.

Niech

$$x = [x_0 \ x_1 \ \dots \ x_7], \quad y = [y_0 \ y_1 \ \dots \ y_7], \quad c = [c_0 \ c_1 \ \dots \ c_7], \quad (3.30)$$

oznacza trzy wektory tej przestrzeni liniowej. Łatwo sprawdzić, że macierz

$$S = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \quad (3.31)$$

jest nieosobliwa i posiada następującą odwrotność

$$S^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \quad (3.32)$$

Stosując następnie przekształcenie afiniczne typu

$$\pi(x) = x \cdot S + c = y = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} + \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix}^T = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix}^T, \quad (3.33)$$

otrzyma się następujące wzory, wiążące ze sobą składowe wektora oryginalnego i wektora przekształconego

$$\begin{aligned} y_0 &= x_1 + x_2 + x_4 + x_5 + x_6 + x_7 + c_0, \\ y_1 &= x_1 + x_2 + x_3 + x_5 + x_6 + c_1, \\ y_2 &= x_1 + x_3 + x_4 + x_5 + c_2, \\ y_3 &= x_0 + x_2 + x_4 + x_5 + x_6 + c_3, \\ y_4 &= x_0 + x_1 + x_4 + c_4, \\ y_5 &= x_0 + x_2 + x_6 + x_7 + c_5, \\ y_6 &= x_1 + x_5 + c_6, \\ y_7 &= x_0 + x_3 + x_6 + x_7 + c_7. \end{aligned} \quad (3.34)$$

Zakładając dalej, że wektory x , y i c reprezentują liczby binarne

$$x_N = \sum_{j=1}^8 x_{8-j} \cdot 2^{8-j}, \quad y_N = \sum_{j=1}^8 y_{8-j} \cdot 2^{8-j}, \quad c_N = \sum_{j=1}^8 c_{8-j} \cdot 2^{8-j}, \quad (3.35)$$

oraz ustalając $c_N = 170 (10101010_2)$, można zauważyć, że powyższe przekształcenie generuje permutację

(0 170 149 133 152 144 22 203 201 158 236 66 86 96 78 251 155 127 17 15 232 105 112 83
243 29 162 19 88 154 199 100 101 221 168 194 247 54 132 32 229 124 254 8 44 72 135 207

226 184 223 255 176 89 34 178 14 80 28 26 102 50 175 6 214 193 24 49 64 1 18 224 239
 173 81 164 111 12 7 110 180 114 4 129 179 182 37 118 47 167 128 11 195 79 67 238 21 36
 206 90 205 181 202 113 235 134 119 151 210 234 62 2 253 231 43 140 166 56 126 169 122
 130 92 177 225 87 216 59 145 174 190 163 171 45 240 242 165 215 121 109 91 117 192 160
 68 42 52 211 82 75 104 200 38 153 40 99 161 252 95 94 230 147 249 204 13 191 27 222 71
 197 51 23 115 188 244 217 131 228 196 139 98 25 137 53 107 39 33 93 9 148 61 237 250 35
 10 123 58 41 219 212 150 106 159 84 55 60 85 143 73 63 186 136 141 30 77 20 156 187 48
 248 116 120 213 46 31 245 97 246 142 241 74 208 189 76 172 233 209 5 57 198 220 16 183
 157 3 69 146 65 185 103 138 218 108 227) (70 125)

Rys. 3.3. Permutacja opisana równaniami (3.34)

składającą się z dwóch rozłącznych cykli.

Przekształcenie odwrotne do przekształcenia (3.33) ma następującą postać

$$\pi^{-1}(y) = x = (y + c) \cdot S^{-1} = \begin{bmatrix} y_0 + c_0 \\ y_1 + c_1 \\ y_2 + c_2 \\ y_3 + c_3 \\ y_4 + c_4 \\ y_5 + c_5 \\ y_6 + c_6 \\ y_7 + c_7 \end{bmatrix}^T \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}^T. \quad (3.36)$$

Wektor c_N ma tutaj taką samą wartość jak dla przekształcenia poprzedniego. Znak + w zależności wynika z charakterystyki ciała, którego elementy są składowymi wektorów rozpatrywanej teraz przestrzeni: w ciele $GF(2)$ $x + x = 0$, czyli $x = -x$.

Operacje macierzowe przedstawione we wzorze (3.36) dają w rezultacie następujący układ równań

$$\begin{aligned} x_0 &= y_0 + y_1 + y_2 + y_3 + y_4 + y_5 + c_0 + c_1 + c_2 + c_3 + c_4 + c_5, \\ x_1 &= y_0 + y_4 + y_5 + y_6 + c_0 + c_4 + c_5 + c_6, \\ x_2 &= y_1 + y_3 + y_4 + y_5 + y_7 + c_1 + c_3 + c_4 + c_5 + c_7, \\ x_3 &= y_1 + y_3 + y_4 + c_1 + c_3 + c_4, \\ x_4 &= y_1 + y_2 + y_3 + y_4 + y_6 + c_1 + c_2 + c_3 + c_4 + c_6, \\ x_5 &= y_0 + y_4 + y_5 + c_0 + c_4 + c_5, \\ x_6 &= y_1 + y_5 + y_6 + y_7 + c_1 + c_5 + c_6 + c_7, \\ x_7 &= y_0 + y_1 + y_2 + y_6 + c_0 + c_1 + c_2 + c_6, \end{aligned} \quad (3.37)$$

który pozwala na wygenerowanie permutacji

(0 227 108 218 138 103 185 65 146 69 3 157 183 16 220 198 57 5 209 233 172 76 189 208
74 241 142 246 97 245 31 46 213 120 116 248 48 187 156 20 77 30 141 136 186 63 73 143
85 60 55 84 159 106 150 212 219 41 58 123 10 35 250 237 61 148 9 93 33 39 107 53 137 25
98 139 196 228 131 217 244 188 115 23 51 197 71 222 27 191 13 204 249 147 230 94 95
252 161 99 40 153 38 200 104 75 82 211 52 42 68 160 192 117 91 109 121 215 165 242 240
45 171 163 190 174 145 59 216 87 225 177 92 130 122 169 126 56 166 140 43 231 253 2 62
234 210 151 119 134 235 113 202 181 205 90 206 36 21 238 67 79 195 11 128 167 47 118
37 182 179 129 4 114 180 110 7 12 111 164 81 173 239 224 18 1 64 49 24 193 214 6 175 50
102 26 28 80 14 178 34 89 176 255 223 184 226 207 135 72 44 8 254 124 229 32 132 54 247
194 168 221 101 100 199 154 88 19 162 29 243 83 112 105 232 15 17 127 155 251 78 96 86
66 236 158 201 203 22 144 152 133 149 170) (70 125)

Rys. 3.4. Permutacja opisana równaniami (3.37)

Porównując permutacje z rys.3.3. i 3.4. łatwo zauważyć, że dla każdego x występuje zależność

$$\pi(x) = y \quad \text{i} \quad \pi^{-1}(y) = x \quad (3.38)$$

$$\text{np.} \quad \begin{array}{l} \pi(0) = 170 \quad \text{i} \quad \pi^{-1}(170) = 0 \\ \pi(1) = 18 \quad \text{i} \quad \pi^{-1}(18) = 1 \end{array} \quad \text{itd.} \quad (3.39)$$

Wobec tego przekształcenie (3.36) rzeczywiście generuje permutację odwrotną do permutacji z rys.3.3.

3.3 Realizacja układów do generowania permutacji za pomocą przekształceń afinicznych

Układy generujące permutacje w oparciu o przekształcenia afiniczne przestrzeni liniowych nad ciałami Galois charakterystyki 2 można stosunkowo łatwo skonstruować za pomocą bramek XOR. Ilość bramek będzie jednak zależna od przyjętej metody realizacji. Można przyjąć wariant układów pracujących w oparciu o ustaloną wartość c , lub wartość tę przyjąć jako element sterujący układem.

Istnieje także możliwość redukcji ilości bramek poprzez uproszczenie równań opisujących przekształcenia permutacyjne. Na przykład układ równań (3.34) występujące w przykładzie 3.2 po uwzględnieniu założonej wartości c_N przyjmie następującą postać

$$\begin{aligned}
 y_0 &= x_1 + x_2 + x_4 + x_5 + x_6 + x_7, \\
 y_1 &= x_1 + x_2 + x_3 + x_5 + x_6 + 1, \\
 y_2 &= x_1 + x_3 + x_4 + x_5, \\
 y_3 &= x_0 + x_2 + x_4 + x_5 + x_6 + 1, \\
 y_4 &= x_0 + x_1 + x_4, \\
 y_5 &= x_0 + x_2 + x_6 + x_7 + 1, \\
 y_6 &= x_1 + x_5, \\
 y_7 &= x_0 + x_3 + x_6 + x_7 + 1.
 \end{aligned} \tag{3.40}$$

możliwą do zrealizowania za pomocą mniej niż 20 dwuwęściowych układów XOR.

Równania (3.37) przy powyższych założeniach uproszczą się do postaci

$$\begin{aligned}
 x_0 &= y_0 + y_1 + y_2 + y_3 + y_4 + y_5 + 1, \\
 x_1 &= y_0 + y_4 + y_5 + y_6 + 1, \\
 x_2 &= y_1 + y_3 + y_4 + y_5 + y_7, \\
 x_3 &= y_1 + y_3 + y_4, \\
 x_4 &= y_1 + y_2 + y_3 + y_4 + y_6, \\
 x_5 &= y_0 + y_4 + y_5 + 1, \\
 x_6 &= y_1 + y_5 + y_6 + y_7 + 1, \\
 x_7 &= y_0 + y_1 + y_2 + y_6 + 1,
 \end{aligned} \tag{3.41}$$

co można zrealizować za pomocą 20 dwuwęściowych bramek XOR.

Oczywiście ostateczna postać takiego układu zależna jest od przyjętej metody optymalizacji i może się nieco różnić od powyższego wyniku.

Układy tego typu mogą być elementami urządzeń szyfrujących, w których zastosowano szyfry podstawieniowe i przestawieniowe. Należy jednak pamiętać, że przekształcenia afiniczne generują wzajemnie odwracalne pary permutacji, co oznacza w większości przypadków konieczność zastosowania dwóch różnych układów w jednym urządzeniu szyfrująco-deszyfrującym.

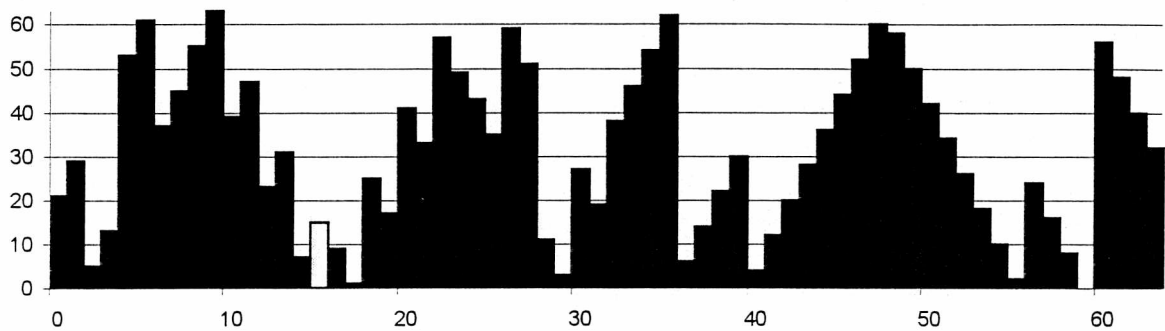
3.4 Własności permutacji wygenerowanych za pomocą przekształceń afinicznych

Permutacje uzyskane za pomocą zaprezentowanych przekształceń afinicznych nad $GF(2^n)$ zawierają co najmniej jeden długi cykl. Występowanie wymienionej cechy w dużym stopniu zależy jednak postaci generującego permutację równania. W trakcie badań prowadzonych nad tym typem przekształcenia odkryto, że zamieniając wartość stałej c z równania (3.6) w zakresie od 0 do 255 dla stałej wartości postaci elementu grupy liniowej uzyskiwano do 2 elementów stałych w permutacji i rząd permutacji zmieniał się ze 127 na 254 z podcyklami o długościach 127, 1 i 2, 254.

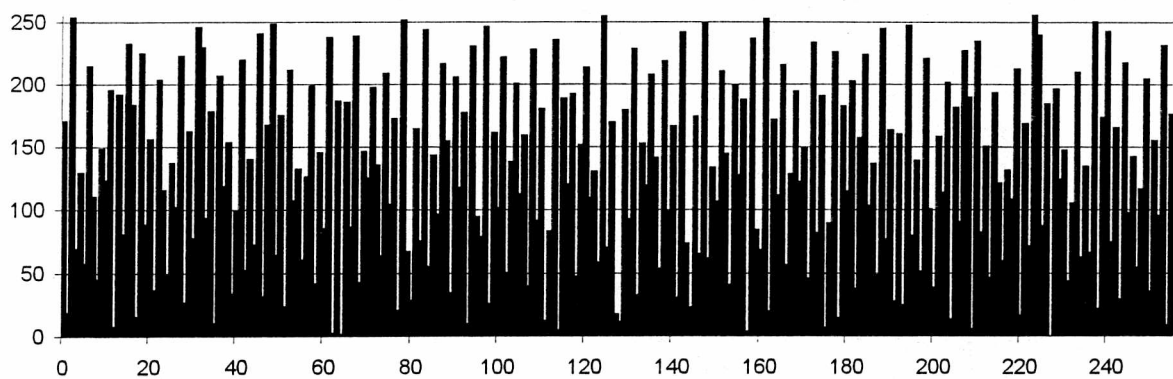
Nieco gorsze wyniki uzyskano testując właściwości permutacji uzyskanych za pomocą przekształcenia afinicznego przestrzeni liniowej nad $GF(8)$. Permutacja przedstawiona w podrozdziale 3.1 jest cykliczna. Składa się z wyłącznie z 7 - elementowych cykli i elementu stałego o wartości 15. Badania polegające na generowaniu permutacji dla wartości stałej zmieniającej się w zakresie 0 do 63 wykazały, że długość stałego cyklu jest elementem stałym niniejszego przekształcenia, zmienna jest natomiast wartość elementu stałego.

Taki wynik doświadczenia prowadzi do wniosku, że charakter permutacji uzyskanej za pomocą wzorów przedstawionych w przykładzie 3.1 zależy od postaci macierzy S , zaś długość cyklu powtarzającego się w permutacji ma związek ze sposobem odwzorowania pojedynczego elementu rozważanej przestrzeni za pomocą elementów $GF(2)$.

Mimo różnic permutacji przedstawionych powyżej wynikających z analizy ich rozkładu na cykle, permutacje wygenerowane za pomocą przekształceń afinicznych zawierają cechy wspólne, które pozwalają odróżnić je od wygenerowanych w inny sposób. Cechy te najłatwiej zaobserwować na histogramie prezentującym przekształcenia opisane w przykładzie 3.1 i 3.2.



Rys. 3.5. Histogram permutacji z rys.3.1.



Rys. 3.6. Histogram permutacji z rys.3.3.

W obydwu przypadkach widoczne są charakterystyczne, cyklicznie powtarzające się „piramidki” trendów liniowych wskazujące na użyteczną cechę tego przekształcenia, polegającą na rozpraszaniu grup wartości liczbowych zgrupowanych w dowolnym krótkim zakresie, poprzez równomierne odwzorowywanie ich w pełen zakres wartości.

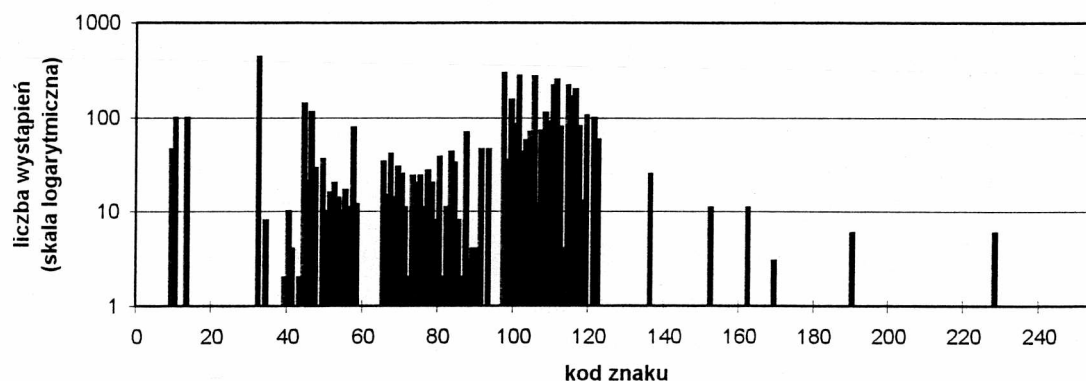
Zagadnienie to dobrze ilustrują zamieszczone na rys. 3.7 oraz rys. 3.8 charakterystyki, pokazujące działanie mechanizmu rozpraszania wartości w prostym kryptosystemie, zmieniającym znaki tekstu jawnego według wzorca, wygenerowanego przez permutację, uzyskaną za pomocą przekształcenia afinicznego nad $GF(2)$. Kryptosystem ten szyfruje dane zgodnie ze wzorem

$$Y_i = P(X_i), \quad (3.42)$$

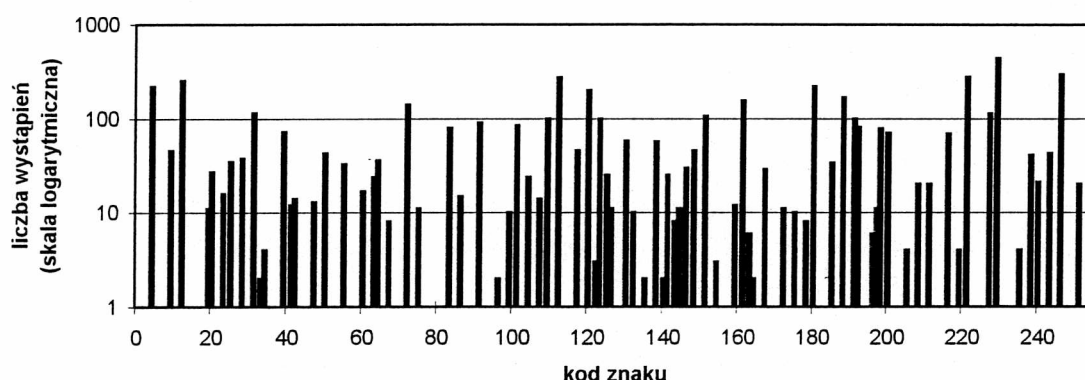
gdzie

X_i, Y_i - odpowiednio wartość i -tego bajtu tekstu jawnego przed i po zaszyfrowaniu,
 P - przekształcenie generujące permutację

Rys. 3.7 i 3.8 pokazują, jaki jest rozkład częstości znaków kodu ASCII w pliku zawierającym tekst jawny oraz w tekst zaszyfrowany wg (3.42). Na osi odciętych umieszczono wartości kodu ASCII poszczególnych znaków, zaś na osi rzędnych ilość wystąpień tych znaków w tekście.



Rys. 3.7. Rozkład kodów ASCII w tekście przed zaszyfrowaniem



Rys. 3. 8. Rozkład kodów ASCII w tekście po zaszyfrowaniu

Do szyfrowania wybrano plik tekstowy wielkości ok. 30kB o rozkładzie częstotliwości wystąpień znaków jak na rys. 3.7. Widać tu, że większość znaków tekstu jawnego mieści się w zakresie kodów z przedziału 32÷130. Zaszyfrowanie pliku dało efekt widoczny na rys. 3.8. Wartości zgrupowane w zakresie znaków o niskich kodach zostały rozproszone po całym zakresie kodów ASCII. Jest to wynik działania szyfru podstawieniowego, w którym przekształcenie P z (3.42) opisują równania (3.34).

Z analizy wzorów (3.34) i (3.37) można także wywnioskować, o ich przydatności w układach realizujących tzw. efekt lawinowy, w których zmiana jednego bitu w tekście jawnym powinna powodować radykalną zmianę postaci szyfrogramu.

Warto także zwrócić uwagę na dużą szybkość działania takich układów. W przypadku realizacji sprzętowej czas propagacji sygnału przez układ generujący permutację zależy wyłącznie od ilości warstw bramek XOR użytych do jego konstrukcji. Przykładowo dla wzorów (3.34) lub (3.37) będzie to 5 warstw.

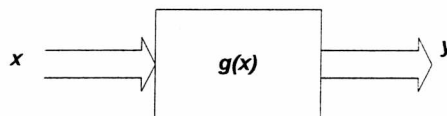
4. Metoda generowania samoodwracalnych permutacji zbioru 2^m - elementowego.

W rozdziale tym przedstawiono metodę projektowania generatorów permutacji z wykorzystaniem układów do obliczania inwersji ciał Galois charakterystyki 2. Rozważania przeprowadzono dla ciał $GF(2^m)$ oraz pewnej klasy ciał określonych jako $GF(q^c)$, gdzie $q = 2^k$, przy czym $m = kc$.

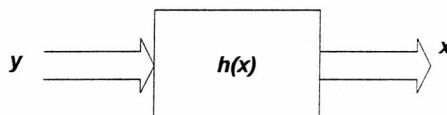
4.1 Permutacje samoodwracalne

Stosowanie par permutacji wzajemnie odwracalnych do konstruowania urządzeń kryptograficznych powoduje konieczność rozdzielenia przetwarzanego sygnału oraz wymusza budowanie oddzielnych układów dla szyfratorów i deszyfratorów; co pokazano na rysunku 4.1

szyfrowanie



deszyfrowanie



Rys.4.1. Szyfrowanie i deszyfrowanie w układach z permutacjami wzajemnie odwracalnymi,

gdzie

g, h - permutacje wzajemnie odwracalne

Wad tych pozbawione są układy zrealizowane z zastosowaniem permutacji samoodwracalnych. W takich układach do generowania permutacji zarówno w procesie szyfrowania jak i deszyfrowania używany jest ten sam blok funkcjonalny, w którym realizowana jest operacja spełniająca zależność

$$f(x_0) = y_0 \Rightarrow f(y_0) = x_0, \quad (4.1)$$

przy czym ważne jest, aby w wygenerowanej permutacji jak najwięcej elementów spełniało zależność $x_0 \neq y_0$.

Jednym z rozwiązań realizujących permutacje samoodwracalne są układy do obliczania odwrotności w ciałach Galois. Takie rozwiązania odznaczają się jednak dużym stopniem złożoności i dlatego ich zastosowanie w systemach kryptograficznych nie jest powszechne. Innym problemem tutaj występującym jest zagadnienie złożoności obliczeń występujących przy ich projektowaniu, która dla zadań dotyczących bezpośredniego obliczania odwrotności w ciałach Galois z bazą wielomianową jest rzędu $O(m^2)$ [Mat91]. W przypadku technologii VLSI rozważano sensowność takich rozwiązań jedynie dla $m < 6$. Z tego powodu popularne były realizacje oparte na własności ciała Galois wyrażającej się wzorem

$$\beta = \beta^{2^m}, \quad (4.2)$$

co po przekształceniach można sprowadzić do postaci

$$\beta^{-1} = (\dots ((\beta)^2) \beta)^2 \dots \beta)^2 \quad \text{dla} \quad \beta \in GF(2^m). \quad (4.3)$$

Jak widać z powyższego wzoru procedura obliczania odwrotności elementu m -tego rozszerzenia ciała $GF(2)$ polega na wielokrotnym podnoszeniu do kwadratu i mnożeniu w tym ciele¹. Nie jest to rozwiązanie optymalne, bowiem jego realizacja układowa wiąże się z zastosowaniem zbyt wielu bloków mnożących. Wystarczy powiedzieć, że aby obliczyć odwrotność elementu ciała $GF(2^8)$ operację mnożenia w tym ciele należy wykonać sześciokrotnie, zaś podnoszenie do kwadratu ośmiokrotnie. Daje to całkowitą liczbę szesnastu kolejno następujących po sobie bloków funkcjonalnych, co wydłuża czas propagacji sygnału przez blok funkcjonalny.

W niniejszym rozdziale zaprezentowano dwie metody projektowania układów do obliczania inwersji w ciałach Galois charakterystyki 2. Przedstawiono sposób realizacji w $GF(2^m)$ oraz dla pewnej klasy ciał $GF(q^c)$, gdzie $q = 2^k$, gdzie $m = kc$.

W rozdziale dokonano porównania obu metod, zwracając szczególną uwagę na sposób wykonywania obliczeń projektowych, złożoność równań generujących permutacje oraz jakość uzyskanych przekształceń, z punktu widzenia własności rozważanych w rozdziale 3.4.

¹ Takie rozwiązanie zostało dokładniej przedstawiono w rozdziale 2.

4.2 Inwersja w $GF(2^m)$ jako permutacja samoodwracalna zbioru 2^m - elementowego

Metodę obliczania odwrotności w ciele $GF(2^m)$ wraz z przykładem 2.1 opisano w rozdziale 2. Przykład 2.1, który dotyczy obliczania inwersji w ciele $GF(2^6)$ może posłużyć do pokazania w jaki sposób układy do wyznaczania odwrotności mnożymy można zastosować do generowania permutacji samoodwracalnych.

W rozpatrywanym przykładzie na wejście inwertera podaje się współczynniki wielomianu (2.3) $A_0, A_1, \dots, A_5$. Można więc przyjąć, że wielomian odpowiada liczbie

$$x_N = \sum_{i=0}^5 A_i \cdot 2^i. \quad (4.9)$$

Z układu równań (2.16) wynika, że odwrotnością wielomianu (2.3) jest wielomian (2.8) o współczynnikach $I_0, I_1, \dots, I_5$. Ten wielomian można skojarzyć z liczbą

$$y_N = \sum_{i=0}^5 I_i \cdot 2^i. \quad (4.10)$$

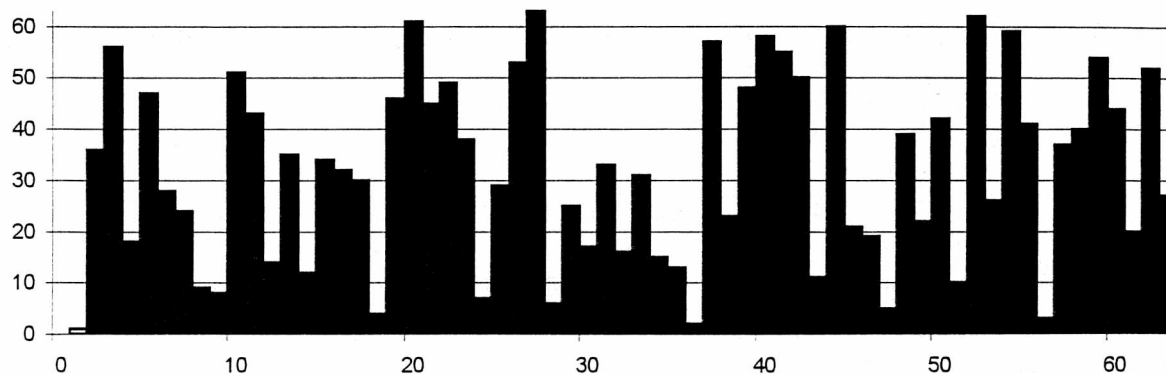
Przyjmując, że $x_N = 0, 1, \dots, 63$, obliczyć można y_N dla każdej wartości x_N .

W wyniku tych obliczeń otrzyma się permutację:

(0)(1)(2 36)(3 56)(4 18)(5 47)(6 28)(7 24)(8 9)(10 51)(11 43)(12 14)
 (13 35)(15 34)(16 32)(17 30)(19 46)(20 61)(21 45)(22 49)(23 38)(25 29)
 (26 53)(27 63)(31 33)(37 57)(39 48)(40 58)(41 55)(42 50)(44 60)(52 62)
 (54 59)

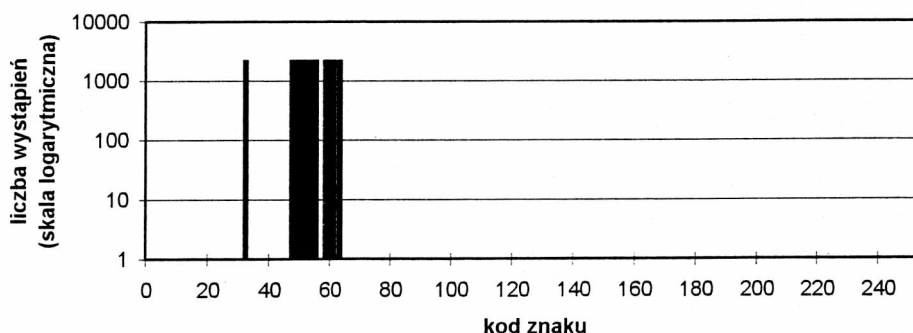
Rys. 4.2. Permutacja opisana wzorami do obliczania inwersji z przykładu 2.1

Jak widać z powyższych wartości permutacja jest rzeczywiście samoodwracalna, ponieważ składa się z 31 (czyli $2^{m-1} - 1$) rozłącznych transpozycji. Jedynymi wartościami, które w przekształceniu przechodzą same na siebie są 0 i 1.



Rys.4.3. Histogram permutacji z przykładu 4.1.

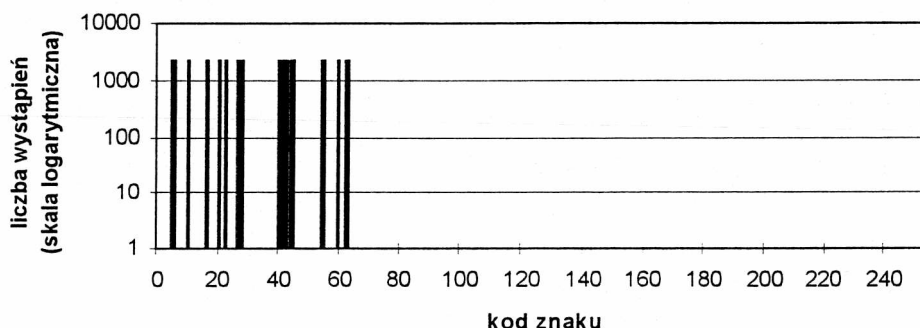
Na podstawie zamieszczonego powyżej histogramu można dostrzec nieliniowy charakter wykorzystanego do wygenerowania permutacji przekształcenia. Brak w nim wyraźnie rozróżnialnych powtarzalnych elementów, szczególnie długookresowych trendów liniowych, tak wyraźnie widocznych w przekształceniach, przedstawionych w poprzednim rozdziale. Dostrzegalna jest także cecha rozpraszania wartości (opisana w poprzednim rozdziale) po całym zakresie dostępnych wartości, o tyle cenniejsza od występującej w przekształceniu afinicznym, że brak tutaj regularności, które w pewnych warunkach mogą umożliwić aproksymację zastosowanej w przekształceniu funkcji. Własność tę łatwo można zauważyć, analizując za pomocą histogramów działanie opisanego w rozdziale 3.4 szyfru podstawieniowego.



Rys. 4.4. Częstość występowania znaków kodu ASCII w pliku przed zaszyfrowaniem

Jak widać z powyższej charakterystyki zamieszczonej na rys. 4.4, w pliku przed zaszyfrowaniem występowały jedynie kody ASCII o wartościach z zakresu 30÷63, czyli

zakresu przetwarzanego przez 6 - bitowy generator permutacji. Po zaszyfrowaniu, zaś w kryptogramie występują wartości także spoza tego zakresu, co można zobaczyć na histogramie z rys. 4.5.



Rys. 4.5. Częstość występowania znaków kodu ASCII w pliku po zaszyfrowaniu

Układ realizujący przedstawioną permutację charakteryzuje się przeciętnym stopniem złożoności o czym może świadczyć fakt, że można go skonstruować za pomocą 130 dwuwęściowych bramek XOR oraz ok. 300 dwuwęściowych układów AND. Oszacowania tego dokonano jednak nie minimalizując opisujących go równań. Nawet pobieżna analiza ich postaci pozwala przypuszczać, że regularności w nich występujące mogą spowodować redukcję przyjętej ilości układów nawet o połowę [Cha93]. Daje to możliwości oszacowania ścieżki krytycznej całego bloku generującego permutację na 50 sekwencyjnie występujących pojedynczych bramek logicznych.

Przedstawiona w powyższym podrozdziale metoda projektowania układów do generowania permutacji samoodwracalnych posiada jedną poważną wadę. Wykonywane w jej trakcie obliczenia są dość kłopotliwe, ze względu na konieczność odwracania macierzy symbolicznej $m \times m$, a otrzymane wyniki cechuje duża złożoność. Stosując jednak specjalistyczne oprogramowanie do wykonywania obliczeń w $GF(2)$, z łatwością można je wykonać nawet za pomocą sprzętu mikrokomputerowego typu PC przeciętnej klasy. Ilustracją tej tezy niech będzie fakt, że obliczenie układu do generowania permutacji samoodwracalnej dla $m = 8$ przeprowadzone na mikrokomputerze PC z zegarem 66MHz i pamięcią RAM 16MB trwało ok.120 min. Wyniki obliczeń w tym przypadku zawierały się na 7 stronach formatu A4, a realizacja układu opisanego tymi równaniami jest praktycznie trudna.

Pewien sposób ominięcia problemu obliczeń macierzowych występujących w trakcie projektowania układów permutacyjnych z jednoczesnym rozszerzeniem ilości dostępnych dla danego m wielomianów nieredukowalnych $P(x)$ przedstawia kolejny podrozdział.

4.3 Inwersja w $GF(q^c)$, $q = 2^k$, $kc = m$ jako permutacja samoodwracalna zbioru 2^m - elementowego

Ciało $GF(q^c)$ jest c - krotnym rozszerzeniem ciała $GF(q)$. Jeżeli $q = 2^k$, to każdy element $GF(q)$ można przedstawić w postaci k - wymiarowego wektora nad $GF(2)$, czyli

$$\forall A \in GF(q) \quad A = [a_0 a_1 a_2 \dots a_{k-1}], \quad (4.10)$$

gdzie

$$a_0, a_1, a_2, \dots, a_{k-1} \in GF(2),$$

co odpowiada wielomianowi nad $GF(2)$ o postaci

$$A(x) = a_0 + a_1 x + \dots + a_{k-1} x^{k-1}. \quad (4.11)$$

Gdy k jest małe (2 lub 3), to łatwo można wyznaczyć wzory wiążące podstawowe operacje w $GF(q)$ z operacjami w $GF(2)$. Daje to jednocześnie możliwość wyrażania zależności występujących w c - tym rozszerzeniu $GF(q)$ za pomocą elementów $GF(2)$.

Biorąc pod uwagę powyższe wnioski rozważmy dowolny wielomian nieredukowalny trzeciego stopnia ponad $GF(q)$, gdzie $q = 2^{2^k}$, $k = 1, 2, \dots$, o postaci

$$P(x) = P_0 + P_1 x + P_2 x^2 + x^3. \quad (4.12)$$

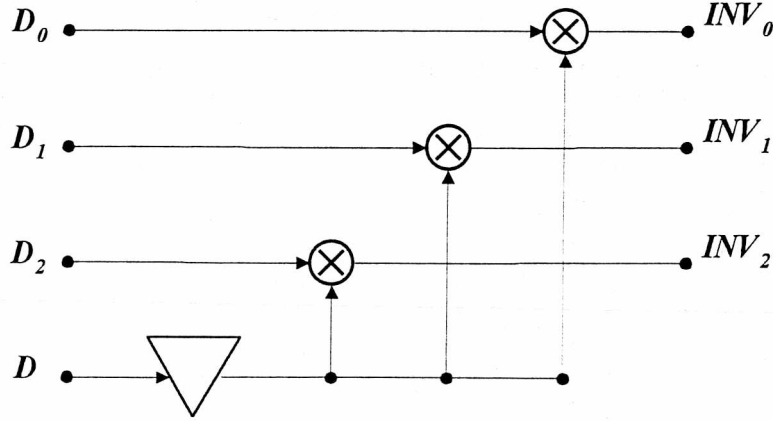
Każdy element ciała $GF(4^3)$ wygenerowanego za pomocą $P(x)$ można przedstawić za pomocą wielomianu

$$A(x) = A_0 + A_1 x + A_2 x^2. \quad (4.13)$$

Odwrotność takiego elementu można wyrazić można jako [Koś95]

$$\begin{aligned} \frac{1}{A(x)} &= INV_0 + INV_1 x + INV_2 x^2, \quad A(x) \neq 0 \\ INV_0 &= \frac{D_0}{D}; \quad INV_1 = \frac{D_1}{D}; \quad INV_2 = \frac{D_2}{D}; \end{aligned} \quad (4.14)$$

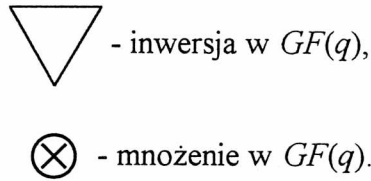
Schemat blokowy, prezentujący sieć działań wynikającą z równania (4.14) pokazano na rys. 4.6.



Rys. 4.6. Schemat obliczania współczynników z równania (4.14).

Obliczenie wartości INV_0, INV_1, INV_2 w $GF(q)$ wymaga wyznaczenia iloczynów D_0, D_1 i D_2 przez odwrotność wartości wyznacznika D .

Działania matematyczne w $GF(q)$ oznaczono na tym schemacie następującymi symbolami:



Rys. 4.7. Symbole operacji w $GF(q)$.

Wartości D, D_0, D_1, D_2 można obliczyć z wzorów

$$\begin{aligned}
 D &= A_0^3 + A_0^2 P_2 A_1 + A_0^2 P_2^2 A_2 + A_0 P_1^2 A_2^2 + A_0 P_1 A_2 P_2 A_1 + A_0 P_0 A_1 A_2 + A_0 P_1 A_1^2 + A_1 P_0 A_2^2 P_1 + \\
 &\quad + P_0 A_2 P_2 A_1^2 + P_0 A_1^3 + P_0^2 A_2^3 \\
 D_0 &= A_0^2 + A_0 P_2 A_1 + A_0 P_2^2 A_2 + P_1^2 A_2^2 + P_1 A_2 P_2 A_1 + P_0 A_1 A_2 + P_0 A_2^2 P_2 + P_1 A_1^2 \\
 D_1 &= A_0 A_1 + P_2 A_1^2 + A_1 P_2^2 A_2 + A_2^2 P_0 + P_1 A_2^2 P_2 \\
 D_2 &= A_1^2 + P_2 A_1 A_2 + A_0 A_2 + A_2^2 P_1
 \end{aligned} \tag{4.15}$$

Oczywiście wszystkie operacje zawarte w powyższych wzorach są realizowane na elementach $GF(q)$ i ulegną one uproszczeniu po przyjęciu konkretnej wartości $P(x)$.

Aby teraz otrzymać powyższe zależności w postaci odpowiedniej do realizacji w $GF(2)$, należy wyznaczyć wzory pozwalające na obliczenie mnożenia, potęgowania i inwersji w $GF(2^k)$.

PRZYKŁAD 4.2 Rozważmy ciało $GF(4^3)$ o wielomianie nieredukowalnym $P(x)=x^3+2$. Biorąc pod uwagę wzory zawarte w (4.12) i fakt, że $P_0=2$ i $P_1=P_2=0$, otrzymamy następujące zależności służące do obliczania inwersji w tym ciele

$$\begin{aligned} D &= A_0^3 + 2 A_0 A_1 A_2 + 2 A_1^3 + 3 A_2^3, \\ D_0 &= A_0^2 + 2 A_1 A_2, \\ D_1 &= A_0 A_1 + 2 A_2^2, \\ D_2 &= A_1^2 + A_0 A_2. \end{aligned} \quad (4.16)$$

Dla ułatwienia analizy problemu, tak jak w przypadku wzorów (4.14), powyższe zależności przedstawiono graficznie. Symbole użyte na schematach z rys.4.9, 4.10, 4.11, 4.12 mają następujące znaczenie:

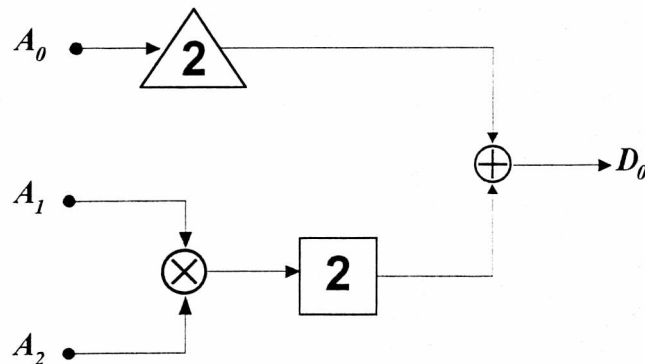
$\otimes$ - mnożenie w $GF(4)$,

$\oplus$ - dodawanie w $GF(4)$,

$\triangle 2$ - potęgowanie w $GF(4)$ z wykładnikiem podanym w trójkącie,

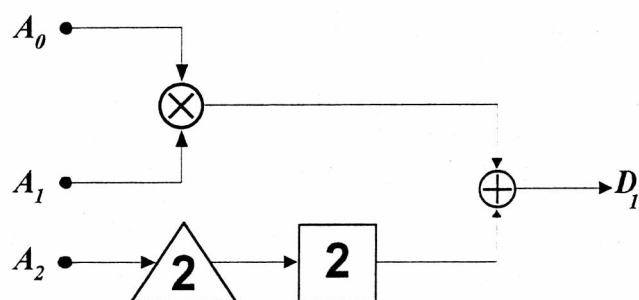
$\square 2$ - mnożenie w $GF(4)$ przez stałą podaną w kwadracie.

Rys. 4.8. Symbole operacji arytmetycznych w $GF(4)$



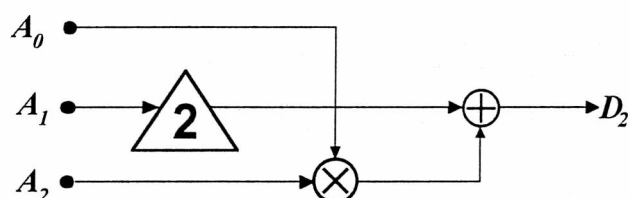
Rys. 4.9. Schemat obliczania wyznacznika D_0 według wzoru (4.16).

Zgodnie ze schematem na rys.4.9, aby wyznaczyć wartość D_0 najpierw oblicza się iloczyn A_1 przez A_2 , potem mnoży go przez stałą 2 a następnie dodaje do kwadratu A_0 .



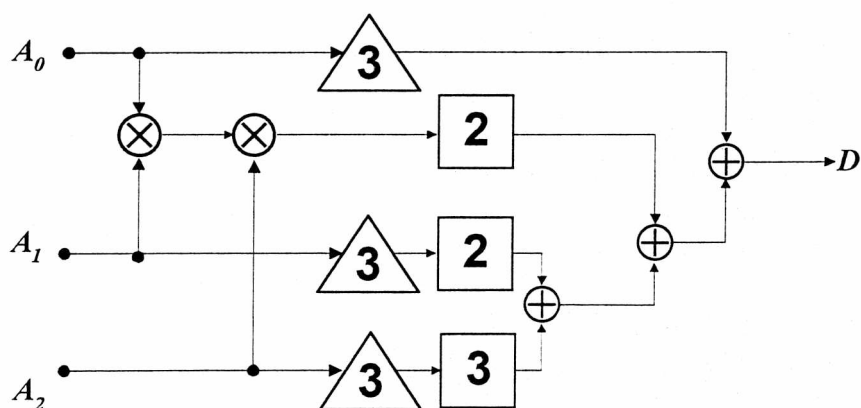
Rys. 4.10. Schemat obliczania wyznacznika D_1 według wzoru (4.16).

Jak wynika z rys.4.10, aby obliczyć D_1 należy iloczyn A_0 przez A_1 dodać do kwadratu A_2 pomnożonego przez 2.



Rys. 4.11. Schemat obliczania wyznacznika D_2 według wzoru (4.16).

Wyznaczenia D_2 dokonuje się poprzez dodanie do iloczynu A_0 przez A_2 do iloczynu A_1 przez stałą 2.



Rys. 4.12. Schemat obliczania wyznacznika D według wzoru (4.16).

Zgodnie ze schematem na rys.4.12, aby obliczyć wartość D , należy wyznaczyć iloczyn A_0 przez A_1 , pomnożyć przez A_2 i przez 2. Wynik trzeba dodać do sumy iloczynów A_1^3 przez 2 i A_2^3 przez 3. Otrzymaną wartość dodaje się do obliczonej trzeciej potęgi A_0 .

Każdy element $GF(4)$ można przedstawić za pomocą dwuwymiarowego wektora nad $GF(2)$, zatem przyjmuje się następujące oznaczenia

$$\begin{aligned} A_0 &= [a_{00} \ a_{01}] & D_0 &= [d_{00} \ d_{01}] & INV_0 &= [inv_{00} \ inv_{01}] \\ A_1 &= [a_{10} \ a_{11}] & D_1 &= [d_{10} \ d_{11}] & INV_1 &= [inv_{10} \ inv_{11}] \\ A_2 &= [a_{20} \ a_{21}] & D_2 &= [d_{20} \ d_{21}] & INV_2 &= [inv_{20} \ inv_{21}] \\ D &= [d_0 \ d_1] \end{aligned} \quad (4.17)$$

Jednocześnie, jak widać zamieszczonej analizy, aby obliczyć inwersję elementu rozpatrywanego ciała $GF(4^3)$ należy wyznaczyć wzory pozwalające na wykonywanie operacji mnożenia, podnoszenia do kwadratu, odwracania, mnożenia przez 2 i przez 3 w ciele $GF(4)$, którego elementy przedstawia się za pomocą dwuwymiarowych wektorów z elementami nad $GF(2)$.

Ciało $GF(4)$ utworzone zostało za pomocą wielomianu drugiego stopnia $p(x)$ nieredukowalnego nad $GF(2)$

$$p(x) = 1 + x + x^2 \quad (4.18)$$

Jednocześnie $GF(4) = \langle \{0,1,2,3\}, +, \cdot \rangle$, gdzie

$$0 = [00], \ 1 = [10], \ 2 = [01], \ 3 = [11].$$

Przyjmując także reprezentację elementów $a, b \in GF(4)$ w postaci dwuwymiarowych wektorów nad $GF(2)$ jako

$$a = [a_0 \ a_1], \quad b = [b_0 \ b_1],$$

co odpowiada wielomianom nad $GF(2)$

$$a(x) = a_0 + a_1 x, \quad b(x) = b_0 + b_1 x, \quad (4.19)$$

łatwo teraz można wyznaczyć podstawowe zależności potrzebne do przeprowadzania obliczeń w $GF(4)$.

$$\begin{aligned}
ab &= \begin{bmatrix} prd_0 & prd_1 \end{bmatrix}, \quad prd_0 = a_0 b_0 + a_1 b_1, \quad prd_1 = a_0 b_1 + a_1 b_0 + a_1 b_1, \\
\frac{1}{a} &= \begin{bmatrix} inv_0 & inv_1 \end{bmatrix}, \quad inv_0 = a_0 + a_1, \quad inv_1 = a_1, \\
a^2 &= \begin{bmatrix} sqr_0 & sqr_1 \end{bmatrix}, \quad sqr_0 = a_0 + a_1, \quad sqr_1 = a_1, \\
2a &= \begin{bmatrix} mb2_0 & mb2_1 \end{bmatrix}, \quad mb2_0 = a_1, \quad mb2_1 = a_0 + a_1, \\
a^3 &= \begin{bmatrix} tr_0 & tr_1 \end{bmatrix}, \quad tr_0 = a_0 + a_0 a_1 + a_1, \quad tr_1 = 0, \\
3a &= \begin{bmatrix} mb3_0 & mb3_1 \end{bmatrix}, \quad mb3_0 = a_0 + a_1, \quad mb3_1 = a_0.
\end{aligned} \tag{4.20}$$

Co pozwala zapisać wzory (4.15) w postaci

$$\begin{aligned}
d_0 &= tr_{00} + mb2_{0120} + mb2_{t_{10}} + mb3_{t_{20}}, \\
d_1 &= tr_{01} + mb2_{0121} + mb2_{t_{11}} + mb3_{t_{21}}, \\
d_{00} &= sqr_{00} + mb2_{120}, \\
d_{01} &= sqr_{01} + mb2_{121}, \\
d_{10} &= prd_{010} + mb2_{s_{20}}, \\
d_{11} &= prd_{011} + mb2_{s_{21}}, \\
d_{20} &= sqr_{10} + prd_{020}, \\
d_{21} &= sqr_{11} + prd_{021}.
\end{aligned} \tag{4.21}$$

Stosując teraz te zależności do obliczenia elementów występujących we wzorach na inwersję w rozpatrywanym $GF(4^3)$ otrzymujemy

$$D^{-1} = \begin{bmatrix} di_0 & di_1 \end{bmatrix}, \quad di_0 = d_0 + d_1, \quad di_1 = d_1, \tag{4.22}$$

$$\begin{aligned}
A_0^2 &= \begin{bmatrix} sqr_{00} & sqr_{01} \end{bmatrix}, \quad sqr_{00} = a_{00} + a_{01}, \quad sqr_{01} = a_{01}, \\
A_1^2 &= \begin{bmatrix} sqr_{10} & sqr_{11} \end{bmatrix}, \quad sqr_{10} = a_{10} + a_{11}, \quad sqr_{11} = a_{11}, \\
A_2^2 &= \begin{bmatrix} sqr_{20} & sqr_{21} \end{bmatrix}, \quad sqr_{20} = a_{20} + a_{21}, \quad sqr_{21} = a_{21},
\end{aligned} \tag{4.23}$$

$$\begin{aligned}
A_0^3 &= \begin{bmatrix} tr_{00} & 0 \end{bmatrix}, \quad tr_{00} = a_{00} + a_{00} a_{01} + a_{01}, \quad tr_{01} = 0, \\
A_1^3 &= \begin{bmatrix} tr_{10} & 0 \end{bmatrix}, \quad tr_{10} = a_{10} + a_{10} a_{11} + a_{11}, \quad tr_{11} = 0, \\
A_2^3 &= \begin{bmatrix} tr_{20} & 0 \end{bmatrix}, \quad tr_{20} = a_{20} + a_{20} a_{21} + a_{21}, \quad tr_{21} = 0,
\end{aligned} \tag{4.24}$$

$$\begin{aligned}
2A_2^2 &= \begin{bmatrix} mb2_{s_{20}} & mb2_{s_{21}} \end{bmatrix}, \quad mb2_{s_{20}} = sq_{21} = a_{21}, \quad mb2_{s_{21}} = sq_{20} + sq_{21} = a_{20}, \\
2A_1^3 &= \begin{bmatrix} mb2_{t_{10}} & mb2_{t_{11}} \end{bmatrix}, \quad mb2_{t_{10}} = 0, \quad mb2_{t_{11}} = tr_{10} = a_{10} + a_{10} a_{11} + a_{11},
\end{aligned} \tag{4.25}$$

$$\begin{aligned}
3A_2^3 &= \begin{bmatrix} mb3_{t_{20}} & mb3_{t_{21}} \end{bmatrix}, \quad mb3_{t_{20}} = tr_{20} = a_{20} + a_{20} a_{21} + a_{21}, \\
&\quad mb3_{t_{21}} = tr_{20} = a_{20} + a_{20} a_{21} + a_{21}, \quad \Rightarrow OR
\end{aligned} \tag{4.26}$$

$$\begin{aligned}
A_0 A_1 &= \begin{bmatrix} prd_{010} & prd_{011} \end{bmatrix}, \quad prd_{010} = a_{00} a_{10} + a_{01} a_{11}, \quad prd_{011} = a_{00} a_{11} + a_{01} a_{10} + a_{01} a_{11}, \\
A_1 A_2 &= \begin{bmatrix} prd_{120} & prd_{121} \end{bmatrix}, \quad prd_{120} = a_{10} a_{20} + a_{11} a_{21}, \quad prd_{121} = a_{10} a_{21} + a_{11} a_{20} + a_{11} a_{21},
\end{aligned} \tag{4.27}$$

$$A_0 A_2 = \begin{bmatrix} prd_{020} & prd_{021} \end{bmatrix}, \quad prd_{020} = a_{00} a_{20} + a_{01} a_{21}, \quad prd_{021} = a_{00} a_{21} + a_{01} a_{20} + a_{01} a_{21}, \tag{4.28}$$

$$2A_1A_2 = [mb2_{120} \ mb2_{121}], \quad mb2_{120} = prd_{121} = a_{10}a_{21} + a_{11}a_{20} + a_{11}a_{21},$$

$$mb2_{121} = prd_{120} + prd_{121} = a_{10}a_{21} + a_{11}a_{20} + a_{10}a_{20}, \quad (4.29)$$

$$A_0[2A_1A_2] = [mb2_{0120} \ mb2_{0121}], \quad mb2_{0120} = a_0mb2_{120} + a_1mb2_{121},$$

$$mb2_{0121} = a_0mb2_{121} + a_1mb2_{120} + a_1mb2_{121}. \quad (4.30)$$

Po podstawieniu powyższych zależności do wzorów (4.16) ostatecznie otrzymuje się równania (4.31)

$$\begin{aligned} inv_{00} &= a_{00} + a_{01} + a_{00}a_{10} + a_{00}a_{11} + a_{00}a_{10}a_{11} + a_{01}a_{20} + a_{11}a_{20} + a_{01}a_{11}a_{20} + a_{00}a_{01}a_{11}a_{20} + \\ &\quad a_{00}a_{10}a_{11}a_{20} + a_{01}a_{10}a_{11}a_{20} + a_{01}a_{21} + a_{10}a_{21} + a_{01}a_{10}a_{21} + a_{00}a_{01}a_{10}a_{21} + a_{11}a_{21} + \\ &\quad a_{00}a_{11}a_{21} + a_{00}a_{01}a_{11}a_{21} + a_{00}a_{10}a_{11}a_{21} + a_{01}a_{10}a_{11}a_{21} + a_{01}a_{20}a_{21} + a_{00}a_{10}a_{20}a_{21} + \\ &\quad a_{01}a_{10}a_{20}a_{21} + a_{00}a_{11}a_{20}a_{21} + a_{01}a_{11}a_{20}a_{21} + a_{00}a_{10}a_{11}a_{20}a_{21} + a_{01}a_{10}a_{11}a_{20}a_{21} \\ inv_{01} &= a_{01} + a_{00}a_{10} + a_{01}a_{10} + a_{00}a_{11} + a_{01}a_{11} + a_{00}a_{10}a_{11} + a_{01}a_{10}a_{11} + a_{00}a_{20} + a_{10}a_{20} + \\ &\quad a_{00}a_{01}a_{10}a_{20} + a_{11}a_{20} + a_{01}a_{11}a_{20} + a_{00}a_{01}a_{11}a_{20} + a_{01}a_{10}a_{11}a_{20} + a_{00}a_{21} + a_{10}a_{21} + \\ &\quad a_{01}a_{10}a_{21} + a_{00}a_{01}a_{10}a_{21} + a_{01}a_{10}a_{11}a_{21} + a_{00}a_{20}a_{21} + a_{01}a_{10}a_{20}a_{21} + a_{01}a_{11}a_{20}a_{21} + \\ &\quad a_{01}a_{10}a_{11}a_{20}a_{21} \\ inv_{10} &= a_{01}a_{10} + a_{00}a_{11} + a_{01}a_{11} + a_{20} + a_{10}a_{20} + a_{01}a_{10}a_{20} + a_{00}a_{01}a_{10}a_{20} + a_{11}a_{20} + a_{00}a_{11}a_{20} + \\ &\quad a_{00}a_{01}a_{11}a_{20} + a_{10}a_{11}a_{20} + a_{00}a_{10}a_{11}a_{20} + a_{01}a_{10}a_{11}a_{20} + a_{00}a_{01}a_{10}a_{11}a_{20} + a_{00}a_{21} + a_{01}a_{21} \\ &\quad + a_{00}a_{01}a_{21} + a_{10}a_{21} + a_{11}a_{21} + a_{10}a_{11}a_{21} + a_{01}a_{10}a_{20}a_{21} + a_{00}a_{11}a_{20}a_{21} + a_{01}a_{11}a_{20}a_{21} \\ inv_{11} &= a_{00}a_{10} + a_{01}a_{10} + a_{00}a_{11} + a_{20} + a_{00}a_{20} + a_{01}a_{20} + a_{00}a_{01}a_{20} + a_{01}a_{10}a_{20} + a_{00}a_{01}a_{10}a_{20} + \\ &\quad a_{00}a_{11}a_{20} + a_{00}a_{01}a_{11}a_{20} + a_{00}a_{10}a_{11}a_{20} + a_{01}a_{10}a_{11}a_{20} + a_{00}a_{01}a_{10}a_{11}a_{20} + a_{21} + a_{10}a_{21} + \\ &\quad a_{00}a_{10}a_{21} + a_{00}a_{01}a_{10}a_{21} + a_{11}a_{21} + a_{00}a_{01}a_{11}a_{21} + a_{10}a_{11}a_{21} + a_{00}a_{10}a_{11}a_{21} + a_{01}a_{10}a_{11}a_{21} \\ &\quad + a_{00}a_{01}a_{10}a_{11}a_{21} + a_{00}a_{10}a_{20}a_{21} + a_{01}a_{10}a_{20}a_{21} + a_{00}a_{11}a_{20}a_{21} \\ inv_{20} &= a_{10} + a_{00}a_{10} + a_{01}a_{10} + a_{00}a_{01}a_{10} + a_{00}a_{11} + a_{01}a_{11} + a_{00}a_{01}a_{11} + a_{00}a_{20} + a_{01}a_{20} + \\ &\quad a_{00}a_{10}a_{20} + a_{00}a_{01}a_{10}a_{20} + a_{11}a_{20} + a_{00}a_{10}a_{11}a_{20} + a_{01}a_{10}a_{11}a_{20} + a_{00}a_{21} + a_{00}a_{01}a_{10}a_{21} + \\ &\quad a_{11}a_{21} + a_{00}a_{10}a_{11}a_{21} + a_{00}a_{10}a_{20}a_{21} + a_{01}a_{10}a_{20}a_{21} + a_{00}a_{01}a_{10}a_{20}a_{21} + a_{11}a_{20}a_{21} \\ inv_{21} &= a_{10} + a_{11} + a_{00}a_{11} + a_{01}a_{11} + a_{00}a_{01}a_{11} + a_{00}a_{20} + a_{10}a_{20} + a_{00}a_{10}a_{20} + a_{00}a_{01}a_{10}a_{20} + \\ &\quad a_{00}a_{01}a_{11}a_{20} + a_{00}a_{10}a_{11}a_{20} + a_{01}a_{21} + a_{10}a_{21} + a_{00}a_{01}a_{10}a_{21} + a_{01}a_{11}a_{21} + a_{00}a_{01}a_{11}a_{21} + \\ &\quad a_{01}a_{10}a_{11}a_{21} + a_{10}a_{20}a_{21} + a_{00}a_{10}a_{20}a_{21} + a_{01}a_{10}a_{20}a_{21} + a_{00}a_{01}a_{10}a_{20}a_{21} + a_{00}a_{11}a_{20}a_{21} + \\ &\quad a_{01}a_{11}a_{20}a_{21} + a_{00}a_{01}a_{11}a_{20}a_{21}. \end{aligned}$$

Przyjmując reprezentację liczb identyczną jak w rozdziale 4.1, za pomocą powyżej przedstawionych równań można wygenerować permutację o następującej postaci

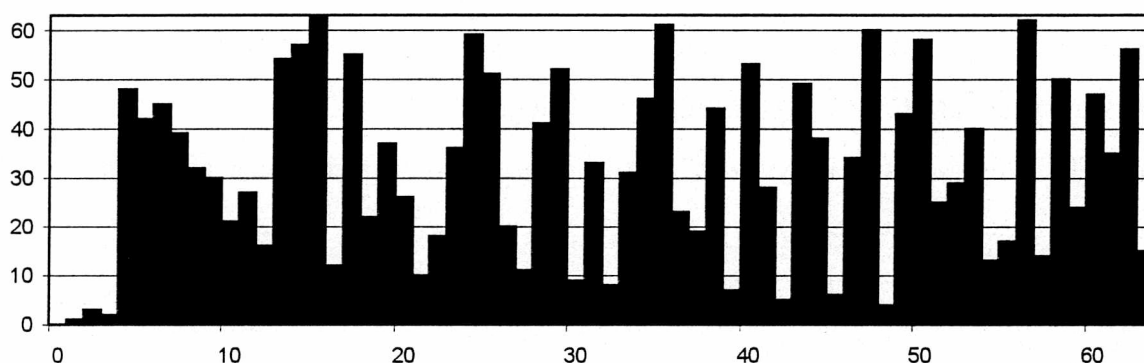
(0)(1)(2 3)(4 48)(5 42)(6 45)(7 39)(8 32)(9 30)(10 21)(11 27)(12 16)
 (13 54)(14 57)(15 63)(17 55)(18 22)(19 37)(20 26)(23 36)(24 59)(25 51)
 (28 41)(29 52)(31 33)(34 46)(35 61)(38 44)(40 53)(43 49)(47 60)(50 58)
 (56 62)

Rys. 4.13. Permutacja z przykładu 4.2

W otrzymanym przekształceniu elementy zbioru liczb odwzorowywane są wzajemnie parami na siebie, czyli podobnie jak w rozdziale 4.2 generowana jest permutacja samoodwracalna 64 - elementowego zbioru. Zawiera ona 31 rozłącznych transpozycji oraz liczby 1 i 0 przechodzącymi same na siebie.

Złożoność równań opisujących powyżej przedstawione przekształcenie jest niemal identyczna jak uzyskanych w przypadku układu do inwersji w $GF(2^m)$. Stąd wniosek, że ten sposób nie pozwala na znaczącą redukcję układów do realizacji opisanego działania.

Z histogramu zamieszczonego na rys. 4.14 można zobaczyć, że uzyskana permutacja ma wyraźnie nieliniowy charakter, bez regularności w występujących transpozycjach oraz bez wyraźnie zarysowanych trendów liniowych.



Rys. 4.14. Histogram permutacji z przykładu 4.2.

Przedstawiona metoda projektowania układów do inwersji w ciałach Galois $GF(q^c)$, $q = 2^k$, $kc = m$ posiada jednak dwie niewątpliwe zalety w stosunku do przedstawionej w rozdziale 4.2. Pierwszą z nich jest zmniejszenie złożoności zadań obliczeniowych służących do zaprojektowania układu do postaci $O(k^2) + O(c^2)$.

Druga, bardzo istotna w zastosowaniach kryptograficznych, to zwiększenie liczby dostępnych wielomianów nieredukowalnych służących do konstrukcji ciał Galois, a więc postaci równań generujących permutację.

5. Układy do podnoszenia do kwadratu i pierwiastkowania kwadratowego w $GF(2^m)$ jako uniwersalne generatory permutacji o charakterze liniowym

W rozdziale przedstawiono metodę generowania permutacji zbiorów 2^m -elementowych za pomocą podnoszenia do kwadratu i pierwiastkowania w ciałach Galois charakterystyki 2. Zaprezentowano także sposób uzyskania permutacji samoodwracalnej zbioru 2^m -elementowego dla parzystej liczby m , przez wielokrotne podnoszenie do kwadratu.

5.1 Podnoszenie do kwadratu i pierwiastkowanie kwadratowe w ciałach Galois charakterystyki 2

Podnoszenie do kwadratu, jako szczególny przypadek mnożenia jest bardzo użytecznym działaniem w arytmetyce związanej z ciałami Galois charakterystyki 2. W tego typu strukturach algebraicznych jest działaniem o charakterze liniowym, dlatego posiada szereg zalet decydujących o jego przydatności w realizacjach sprzętowych. Dzięki prostocie podnoszenia do kwadratu i pierwiastkowania kwadratowego w $GF(2^m)$, operacje te są szczególnie przydatne w urządzeniach o architekturze równoległej.

Jeśli rozważy się ciało $GF(2^m)$ utworzone za pomocą wielomianu nieredukowalnego $P(x)$ z bazą wielomianową, to oznaczając dowolny element $GF(2^m)$ jako $A(x) = a_0 + a_1x + \dots + a_{m-1}x^{m-1}$, jego kwadrat $K(x) = k_0 + k_1x + \dots + k_{m-1}x^{m-1}$, otrzyma się równanie

$$K(x) = A^2(x) \bmod P(x). \quad (5.1)$$

Równanie (5.1) można przedstawić w postaci macierzowej jako

$$K = Q \cdot A, \quad (5.2)$$

czyli

$$\begin{bmatrix} k_0 \\ k_1 \\ \vdots \\ k_{m-1} \end{bmatrix} = Q \cdot \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{m-1} \end{bmatrix}. \quad (5.3)$$

Przyjmując, że zbiór elementów

$$\beta^i = [b_{i,0} \quad b_{i,1} \quad \cdots \quad b_{i,m-1}], \quad (5.4)$$

gdzie

$$x^i \equiv \sum_{k=1}^m b_{i,m-k} \cdot x^{m-k} \pmod{P(x)}, \quad b_{i,j} \in GF(2), \quad (5.5)$$

stanowi bazę wielomianową $GF(2^m)$ nad $GF(2)$ można stosunkowo łatwo wyznaczyć macierz Q o wymiarach $m \times m$, bowiem jej kolumny utworzone są przez elementy $\beta^{2^i}, i = 0, 1, \dots, m-1$ tej bazy [Koś97]. Wobec tego

$$Q = \begin{bmatrix} b_{0,0} & b_{2,0} & \cdots & b_{2^{m-2},0} \\ b_{0,1} & b_{2,1} & \cdots & b_{2^{m-2},1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{0,m-1} & b_{2,m-1} & \cdots & b_{2^{m-2},m-1} \end{bmatrix}. \quad (5.6)$$

Przyjęcie zależności (5.6) pozwala na znaczne uproszczenie obliczeń prowadzących do otrzymania ostatecznych wzorów opisujących podnoszenie do kwadratu w $GF(2^m)$.

Pierwiastek kwadratowy $R(x) = r_0 + r_1 x + \dots + r_{m-1} x^{m-1}$ dowolnego elementu ciała $A(x) \in GF(2^m)$ jest także elementem tego ciała.

$$R(x) = \sqrt{A(x)} \pmod{P(x)}. \quad (5.7)$$

Ponieważ pierwiastkowanie kwadratowe jest operacją odwrotną do podnoszenia do kwadratu, z równania (5.2) możemy wywnioskować, że prawdziwa jest zależność

$$R = Q^{-1} \cdot A, \quad (5.8)$$

czyli

$$\begin{bmatrix} r_0 \\ r_1 \\ \vdots \\ r_{m-1} \end{bmatrix} = Q^{-1} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{m-1} \end{bmatrix}, \quad (5.9)$$

gdzie

Q^{-1} - odwrotność binarnej macierzy przekształcenia z wzoru (5.6).

Czyli aby otrzymać zależności definiujące pierwiastek kwadratowy w $GF(2^m)$ należy obliczyć macierz Q związaną z podnoszeniem do kwadratu w tym ciele a następnie wyznaczyć jej odwrotność.

5.2 Wzajemnie odwracalne permutacje zbioru 2^m - elementowego generowane za pomocą potęgowania w $GF(2^m)$ z wykładnikami 2^k i 2^{-k}

Potęgowanie z wykładnikiem 2^k lub 2^{-k} , $k = 1, 2, 3, \dots$ jest wielokrotnym podnoszeniem do kwadratu lub obliczaniem pierwiastka kwadratowego. Prostota realizacji tych obliczeń w ciałach $GF(2^m)$ powoduje, że wyznaczanie zależności dla kolejnych wartości k jest niezwykle proste i nie zmienia w znaczącym stopniu złożoności wzorów obliczeniowych. Jednocześnie uzyskuje się w ten sposób kolejne układy generujące różne permutacje różniące się od siebie właściwościami. Dla dowolnego elementu $A(x) \in GF(2^m)$ operacje potęgowania w wykładnikami 2^k i 2^{-k} są wzajemnie odwracalne. Wobec tego takie przekształcenia generują wzajemnie odwracalne permutacje zbioru 2^m - elementowego.

PRZYKŁAD 5.1 Dane jest ciało $GF(2^8)$ o wielomianie nieredukowalnym $P(x) = 1 + x^2 + x^3 + x^7 + x^8$. Postać dowolnego elementu ciała jest następująca $A(x) = a_0 + a_1 x + \dots + a_7 x^7$. Macierz Q do obliczenia kwadratu elementu tego ciała wyznaczona na podstawie (5.6) wynosi

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}, \quad (5.10)$$

czyli

$$\begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \\ k_4 \\ k_5 \\ k_6 \\ k_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix}, \quad (5.11)$$

co po pomnożeniu daje następujące zależności

$$\begin{aligned} k_0 &= a_0 + a_4 + a_5 + a_6, \\ k_1 &= a_5 + a_6, \\ k_2 &= a_1 + a_4 + a_7, \\ k_3 &= a_4 + a_6 + a_7, \\ k_4 &= a_2 + a_6, \\ k_5 &= a_5 + a_7, \\ k_6 &= a_3 + a_7, \\ k_7 &= a_4 + a_5. \end{aligned} \tag{5.12}$$

Układ ten można zrealizować za pomocą 12 bramek XOR. Przyjmując, podobnie jak w poprzednich rozdziałach, reprezentację liczb w postaci

$$x_N = \sum_{j=1}^8 a_{8-j} \cdot 2^{8-j}, \quad y_N = \sum_{j=1}^8 k_{8-j} \cdot 2^{8-j}, \tag{5.13}$$

można się przekonać, że za pomocą (5.12) można wygenerować permutację

(0)(1)(2 4 16 141 61 127 96 184)(3 5 17 140 60 126 97 185)(6 20 157 176 66 31 216 186)(7
21 156 177 67 30 217 187)(8 64 27 200 55 59 107 253)(9 65 26 201 54 58 106 252)(10 68 11
69)(12 80 150 245 72 91 211 255)(13 81 151 244 73 90 210 254)(14 84 134 120 117 36 179
71)(15 85 135 121 116 37 178 70)(18 137 45 242 93 199 98 188)(19 136 44 243 92 198 99
189)(22 153 160 207 34 167 218 190)(23 152 161 206 35 166 219 191)(24 205 38 183 87
131 105 249)(25 204 39 182 86 130 104 248)(28 221 171 138 40 227 209 251)(29 220 170
139 41 226 208 250)(32 163 202 51 43 230 192 119)(33 162 203 50 42 231 193 118)(46 247
76 75 94 194 115 48)(47 246 77 74 95 195 114 49)(52 62 122 113)(53 63 123 112)(56 110
236 132 124 101 169 142)(57 111 237 133 125 100 168 143)(78 79)(82 146 229 197 102 172
159 180)(83 147 228 196 103 173 158 181)(88 214 238 128 108 232 148 241)(89 215 239
129 109 233 149 240)(144 225 213 235 145 224 212 234)(154 165 222 174 155 164 223
175)

Rys. 5.1. Permutacja opisana równaniami (5.12)

Odwracając macierz Q przekształcenia występującą we wzorze (5.12) otrzymujemy

$$Q^{-1} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (5.14)$$

co pozwala uzyskać równania do obliczenia pierwiastka kwadratowego w tym ciele. Mają one postać

$$\begin{aligned} r_0 &= a_0 + a_3 + a_5, \\ r_1 &= a_2 + a_5 + a_7, \\ r_2 &= a_3 + a_4 + a_5 + a_7, \\ r_3 &= a_1 + a_3 + a_6 + a_7, \\ r_4 &= a_1 + a_3 + a_5, \\ r_5 &= a_1 + a_3 + a_7 + a_7, \\ r_6 &= a_3 + a_5 + a_7, \\ r_7 &= a_1 + a_3 + a_7. \end{aligned} \quad (5.15)$$

Równania te można zrealizować za pomocą 19 dwuwejściowych układów XOR. Przyjmując reprezentację liczb

$$x_N = \sum_{j=1}^8 a_{8-j} \cdot 2^{8-j}, \quad y_N = \sum_{j=1}^8 r_{8-j} \cdot 2^{8-j} \quad (5.16)$$

otrzymujemy permutację

(0)(1)(2 184 96 127 61 141 16 4)(3 185 97 126 60 140 17 5)(6 186 216 31 66 176 157 20)(7 187 217 30 67 177 156 21)(8 253 107 59 55 200 27 64)(9 252 106 58 54 201 26 65)(10 69 11 68)(12 255 211 91 72 245 150 80)(13 254 210 90 73 244 151 81)(14 71 179 36 117 120 134 84)(15 70 178 37 116 121 135 85)(18 188 98 199 93 242 45 137)(19 189 99 198 92 243 44 136)(22 190 218 167 34 207 160 153)(23 191 219 166 35 206 161 152)(24 249 105 131 87 183 38 205)(25 248 104 130 86 182 39 204)(28 251 209 227 40 138 171 221)(29 250 208 226 41 139 170 220)(32 119 192 230 43 51 202 163)(33 118 193 231 42 50 203 162)(46 48 115 194 94 75 76 247)(47 49 114 195 95 74 77 246)(52 113 122 62)(53 112 123 63)(56 142 169 101 124 132 236 110)(57 143 168 100 125 133 237 111)(78 79)(82 180 159 172 102 197 229 146)(83 181 158 173 103 196 228 147)(88 241 148 232 108 128 238 214)(89 240 149

233 109 129 239 215)(144 234 212 224 145 235 213 225)(154 175 223 164 155 174 222
165)

Rys. 5.2. Permutacja opisana wzorami (5.15)

Porównując permutacje z rys.5.1 i 5.2 łatwo można dostrzec przykłady wzajemnie odwrotnych odwzorowań np. $2 \rightarrow 184$ w 5.1, $184 \rightarrow 2$ itd. co wyraźnie wskazuje na wzajemnie odwracalny charakter użytych przekształceń.

Mając wyznaczoną postać równań opisujących podnoszenie do kwadratu i pierwiastkowanie kwadratowe w danym ciele $GF(2^m)$, określenie zależności pozwalających na obliczenie potęgi elementu $A^{2^k}(x)$ jest równie łatwe jak podniesienie do k - tej potęgi (k - krotne pomnożenie) $k = 2, 3, \dots, m-1$, macierzy Q lub jej odwrotności w arytmetyce $GF(2)$ czyli

$$A^{2^k}(x) = K^{2^{k-1}}(x) = Q^k \cdot A. \quad (5.17)$$

Dowód :

Z wzoru (5.2) mamy

$$A^2 = K = Q \cdot A,$$

czyli

$$\begin{aligned} A^{2^1} &= K = Q \cdot A, \\ A^{2^2} &= K^2 = Q \cdot K = Q \cdot (Q \cdot A), \\ A^{2^3} &= K^{2^2} = Q \cdot K^2 = Q \cdot (Q \cdot K) = Q \cdot (Q \cdot (Q \cdot P)), \\ &\vdots \\ &\vdots \\ &\vdots \\ A^{2^k} &= K^{2^{k-1}} = Q \cdot (Q \cdot (Q \cdot (\dots (Q \cdot P) \dots)) = Q^k \cdot A. \end{aligned} \quad (5.18)$$

c . b . d . o.

Stosując powyższy wzór do wartości z przykładu 5.1 otrzymuje się następujące k - te potęgi macierzy Q z wzoru (5.8)

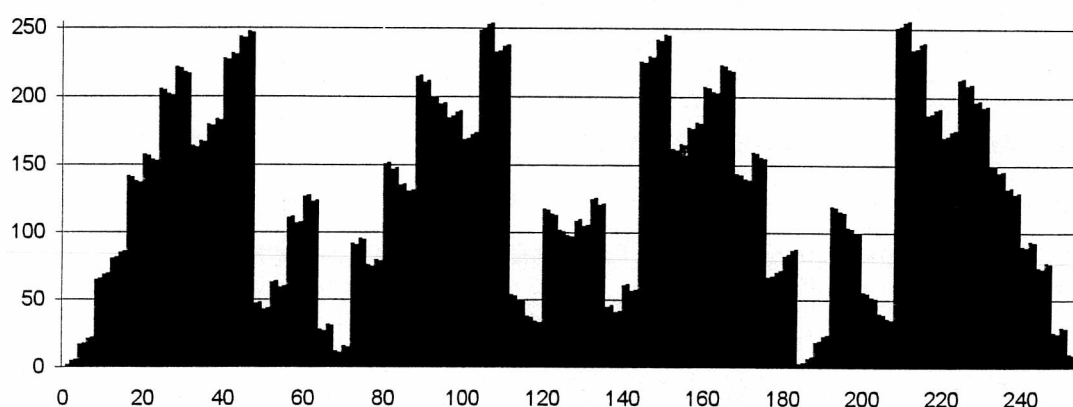
$$\begin{aligned}
 & \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}^2 = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \\
 & \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}^3 = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \dots \text{itd.}
 \end{aligned} \tag{5.19}$$

Wyznaczenia zależności dotyczących k -krotnego pierwiastkowania kwadratowego można dokonać poprzez obliczenie odwrotności otrzymanej macierzy k -krotnego podnoszenia do kwadratu lub podobnie jak we wzorze (5.19) poprzez k -tą potęgę macierzy opisującej pierwiastkowanie kwadratowe.

W celu zaprojektowania układu generującego wzajemnie odwracalne permutacje należy wyznaczyć dwa układy równań generujących k -krotny kwadrat K^k i k -krotny pierwiastek kwadratowy R^k elementu $A(x) \in GF(2^m)$ o własności

$$R^k(K^k(A(x))) = A(x). \tag{5.20}$$

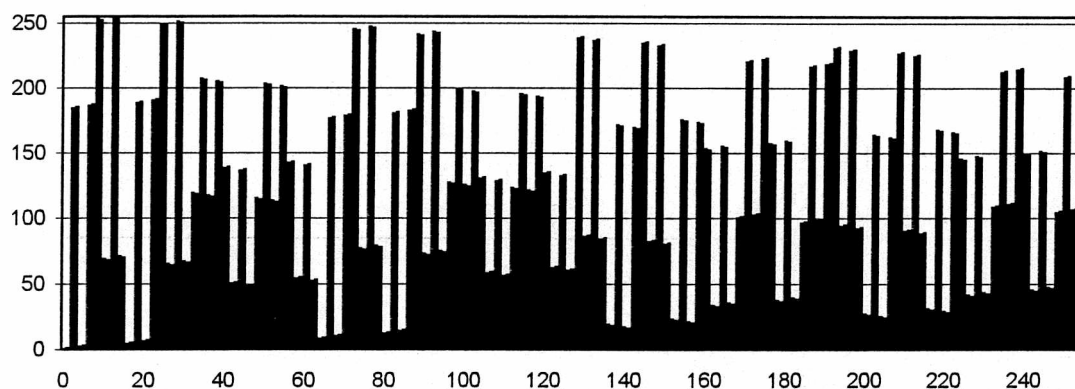
Tak wyznaczone układy generują permutacje o charakterze liniowym zawierające cykle o parzystej długości. Charakteryzują je występujące w podcyklach krótkie liniowo rosnące lub malejące sekwencje liczbowe. Łatwo to zaobserwować z histogramu permutacji z rys.5.1 przedstawionego na rys.5.3.



Rys. 5.3. Histogram permutacji z rys.5.1.

Dodatkowo z histogramu widać także regularność polegającą na tym, że kształt elementów histogramu dla wartości z zakresu $128 \div 255$ jest jakby obrazem symetrii elementów z zakresu $0 \div 127$ z osią przechodzącą przez środek zakresu wartości.

Te same cechy można zaobserwować w przekształceniu opartym na pierwiastkowaniu kwadratowym, co pokazuje histogram zamieszczony na rys.5.4.



Rys.5.4. Histogram permutacji z rys.5.2.

Złożoność opisywanych układów równań nie przekracza $O(m)$, zaś ilość bramek XOR, za pomocą których można je zrealizować jest mniejsza niż $\frac{m(m-1)}{2}$.

5.3 Generowanie permutacji samoodwracalnych zbioru 2^m - elementowego za pomocą układów do wielokrotnego podnoszenia do kwadratu w $GF(2^m)$, m - liczba parzysta

Dla każdego elementu $\alpha \in GF(q^m)$ spełniona jest zawsze zależność

$$\alpha^{q^m} = \alpha \quad (5.21)$$

wobec tego dla $\alpha \in GF(2^m)$ prawdziwe jest

$$\alpha^{2^m} = \alpha, \quad (5.22)$$

co można zapisać także jako

$$(\alpha^{2^{\frac{m}{2}}})^{2^{\frac{m}{2}}} = \alpha, \quad (5.23)$$

czyli

$$\forall x_0, y_0 \in GF(2^m), (x_0)^{2^{\frac{m}{2}}} = y_0 \Rightarrow (y_0)^{2^{\frac{m}{2}}} = ((x_0)^{2^{\frac{m}{2}}})^{2^{\frac{m}{2}}} = x_0. \quad (5.24)$$

Wobec tego spełniony jest warunek samoodwracalności przekształcenia permutacyjnego podany we wzorze (4.1) rozdz.4.

PRZYKŁAD 5.2 W tym przykładzie rozważa się takie samo ciało, jak w przykładzie 5.1. Wobec tego potęgowanie z wykładnikiem 2^4 będzie przekształceniem, dającym permutację samoodwracalną, ponieważ zgodnie z zasadami opisanymi poprzednio, oblicza się

$$Q^4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}^4 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (5.25)$$

biorąc pod uwagę układ równań

$$\begin{bmatrix} k_0 \\ k_1 \\ \vdots \\ k_{m-1} \end{bmatrix} = Q^4 \cdot \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{m-1} \end{bmatrix}, \quad (5.26)$$

otrzyma się

$$\begin{aligned}
 k_0^4 &= a_0 + a_1 + a_2 + a_3 + a_5 + a_6 + a_7, \\
 k_1^4 &= a_2 + a_3 + a_5 + a_6, \\
 k_2^4 &= a_1 + a_2 + a_3, \\
 k_3^4 &= a_1 + a_2 + a_5 + a_6, \\
 k_4^4 &= a_1 + a_2 + a_3 + a_6 + a_7, \\
 k_5^4 &= a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7, \\
 k_6^4 &= a_2 + a_4 + a_7, \\
 k_7^4 &= a_7.
 \end{aligned} \tag{5.27}$$

W powyższym układzie równań symbol k_i^4 oznacza wartość i - tego bitu wyjściowego układu do podnoszenia do 4 potęgi.

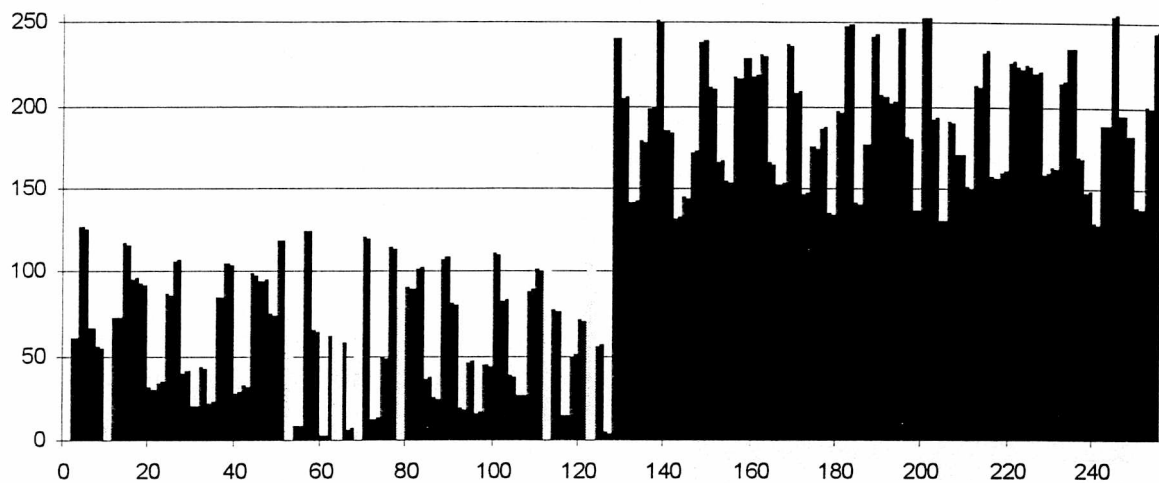
Przyjmując reprezentację liczb tak jak we wzorze (5.11) otrzymuje się permutację
 (0)(1)(2 61)(3 60)(4 127)(5 126)(6 66)(7 67)(8 55)(9 54)(10)(11)(12 72)(13 73)(14 117)(15
 116)(16 96)(17 97)(18 93)(19 92)(20 31)(21 30)(22 34)(23 35)(24 87)(25 86)(26 106)(27
 107)(28 40)(29 41)(32 43)(33 42)(36 84)(37 85)(38 105)(39 104)(44 99)(45 98)(46 94)(47
 95)(48 75)(49 74)(50 118)(51 119)(52)(53)(56 124)(57 125)(58 65)(59 64)(62)(63)(68)(69)
 (70 121)(71 120)(76 115)(77 114)(78)(79)(80 91)(81 90)(82 102)(83 103)(88 108)(89
 109)(100 111)(101 110)(112)(113)(122)(123)(128 241)(129 240)(130 204)(131 205)(132
 142)(133 143)(134 179)(135 178)(136 198)(137 199)(138 251)(139 250)(140 185)(141
 184)(144 145)(146 172)(147 173)(148 238)(149 239)(150 211)(151 210)(152 166)(153
 167)(154 155)(156 217)(157 216)(158 228)(159 229)(160 218)(161 219)(162 231)(163
 230)(164 165)(168 237)(169 236)(170 208)(171 209)(174 175)(176 186)(177 187)(180
 197)(181 196)(182 248)(183 249)(188 242)(189 243)(190 207)(191 206)(192 202)(193
 203)(194 247)(195 246)(200 253)(201 252)(212 213)(214 232)(215 233)(220 226)(221
 227)(222 223)(224 225)(234 235)(244 254)(245 255)

Rys. 5.5. Permutacja opisana wzorami (5.27)

Jak widać otrzymano permutację samoodwracalną złożoną ze 121 transpozycji i liczb 0, 1, 10, 11, 52, 53, 62, 63, 68, 69, 112, 113, 122, 123, które w przekształceniu przechodzą same na siebie.

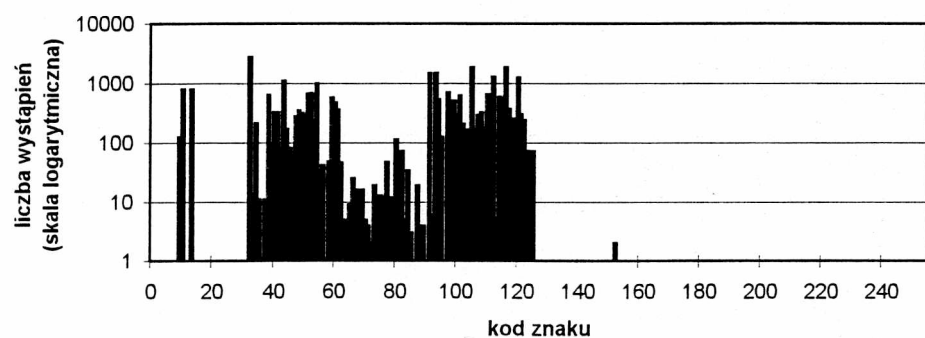
Widoczny jest w permutacji liniowy charakter przekształcenia. Wyraźnie wskazują na to sekwencje par liczbowych występujących w transpozycjach np. ... (214 232)(215 233) ...,

oraz podcykli jednoelementowych np. 0,1 ... 52, 53 ... itd. Te cechy permutacji pokazuje także zamieszczony na rys.5.5 histogram.

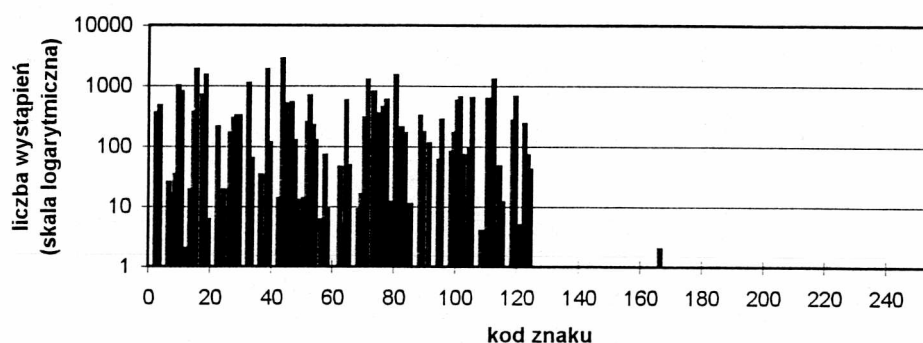


Rys. 5.6. Histogram permutacji z rys.5.5.

Na histogramie z rys. 5.6 widać cechę przekształcenia, polegającą na ograniczeniu zakresu wartości obrazów do zakresu wartości odwzorowywanych, czyli liczby 0÷127 przekształcane są na liczby o wartościach mniejszych niż 128, zaś liczby z zakresu 128÷255 na wartości większe niż 127. Jest to spowodowane tym, że w wyniku realizacji przekształcenia nie zmienia się wartość najbardziej znaczącego bitu. Wpływ takiej cechy przekształcenia na działanie prostego szyfru podstawieniowego stosującego opisywane przekształcenie można zaobserwować na histogramach zamieszczonych na rys.5.7 i 5.8.



Rys. 5.7 Rozkład kodów ASCII w pliku przed zaszyfrowaniem



Rys. 5.8 Rozkład kodów ASCII w pliku po zaszyfrowaniu

Wyraźnie widać, że wartości występujące w kryptogramie są zgrupowane niemal w identycznych zakresach, jak w tekście źródłowym.

Układ równań przedstawiony we wzorze (5.27) można zrealizować za pomocą mniej niż 26 bramek XOR, co stanowi potwierdzenie tezy o małej złożoności układów do wielokrotnego podnoszenia do kwadratu w układach realizujących działania zgodnie z zasadami arytmetyki w $GF(2^m)$.

Mimo jednoznacznie liniowego charakteru odwzorowań występujących w permutacjach opisanych powyżej oraz innych niekorzystnych z punktu widzenia kryptografii cech przekształcenia nadają się do zastosowania w systemach kryptograficznych, ze względu na małą złożoność układów oraz prostotę ich projektowania. Wydaje się jednak celowe stosowanie ich wyłącznie w zestawach układowych generujących permutacje o różnym charakterze, zacierających zaprezentowane niekorzystne charakterystyki.

6. Hybrydowa metoda generowania par permutacji wzajemnie odwracalnych i permutacji samoodwracalnych zbioru 2^n - elementowego

Generowanie permutacji dla celów kryptograficznych dużych zbiorów 2^n - elementowych za pomocą pojedynczych układów do rachowania w ciałach Galois charakterystyki 2 jest zadaniem bardzo złożonym. Szczególnie dotyczy to działań o charakterze nieliniowym, które opisuje się równaniami o dużej złożoności, co czasami wręcz uniemożliwia ich praktyczne zastosowanie.

Innym problemem jest uzyskiwanie takich permutacji, które charakteryzują się określonymi, użytecznymi w kryptografii cechami oraz dają się łatwo realizować w postaci sprzętowej. W tym przypadku często w praktyce niemożliwe jest dobranie takiego pojedynczego przekształcenia, które generowałoby permutację, odpowiadającą określonym wymaganiom.

Rozwiązaniem tego problemu jest generowanie takiej permutacji P , która byłaby złożeniem co najmniej dwóch permutacji p_i działających na podzbiorach elementów permutowanego przez P zbioru, czyli

$$P = p_0 p_1 p_2 \dots p_i \dots p_{m-1}, \quad (6.1)$$

gdzie

P - permutacja zbioru 2^n - elementowego,

p_i - permutacja działająca na 2^{k_i} - elementowym podzbiorze zbioru 2^n - elementowego.

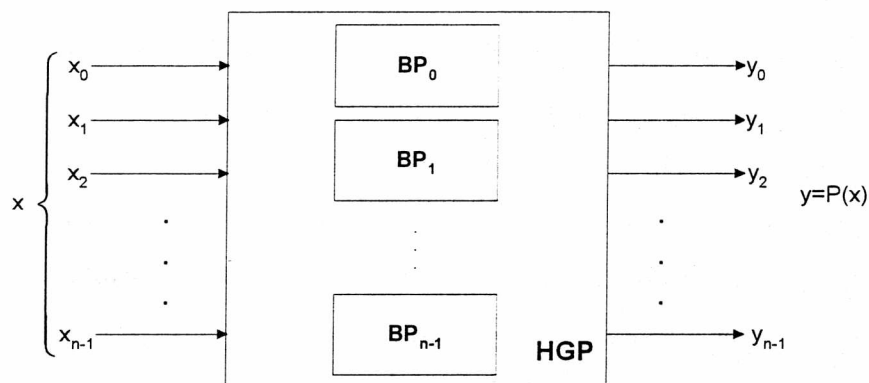
Przy czym $k_i, n, i \in N$, oraz $k_i \leq n$.

Ten sposób nazwano hybrydową metodą generowania permutacji. Zagadnienie powyższe sprowadzono w pracy do przetwarzania n - bitowych reprezentacji elementów permutowanego zbioru za pomocą połączonych w odpowiedni sposób układów, realizujących różne zdefiniowane w ciałach Galois charakterystyki 2 przekształcenia permutacyjne, działające na elementach, reprezentowanych za pomocą n lub mniej bitów. W niniejszym rozdziale przedstawiono sposoby generowania permutacji metodą hybrydową oraz zaprezentowano wyniki testów, polegających na składaniu różnych permutacji poprzez łączenie układów, generujących przekształcenia opisane w przykładach zaprezentowanych w poprzednich rozdziałach.

W celu osiągnięcia dużej czytelności przykładów prezentowane problemy zilustrowano hybrydowymi generatorami permutacji zbioru 256-elementowego.

6.1 Założenia i definicje

W ramach rozważań nad strukturami złożonymi z wielu różnych, połączonych ze sobą generatorów permutacji zastosowano szereg określeń, których znaczenie wymaga precyzyjnego wyjaśnienia. Podstawowym elementem analizowanym w niniejszym rozdziale, jest **hybrydowy generator permutacji HGP**¹, czyli układ generujący permutację metodą hybrydową poprzez przetwarzanie n - bitowego bloku danych. Układ ten złożony jest z mniejszych elementów zwanych dalej **blokami permutacyjnymi BP** lub generatorami permutacji albo po prostu generatorami.



Rys. 6.1 Idea hybrydowego generatora permutacji n - bitowego bloku danych.

Zgodnie z rys.6.1 hybrydowy generator permutacji przetwarza n -bitową wartość x w n -bitową wartość $y = P(x)$, gdzie $P(x)$ - przekształcenie permutacyjne. Strukturę HGP stanowi system połączonych ze sobą, na różne sposoby, bloków permutacyjnych BP. Warto przy tym zauważyć, że na przykład sekwencyjne łączenie bloków permutacyjnych równoważne jest składaniu (mnożeniu) permutacji.

Ze względu na brak ustalonych zasad opisu tego typu problemów, w dalszym ciągu niniejszej pracy zastosowano także własną konwencję zapisu charakteryzującego poszczególne bloki permutacyjne. Ogólna postać formuły opisującej pojedynczy BP jest następująca

¹ W następnych rozdziałach taki hybrydowy generator permutacji nazywany będzie także skrzynką podstawieniową lub s-skrzynką (z ang. substitution box)

$$\mathbb{F}_{r_{\min} : r_{\max}}^q \langle \{p(x)\}; \{A\} \rangle, \quad (6.2)$$

gdzie

$\mathbb{F}$ - przekształcenie zastosowane do generowania permutacji,

q - liczba elementów ciała $GF(q^n)$,

$p(x)$ - wielomian nieredukowalny stopnia n nad $GF(q)$, służący do skonstruowania ciała $GF(q^n)$,

A - inne parametry charakteryzujące zastosowane operacje matematyczne,

$r_{\min} : r_{\max}$ - zakresy bitów, przetwarzanych przez układ.

Umieszczenie we wzorze (6.2) parametru w nawiasach $\{\}$ oznacza, że wystąpienie jest opcjonalne i zależy od typu opisywanego przekształcenia. Na przykład, dla generatorów, stosujących przekształcenie afiniczne nad $GF(2)$, opisanych w rozdz. 3.2, parametrami wzoru (6.2) są macierze S i S^{-1} , oraz wielomian nieredukowalny. Oczywiście jest przy tym, że w celu pełnego określenia wykorzystanego przekształcenia, postaci tych macierzy powinny być przedstawione dodatkowo, poza formułą opisującą generator permutacji.

Przyjmując taki sposób opisu, można zauważyć, że przedstawione w poprzednich rozdziałach pracy generatory permutacji można opisać stosując następujące symbole:

A - przekształcenie afiniczne $\pi(x) = xS + c$,

A^{-1} - odwrotne przekształcenie afiniczne $\pi^{-1}(y) = (y + c)S^{-1}$,

I - inwersja w ciele Galois charakterystyki 2,

$\mathbb{K}^*$ - k - krotne podnoszenie do kwadratu,

$\mathbb{K}^k$ - k - krotne pierwiastkowanie kwadratowe,

Dodatkowo wprowadzono także funkcję² $\mathbb{X}$ oznaczającą dodawanie modulo 2 n -bitowej liczby binarnej.

PRZYKŁAD 6.1 Zgodnie z powyższym opisem 6-bitowy generator permutacji BP_1 , zrealizowany w oparciu o równania przedstawione w przykładzie 4.1 można przedstawić następująco

$$\mathbb{I}_{0:5}^2 \langle x^6 + x^3 + 1 \rangle. \quad (6.3)$$

² Wprowadzenie tej funkcji jest bardzo istotne z punktu widzenia analizy wpływu poszczególnych składników generatora hybrydowego na działanie całości dotyczy to np. przekształcenia afinicznego nad $GF(2)$, co zostanie wyjaśnione w dalszej części niniejszego rozdziału.

Składanie generatorów permutacji opisywane jest znakiem $\mapsto$, oznaczającym jednocześnie kolejność wystąpień poszczególnych układów w procesie przetwarzania.

PRZYKŁAD 6.2 Niech dany będzie hybrydowy generator permutacji HGP_1 złożony z dwóch bloków permutacyjnych, opisanych odpowiednio w rozdziałach 4.1 i 3.1. Oto formuła definiująca jego strukturę

$$\overset{2}{\underset{0.5}{I}} \langle x^6 + x^3 + 1 \rangle \mapsto \overset{8}{\underset{0.5}{A}} \langle x^3 + x + 1; S, c = 0 \rangle, \quad (6.4)$$

Działanie generatora hybrydowego opisanego powyżej łatwo można zinterpretować jako przekształcenie 6-bitowego bloku danych, najpierw przez generator permutacji obliczający inwersję w ciele $GF(2^6)$ o wielomianie nieredukowalnym $p(x) = x^6 + x^3 + 1$ (rozd. 4.1). Następnie blok ten przetwarzany jest za pomocą układu, realizującego przekształcenie afiniczne w ciele $GF(8)$ z wartością parametru $c = 0$ i wielomianem nieredukowalnym $p(x) = x^3 + x + 1$ służącym do konwersji z $GF(8)$ na $GF(2)$ (rozd. 3.1).

Rozważane w dalszym ciągu niniejszej pracy przykłady odnosić się będą niemal wyłącznie do układów, przedstawionych w poprzednich rozdziałach przy zapisie, prezentującym strukturę generatora hybrydowego, występujące oznaczenia zostaną tak zredukowane, aby pozwalały na jednoznaczna identyfikację zastosowanego przekształcenia. Np. zapis przedstawiony w przykładzie 6.2 w takiej skróconej postaci ma postać

$\overset{2}{\underset{0.5}{I}} \mapsto \overset{8}{\underset{0.5}{A}} \langle c = 0 \rangle$ lub w postaci ogólnej, przy wcześniej zdefiniowanych funkcjach

opisujących kolejne BP, jako $\overset{2}{\underset{0.5}{BP}}_1 \mapsto \overset{8}{\underset{0.5}{BP}}_2$.

6.2 Łączenie bloków permutacyjnych

Zagadnienie łączenia bloków permutacyjnych zawiera szereg aspektów, które muszą być wzięte pod uwagę w trakcie prac projektowych nad układami hybrydowymi. Niektóre z nich muszą być bezwzględnie uwzględnione, inne zaś zależą od specyfiki zastosowania.

6.2.1 Kolejność bloków permutacyjnych

Podstawowym wymaganiem wobec wzajemnie odwracalnych HGP, czyli pary HGP i HGP^{-1} , jest zachowanie kolejności bloków permutacyjnych, generujących permutację. Wynika to bezpośrednio z nieprzemienności działań przy składaniu permutacji. Aby uzyskać efekt odwrócenia przekształcenia, powstałego w wyniku działania dowolnego HGP, przekształcenia występujące w HGP_i^{-1} muszą występować w kolejności odwrotnej do sposobu generowania zastosowanego w HGP_i . Wobec tego, dla każdego HGP złożonego np. z n bloków permutacyjnych

$$F_0 \mapsto F_1 \mapsto F_2 \mapsto \dots \mapsto F_{n-1}, \quad (6.5)$$

struktura generatora realizującego przekształcenie odwrotne musi być następująca

$$(F_{n-1})^{-1} \mapsto (F_{n-2})^{-1} \mapsto \dots \mapsto (F_2)^{-1} \mapsto (F_1)^{-1} \mapsto (F_0)^{-1}. \quad (6.6)$$

Powyższe zależności dotyczą oczywiście tylko kolejnych BP, przetwarzających te same pozycje bitowe.

PRZYKŁAD 6.3 Niech dany będzie 16-bitowy HGP₁ o strukturze $\overset{2}{A}_{0:7} \langle c=0 \rangle \mapsto \overset{2}{I}_{8:13} \mapsto \overset{4}{I}_{10:15}$.

Wobec tego HGP₁⁻¹ można opisać jako $\overset{4}{I}_{10:15} \mapsto \overset{2}{I}_{8:13} \mapsto \overset{2}{A}_{0:7} \langle c=0 \rangle$. Lecz także zapis o postaci

$\overset{2}{A}_{0:7} \langle c=0 \rangle \mapsto \overset{4}{I}_{10:15} \mapsto \overset{2}{I}_{8:13}$ jest poprawny, bowiem 8 najmniej znaczących bitów przetwarzanych

jest tylko przez jeden układ i nie ma większego znaczenia, w którym miejscu opisu HGP₁⁻¹ znajdzie się funkcja charakteryzująca to przekształcenie³.

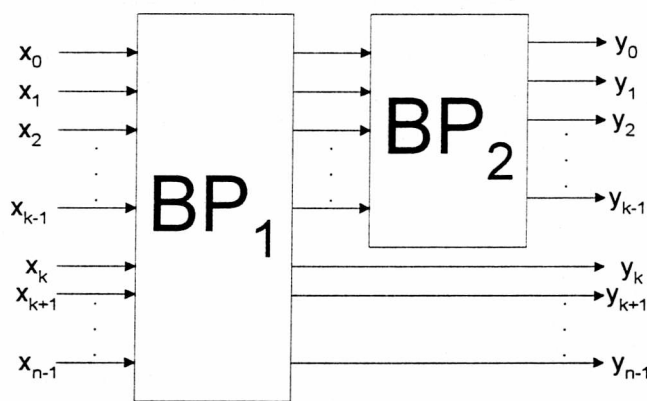
³ Dla większej czytelności zapisu w dalszej części pracy stosowany będzie jedynie wariant zachowujący odwrotną kolejność działań bez względu na to czy można to zapisać inaczej.

6.2.2 Konfiguracje bloków permutacyjnych

Przy łączeniu bloków permutacyjnych istotne jest ich wzajemne położenie względem siebie. Wyróżnić tutaj można dwa podstawowe ustawienia, związane ze sposobem łączenia: sekwencyjne i równoległe.

Połączenie sekwencyjne

Cechą charakterystyczną tego typu połączenia jest to, że pozycje bitowe n - bitowego bloku danych od 0 do $k-1$, gdzie $k \in [0, n-1]$, po przekształceniu przez pierwszy układ BP_1 są modyfikowane przez drugi BP_2 zaś bity $k \div n-1$ po przetworzeniu przez BP_1 pozostają nie zmienione. Ilustruje to rysunek 6.2.



Rys. 6.2 Przykład sekwencyjnego połączenia dwóch bloków permutacyjnych.

W przypadku sekwencyjnego generowania par permutacji wzajemnie odwracalnych istotne jest to, iż zarówno układ zastosowany w generatorze HGP jak i HGP^{-1} należy zwrócić uwagę na przekształcanie tych samych pozycji bitowych oczywiście z uwzględnieniem zasad podanych w 6.2.1. Jest to warunkiem poprawnego generowania wzajemnego odwracalnych permutacji przez obydwa układy.

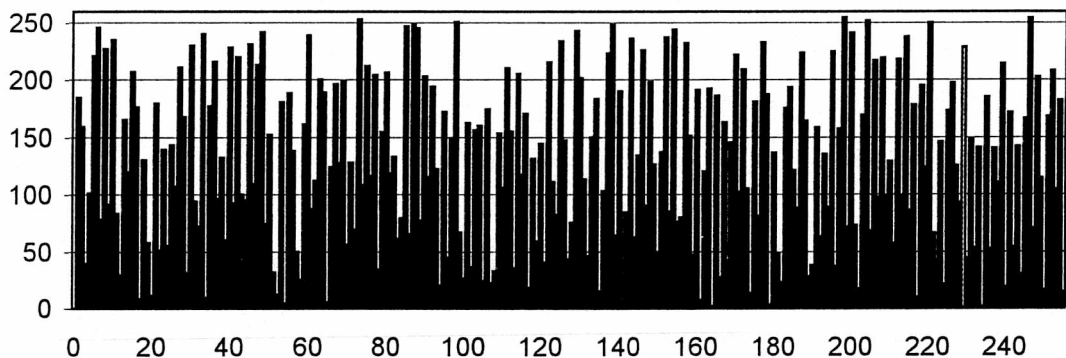
PRZYKŁAD 6.4 Niech dany będzie 8-bitowy HGP_1 o strukturze $\overset{2}{\underset{0:7}{\mathbb{K}}}^4 \mapsto \overset{2}{\underset{2:7}{\mathbb{I}}} \mapsto \overset{2}{\underset{0:7}{\mathbb{A}}}\langle c=0 \rangle$, który generuje permutację

(0)(1 184 193 135 15 207 97 149 126 147 90 203 169 145 134 183 175 181 48 242 54 4 101
 162 120 144 62 112 155 76 116 170 222 42 220 250 16 176 81 118 131 113 35 177 233 141
 6 246 254 182 23 139 64 189 28 167 163 192 63 200 241 171 102 36 216 178 187 224 21
 179 3 39 60 239 214 238 110 106 174 13 165 186 88 245 166 27 211 57 50 152 237 140 190
 38 132 46 109 153 85 247 70 56 138 248 202 17 8 227 125 234)(2 159 47 213 98 251 168 41
 92 194 89 77 204 252 208 219 123 111 210 129 243 142 84 79 154 244 30 230 44 95 172

209 99 67 196 37 96 45 231 148 198 255 14 119 59 161 7 78 34 9 91 115 117 18 130 201 73
 253 104 160 191 158 151 137 223 146 226 197 157 232 53 180 136 103 156 80 206 217 10
 235 185 121 40 228 93 122 215 86 65 5 221 66 124 82 133 150 49 74 108 33 240 19 58 25
 143 236 52 12 29 31 94 20 11 83 61 87 249 114 205 68 127 43 100 26 107 22 51 32 72 69
 199 71 128 75 212 218 195 225 173 105 24 55 188 164)(229)

Rys. 6.3. Permutacja wygenerowana przez HGP_1 z przykładu 6.4

W permutacji tej można wyróżnić dwa podcykle o długościach 110 i 144, oraz dwa elementy stałe 0 i 229. Warto zauważyć także, iż w wyniku złożenia przekształceń w tym hybrydowym generatorze permutacji w trakcie przetwarzania bloku danych zostaje zmieniona wartość najbardziej znaczącego bitu, co w konsekwencji prowadzi do likwidacji regularności w permutacji generowanej za pomocą przekształcenia $\mathbb{K}_{0:7}^{2,4}$ (patrz rozdz. 5). Fakt ten dobrze prezentuje rys. 6.4.



Rys.6.4. Histogram permutacji z rys 6.3. wygenerowanej przez HGP_1 .

W tym przykładzie, HGP_1^{-1} musi mieć strukturę $\mathbb{A}_{0:7}^{-1} \langle c=0 \rangle \mapsto \mathbb{I}_{2:7} \mapsto \mathbb{K}_{0:7}^{2,4}$. Można zauważyć, że taki układ ten generuje permutację odwrotną do wygenerowanej przez HGP_1 . Jest to permutacja:

(0)(1 234 125 227 8 17 202 248 138 56 70 247 85 153 109 46 132 38 190 140 237 152 50 57
 211 27 166 245 88 186 165 13 174 106 110 238 214 239 60 39 3 179 21 224 187 178 216 36
 102 171 241 200 63 192 163 167 28 189 64 139 23 182 254 246 6 141 233 177 35 113 131
 118 81 176 16 250 220 42 222 170 116 76 155 112 62 144 120 162 101 4 54 242 48 181 175
 183 134 145 169 203 90 147 126 149 97 207 15 135 193 184)(2 164 188 55 24 105 173 225
 195 218 212 75 128 71 199 69 72 32 51 22 107 26 100 43 127 68 205 114 249 87 61 83 11

20 94 31 29 12 52 236 143 25 58 19 240 33 108 74 49 150 133 82 124 66 221 5 65 86 215
 122 93 228 40 121 185 235 10 217 206 80 156 103 136 180 53 232 157 197 226 146 223 137
 151 158 191 160 104 253 73 201 130 18 117 115 91 9 34 78 7 161 59 119 14 255 198 148
 231 45 96 37 196 67 99 209 172 95 44 230 30 244 154 79 84 142 243 129 210 111 123 219
 208 252 204 77 89 194 92 41 168 251 98 213 47 159)(229)

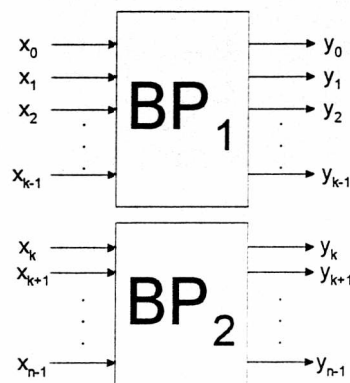
Rys. 6.5. Permutacja wygenerowana przez HGP_1^{-1} z przykładu 6.4

Zgodnie z przewidywaniami, permutacja wygenerowana przez HGP_1^{-1} ma identyczną strukturę jak permutacja 6.1 i jest permutacją do niej odwrotną, co łatwo zidentyfikować po kilku zaznaczonych w permutacji podkreśleniami parach liczbowych.

Godny uwagi jest także fakt, że sekwencyjne łączenie bloków permutacyjnych stwarza możliwość dopasowywania struktury hybrydowego generatora permutacji do możliwości i wymagań technicznych poprzez uproszczenie jego budowy. Można to uczynić analizując równania opisujące sposób generowania kolejnych bitów i pod tym kątem je redukować. Istotny jest tutaj jednakże dobór takiej kolejności bloków permutacyjnych, która pozwoli na redukcję złożoności a nie skomplikuje jeszcze hybrydowego generatora permutacji⁴.

Połączenie równoległe

Inna konfiguracja bloków permutacyjnych dotyczy równoległego przetwarzania bitów przez dwa generatory permutacji BP_1 i BP_2 . Przy tym połączeniu każdy z układów przetwarza oddzielny zakres pozycji bitowych, czyli żadna pozycja bitowa nie jest w takim HGP przetwarzana przez obydwa układy.



Rys. 6.6. Równoległe połączenie dwóch bloków permutacyjnych

⁴ Pomocne w tym mogą być programy komputerowe dostępne w sieci Internet („Espresso”, „minim” i inne) służące do minimalizacji wyrażeń boole’aowskich lub inne używane w komercyjnych programach wspomagających projektowanie układów cyfrowych (OrCAD, AutoCAD i inne).

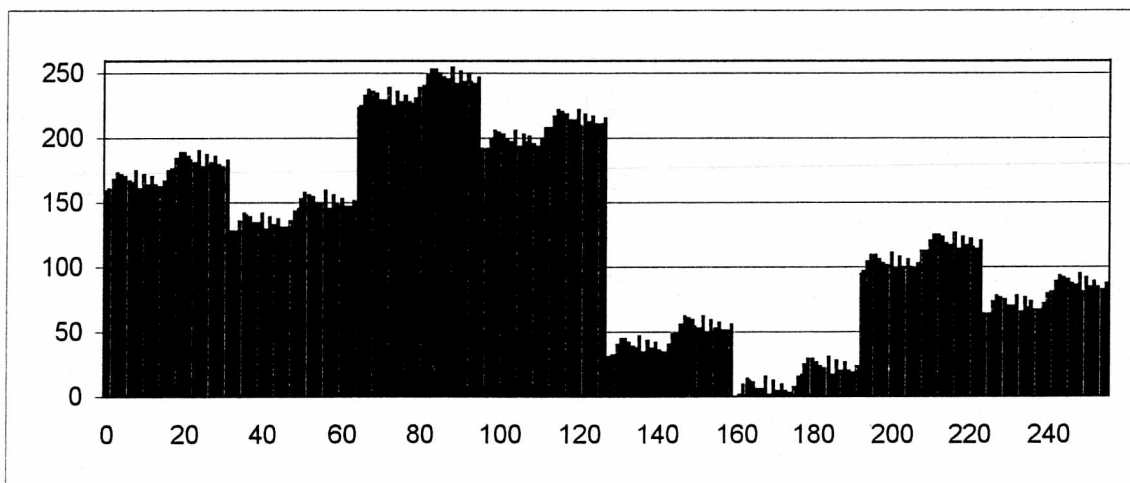
Takie połączenie bloków permutacyjnych pozwala na generowanie permutacji zbiorów o dużej ilości elementów za pomocą bloków permutacyjnych operujących na niewielkiej liczbie bitów.

PRZYKŁAD 6.5 Dany jest 8-bitowy HGP_1 o strukturze przetwarzania $\overset{2}{I}_{0:3} \langle x^4 + x + 1 \rangle \mapsto \overset{4}{X}_{4:7} \langle 160 \rangle^5$. Przy czym blok permutacyjny realizujący inwersję w ciele $GF(2^4)$ opisany jest układem równań

$$\begin{aligned} y_0 &= x_0 + x_1 + x_2 + x_3 + x_0x_2 + x_1x_2 + x_0x_1x_2 + x_1x_2x_3, \\ y_1 &= x_3 + x_0x_1 + x_0x_2 + x_1x_2 + x_1x_3 + x_0x_1x_3, \\ y_2 &= x_2 + x_3 + x_0x_1 + x_0x_2 + x_0x_3 + x_0x_2x_3, \\ y_3 &= x_1 + x_2 + x_3 + x_0x_3 + x_1x_3 + x_2x_3 + x_1x_2x_3. \end{aligned} \quad (6.7)$$

Z opisu struktury HGP widać wyraźnie, że BP, oparty na inwersji działa na 4 najmniej znaczących bitach bloku, zaś generator, realizujący sumowanie modulo 2 zmienia, jedynie bit 5 i 7.

Tak skonstruowany HGP generuje permutację, której histogram zamieszczony został na rysunku 6.7.



Rys. 6.7. Histogram permutacji z przykładu 6.5.

Wyraźnie dostrzegalna jest na histogramie własność permutacji polegająca na cyklicznym powielaniu na różnych poziomach odniesienia wartości generowanych przez układ przetwarzający 4 najmniej znaczące bity. Wartości odwzorowań dla liczb powyżej 15 można

⁵ W tym przykładzie zastosowano dwa bloki permutacyjne przetwarzające niezależne układy bitów, struktura HGP_1^{-1} jest identyczna jak HGP_1 ; układ generuje permutację samoodwracalną.

tutaj łatwo przewidzieć, biorąc za podstawę wyniki z zakresu $0 \div 15$ i uwzględniając przesunięcie, wynikające z aktualnego układu wartości czterech najbardziej znaczących bitów, który zmienia się co 16 pozycji. Można to łatwo sprawdzić, analizując podkreślone wartości występujące w przykładzie na rys.6.8.

(0 160)(1 161)(2 169)(3 174)(4 173)(5 171)(6 167)(7 166)(8 175)(9 162)(10 172)(11 165)(12 170)(13 164)(14 163)(15 168)(16 176)(17 177)(18 185)(19 190)(20 189)(21 187)(22 183)(23 182)(24 191)(25 178)(26 188)(27 81)(28 186)(29 180)(30 179)(31 184)(32 128)(33 129)(34 137)(35 142)(36 141)(37 139)(38 135)(39 134)(40 143)(41 130)(42 140)(43 133)(44 138)(45 132)(46 131)(47 136)(48 144)(49 145)(50 153)(51 158)(52 157)(53 155)(54 151)(55 150)(56 159)(57 146)(58 156)(59 149)(60 154)(61 148)(62 147)(63 152)(64 224)(65 225)(66 233)(67 238)(68 237)(69 235)(70 231)(71 230)(72 239)(73 226)(74 236)(75 229)(76 234)(77 228)(78 227)(79 232)(80 240)(81 241)(82 249)(83 254)(84 253)(85 251)(86 247)(87 246)(88 255)(89 242)(90 252)(91 245)(92 250)(93 244)(94 243)(95 248)(96 192)(97 193)(98 201)(99 206)(100 205)(101 203)(102 199)(103 198)(104 207)(105 194)(106 204)(107 197)(108 202)(109 196)(110 195)(111 200)(112 208)(113 209)(114 217)(115 222)(116 221)(117 219)(118 215)(119 214)(120 223)(121 210)(122 220)(123 213)(124 218)(125 212)(126 211)(127 216)

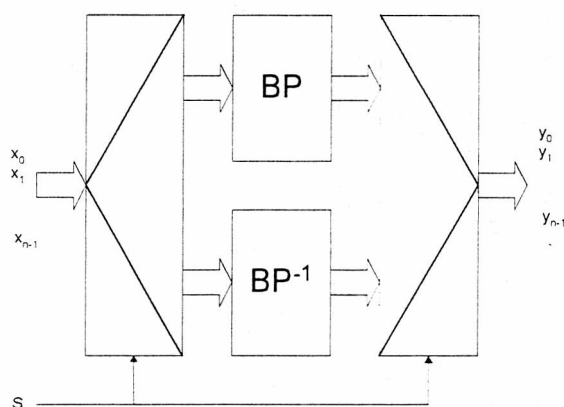
Rys. 6.8. Permutacja wygenerowana za pomocą (6.7)

Biorąc jako przykład liczbę 5 (00000101_2), która w przekształceniu przechodzi w 171 (10101011_2) można obliczyć wartość permutacji dla 21 (ustawienie bitu piątego). Wystarczy dodać do wartości odwzorowania dla 5 wartość, wynikającą z ustawienia bitu piątego, czyli 16, aby otrzymać wynik $171+16=187$ (10111011_2). Stąd łatwo wyciąga się wniosek, iż obrazem 21 w tej permutacji jest 187.

6.2.3 Uwarunkowania strukturalne konstruowania wzajemnie odwracalnych i samoodwracalnych hybrydowych generatorów permutacji

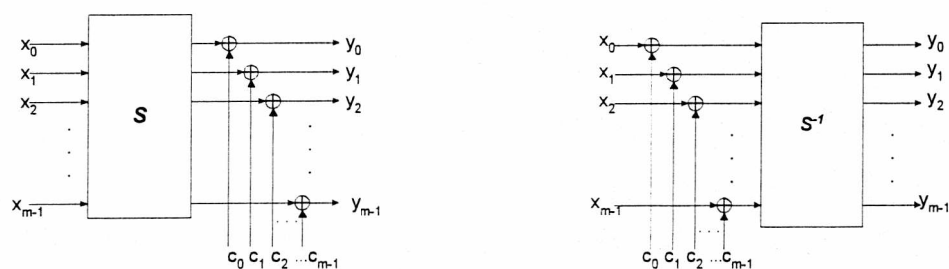
Zgodnie z treścią poprzednich rozdziałów permutacje można podzielić na dwa rodzaje. Pierwszy z nich to permutacje samoodwracalne, w których jeden BP generuje permutację P spełniającą zależność $P \cdot P^{-1} = e$, gdzie e - permutacja identycznościowa (patrz rozdz.2.1 i 4 oraz 5.2). Drugi rodzaj to permutacje wzajemnie odwracalne (patrz rozdz. 3, 5.1), w których występują dwa różne bloki permutacyjne BP_1 i BP_2 generujące różne permutacje P_1 i P_2 , które

spełniają zależność $P_1 P_2 = P_2 P_1 = e$. To rozdzielenie funkcji na dwa różne bloki permutacyjne powoduje, że urządzenie stosujące takie rozwiązanie musi zawierać układy przełączające. Schemat ogólny takiego urządzenia przedstawiono na rys. 6.9.



Rys. 6.9. Schemat urządzenia z zastosowaniem elementów przełączających generującego wzajemnie odwracalne permutacje. Przełączaniem steruje sygnał s .

Specyfika każdego stosowanego generatora permutacji warunkuje sposób jego zastosowania przy konstruowaniu hybrydowych generatorów permutacji. Szczególnie dotyczy to rozwiązań z zastosowaniem przekształceń afinicznych przestrzeni liniowych nad ciałami Galois charakterystyki 2 dla przypadku, gdy wartość dodawanej stałej c (patrz rozdz. 3) jest różna od zera. Wtedy mamy do czynienia ze złożeniem dwóch operacji, pierwszej odwzorowującej postać układu równań związanego z postacią nieosobliwej macierzy nad $GF(q)$, oraz drugiej, realizującej sumowanie modulo 2. Parę takich wzajemnie odwracalnych bloków permutacyjnych zaprezentowano na rysunku 6.10.



Rys. 6.10. Przykład pary bloków permutacyjnych z zastosowaniem przekształcenia afinicznego nad $GF(2)$

Z powyżej przedstawionej przyczyny, łatwiej rozpatrywać taki blok permutacyjny jako hybrydowy generator permutacji o strukturze $\text{HGP} \xrightarrow[r_{\min}:r_{\max}]{} \overset{q}{A} \mapsto \mathbb{X} \langle c \rangle$, zaś $\text{HGP}^{-1} \xrightarrow[r_{\min}:r_{\max}]{} \mathbb{X} \langle c \rangle \mapsto \overset{q}{A}$. Jak widać w opisie wystąpiła także modyfikacja określenia związanego z funkcją afiniczną polegająca na pominięciu parametru dotyczącego stałej. Tak zmodyfikowane podejście do bloków permutacyjnych stosujących przekształcenie afiniczne będzie odąd stosowane w dalszym ciągu niniejszej pracy.

Konstruowanie hybrydowych generatorów permutacji samoodwracalnych, jest zagadnieniem dość prostym, choć wymaga zachowania pewnych zasad przy łączeniu bloków permutacyjnych. Przede wszystkim należy zwrócić uwagę na to, że w takim generatorze kolejność bloków składowych nie może być dowolna. Zjawisko takie zachodzi w układach zbudowanych przy zastosowaniu parami bloków generujących permutacje wzajemnie odwracalne czyli np.

$$\text{BP}_1 \mapsto \text{BP}_2 \mapsto \dots \mapsto \text{BP}_k \mapsto \text{BP}_k^{-1} \mapsto \dots \mapsto \text{BP}_2^{-1} \mapsto \text{BP}_1^{-1}. \quad (6.8)$$

Jednak wynikiem działania takiego HGP byłaby zawsze permutacja identycznościowa. Aby to zmienić, między blokami permutacyjnymi należy umieścić taki pojedynczy blok BP_s , który generuje permutację samoodwracalną, i który nie będzie pozwalał na odwracanie permutacji przez pary generatorów do niego przylegające. Wobec tego ogólna struktura układu hybrydowego generującego permutację samoodwracalną jest następująca

$$\text{BP}_1 \mapsto \text{BP}_2 \mapsto \dots \mapsto \text{BP}_k \mapsto \text{BP}_s \mapsto \text{BP}_k^{-1} \mapsto \dots \mapsto \text{BP}_2^{-1} \mapsto \text{BP}_1^{-1}. \quad (6.9)$$

Jak łatwo zauważyć hybrydowy generator permutacji generujący permutację samoodwracalną skonstruowany, według powyższej metody, zawsze będzie składał się z nieparzystej ilości bloków permutacyjnych.

PRZYKŁAD 6.6 Dany jest HGP o strukturze zbudowanej według powyżej podanych zasad

$$\overset{2}{I} \xrightarrow[0:5]{} \overset{2}{A} \xrightarrow[0:7]{} \overset{4}{I} \xrightarrow[0:5]{} \overset{2}{A}^{-1} \xrightarrow[0:5]{} \overset{2}{I}. \text{ Można łatwo sprawdzić że, permutacja wygenerowana przez}$$

ten układ jest samoodwracalna.

(0)(1 99)(2 36)(3 119)(4 225)(5 61)(6 237)(7 48)(8 111)(9 14)(10 89)(11 203)(12 46)(13 93)(15 54)(16 67)(17 101)(18 246)(19 52)(20 117)(21 108)(22 130)(23 129)(24 204)(25 112)(26)(27 142)(28 88)(29 105)(30 47)(31 51)(32 200)(33 49)(34 188)(35 192)(37 229)(38 167)(39 159)(40 157)(41 151)(42 176)(43 95)(44 66)(45 143)(50 232)(53 183)(55 241)(56 100)(57 98)(58 195)(59 168)(60 144)(62 184)(63 228)(64 155)(65 109)(68 127)(69 133)(70

77)(71 158)(72 146)(73 78)(74 207)(75 96)(76 121)(79 239)(80 92)(81 213)(82)(83 180)(84 148)(85 132)(86)(87 91)(90 208)(94 215)(97 156)(102 115)(103 216)(104 114)(106 244)(107 250)(110 254)(113 191)(116 217)(118 201)(120 179)(122 194)(123 205)(124 172)(125 163)(126 249)(128 136)(131 186)(134 169)(135 175)(137 197)(138 221)(139 206)(140 150)(141)(145 181)(147 219)(149 190)(152 174)(153 233)(154 226)(160 222)(161 238)(162 223)(164 165)(166 255)(170 189)(171 214)(173 253)(177)(178 234)(182 242)(185 235)(187 211)(193 198)(196)(199 202)(209 245)(210 227)(212 220)(218 243)(224 251)(230 248)(231 247)(236 252)(240)

Rys. 6.11. Permutacja samoodwracalna z przykładu 6.6

6.3 Analiza własności permutacji generowanych metodą hybrydową

6.3.1 Założenia i oprogramowanie pomocnicze.

W celu osiągnięcia dużej czytelności wyników testowania hybrydowych generatorów permutacji o różnej strukturze ilość bitów przetwarzanych przez HGP ograniczono do 8 bitów. Testy przeprowadzono na generatorach hybrydowych złożonych wyłącznie z bloków permutacyjnych opisanych równaniami prezentowanymi w niniejszej pracy.

Oprogramowanie do testowania układów, którego niektóre podstawowe pliki źródłowe zamieszczono w dodatkach do niniejszej pracy, napisano w języku C.

Jako narzędzia generującego programową postać równań opisujących bloki permutacyjne wykorzystano także napisaną przez autora procedurę pakietu „Mathematica ver.2.0” pozwalającą na obliczenie wzorów do realizacji operacji w ciałach $GF(2^n)$ dla dowolnego wielomianu nieredukowalnego $p(x)$. Treść tej procedury została zamieszczona w dodatku.

Do analizy permutacji uzyskanych metodą hybrydową wykorzystano m.in. bezpłatnie pozyskany przez sieć Internet pakiet matematyczny „GAP” w wersji 3.4 dla DOS opracowany przez Wydział Matematyki Uniwersytetu w Aachen [WWWGA], Niemcy oraz arkusz kalkulacyjny „Excel” w wersji dla Windows 95 do wykonania histogramów.

6.3.2 Wyniki testów

Testy układów do generowanie metodą hybrydową permutacji zbiorów o liczności 2^n wykazały szereg uwarunkowań, związanych z przetwarzaniem dużych binarnych bloków

danych za pomocą połączonych generatorów permutacji, działających na mniejszej liczbie bitów. Szczególnie dotyczy to ograniczeń, związanych z hybrydowymi generatorami permutacji, połączonymi równolegle. Ten sposób łączenia pozwoliłby bowiem najprościej zwiększyć ilość przetwarzanych bitów.

W celu ułatwienia analizy ciągów permutacyjnych generowanych za pomocą układów przetwarzających tylko część pozycji bitowych, opracowano tabelę pomocniczą, zawierającą dziesiętne wartości liczb, których pozycje bitowe nie podlegają przetwarzaniu (przechodzą same na siebie) przez układy permutacyjne przy założonych wartościach ilości przetwarzanych przez układ bitów. Dane opracowane dla liczb 8 bitowych przedstawiono w tabeli 6.1.

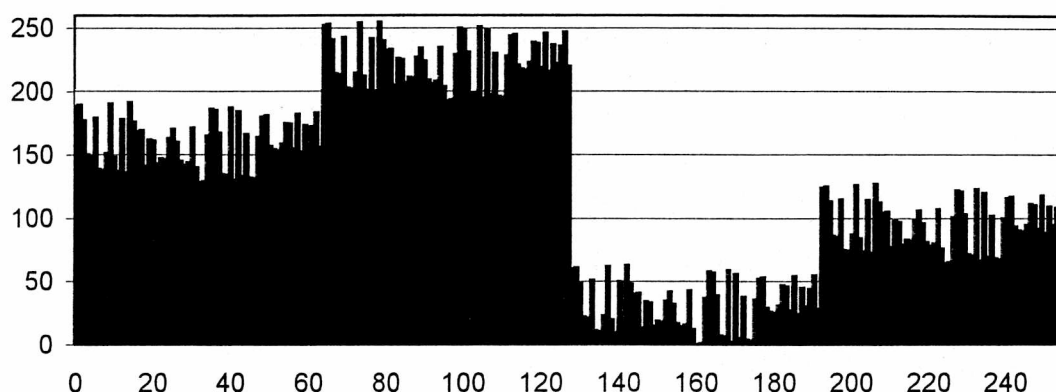
Tabela 6.1 Pomocnicza tablica wartości do analizy permutacji hybrydowych.

Ilość bitów	zakres bitów	Wartości dziesiętne liczb nie zmienianych w trakcie przetwarzania
4	0-3	16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240
	1-4	1, 32, 33, 64, 65, 96, 97, 128, 129, 160, 161, 192, 193, 224, 225
	2-5	1, 2, 3, 64, 65, 66, 67, 128, 129, 130, 131, 192, 193, 194, 195
	3-6	1, 2, 3, 4, 5, 6, 7, 128, 129, 130, 131, 132, 133, 134, 135
	4-7	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
6	0-5	64, 128, 192
	1-6	1, 128, 129
	2-7	1, 2, 3

Wyniki testów pokazały, że w permutacjach, wygenerowanych przez hybrydowe generatory permutacji o strukturze równoległej zaobserwować można efekt cyklicznej propagacji odwzorowania realizowanego przez pojedynczy BP, działający na najmniej znaczących bitach na cały zakres liczb⁶. Zaobserwowano, że nawet stosowanie połączenia równoległo-sekwencyjnego, z użyciem bloków permutacyjnych przetwarzających zakres bitów mniejszy niż zakres przetwarzany przez układ hybrydowy, nie wyeliminowało tego zjawiska. Na rys.6.12 przedstawiono histogram permutacji wygenerowanej przez HGP o strukturze

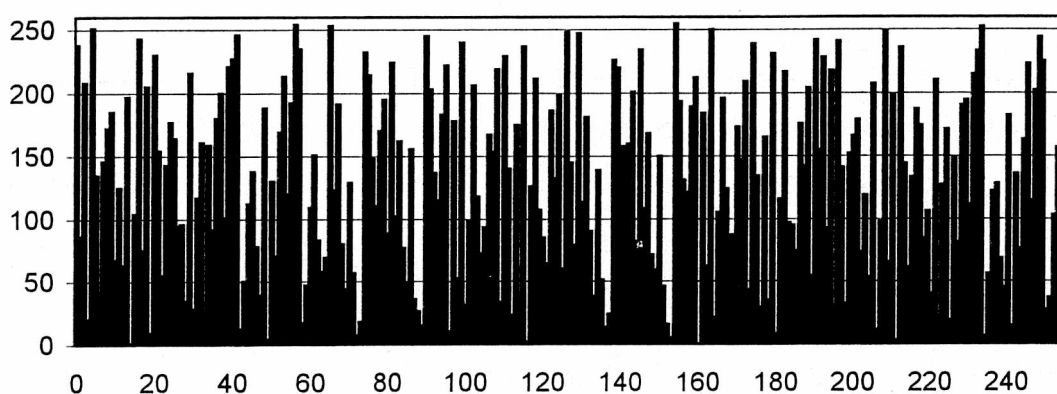
$$\overset{2}{\underset{0:3}{I}}\langle x^4 + x + 1 \rangle \mapsto \overset{2}{\underset{4:7}{X}}\langle 160 \rangle \mapsto \overset{2}{\underset{2:7}{I}}\langle x^4 + x + 1 \rangle.$$

⁶ Widać to wyraźnie w przykładzie z rozdz.6.2.2 dotyczącym połączenia równoległego.



Rys. 6.12. Histogram permutacji testowanego układu równoległo - sekwencyjnego

Porównując ten wynik z histogramem na rys. 6.7, łatwo można zauważyć, że sekwencyjne dołączenie jeszcze jednego układu spowodowało jedynie zmianę długości cyklu powtarzalności powielanego odwzorowania. Testy wielu hybrydowych generatorów permutacji udowodniły, że jedynie sekwencyjne zastosowanie bloku permutacyjnego działającego na pełnym zakresie bitów przetwarzanego przez HGP, likwiduje opisane powyżej zjawisko powtarzalności sekwencji. Na rys. 6.13 umieszczono histogram permutacji uzyskanej za pomocą HGP z przykładu 6.5 z dołączonym sekwencyjnie BP realizującym przekształcenie afiniczne na bloku 8 bitowym $\mathcal{A}_{0.7}^2 \langle c = 0 \rangle$ opisane w rozdz. 3.

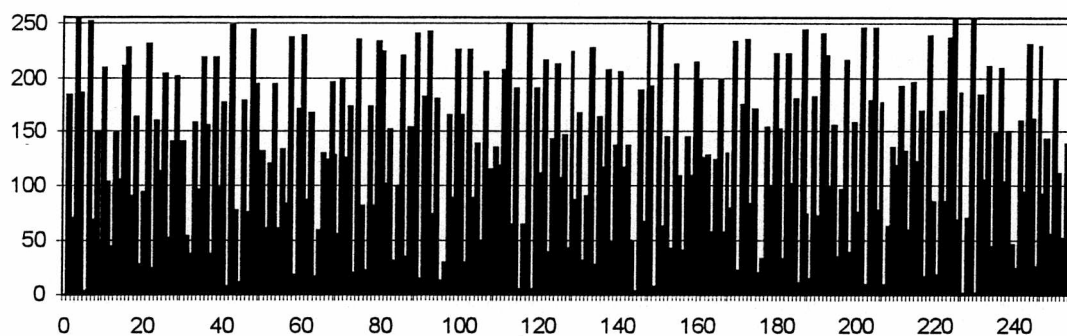


Rys. 6.13. Histogram permutacji wygenerowanej przez HGP z dołączonym BP realizującym przekształcenie afiniczne w $GF(2^8)$

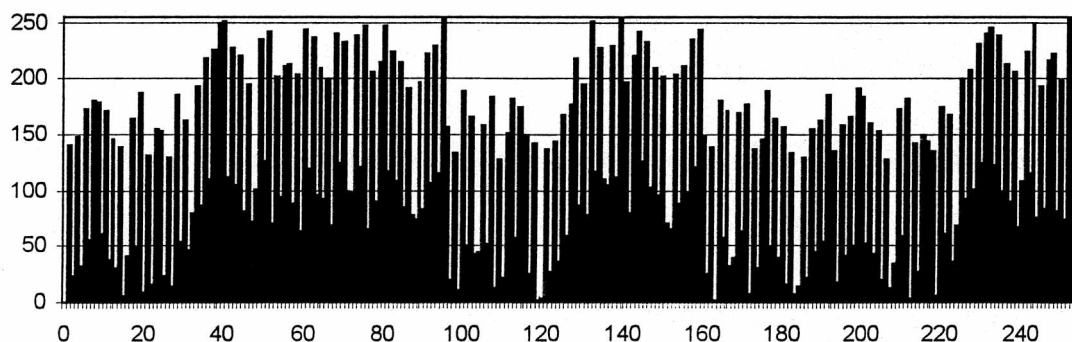
Mimo, iż na histogramie z rys. 6.13 także można zaobserwować pewne regularności, to jednak nie są one tak jednoznaczne, jak to było w poprzednio opisywanych przypadkach.

Wydaje się jednak, że w celu pełnego wyeliminowania opisywanej cechy niezbędne jest sekwencyjnie dołączenie co najmniej dwóch bloków permutacyjnych.

W trakcie badania permutacji wygenerowanych metodą hybrydową stwierdzono także, że bloki posiadające cechę rozpraszania wartości po całym dostępnym zakresie, stosowane w układach hybrydowych, zdecydowanie poprawiają charakterystyki całego HGP pod tym względem. Przykładem może być hybrydowy generator permutacji HGP_1 o strukturze $\mathbb{K}_{0:7}^2 \mapsto \mathbb{A}_{0:7}^2$, w którym cecha układu podnoszącego do kwadratu⁷ skompensowana została permutacją wygenerowaną przez blok, realizujący przekształcenie afiniczne nad $GF(2)$. Warto jednak zwrócić uwagę na fakt, iż istotna jest w tym układzie kolejność występowania bloków permutacyjnych. Dla HGP_2 o odwrotnej kolejności, czyli $\mathbb{A}_{0:7}^2 \mapsto \mathbb{K}_{0:7}^2$, uzyskana permutacja miała inne właściwości rozpraszające. Fakt ten można zaobserwować porównując histogramy dwóch permutacji wygenerowanych za pomocą HGP_1 i HGP_2 .



Rys. 6.14. Histogram permutacji wygenerowanej przez HGP_1



Rys. 6.15. Histogram permutacji wygenerowanej przez HGP_2

⁷ Opisana została w rozdziale 5.2

Jak widać z charakterystyk zamieszczonych na rys. 6.14 i 6.15 zmieniając kolejność występowania bloków w generatorze hybrydowym uzyskiwano za każdym razem permutacje o innej charakterystyce. Dowodzi to, że o cechach wygenerowanej w ten sposób permutacji decyduje nie tylko skład i usytuowanie względem siebie bloków permutacyjnych, ale także kolejność ich występowania, co prowadzi do wniosku, iż HGP złożony z k składowych bloków permutacyjnych o ustalonej strukturze może wygenerować co najmniej $n_p = k!$ różnych permutacji. Gdyby jednak uwzględnić jeszcze możliwość przesuwania zakresu przetwarzanych przez pojedynczy BP bitów w ramach przetwarzanego przez HGP bloku a także dobór bloków o różnych strukturach to

$$n_p = m^k \prod_{i=1}^m (n - l_i + 1), \quad (6.10)$$

gdzie

m - ilość typów bloków permutacyjnych,

k - ilość bloków permutacyjnych w HGP,

n - ilość bitów przetwarzana przez HGP,

i - numer kolejnego bloku permutacyjnego,

l_i - ilość bitów przetwarzana przez i - ty blok permutacyjny.

Na przykład za pomocą 8-bitowego HGP złożonego z bloków o strukturze: dwa różne 4-bitowe, jeden 8-bitowy i jeden 6-bitowy, zamieniając je jedynie miejscami i przesuwając zakresy przetwarzania, można wygenerować 1800 różnych permutacji, ponieważ

$$n_p = k! \prod_{i=1}^m (n - l_i + 1) = 4! \cdot (5 \cdot 5 \cdot 3) = 1800, \quad (6.11)$$

Zawsze jednak w takim przypadku należy wziąć pod uwagę fakt, że nie każda z nich będzie miała pożądaną przez nas własność.

7. Konstruowanie generatorów permutacji dla zastosowań kryptograficznych

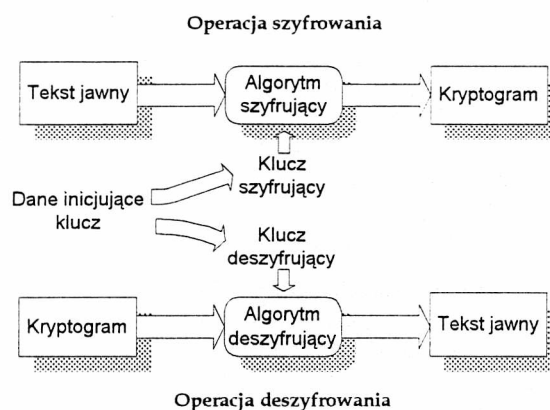
W rozdziale przedstawiono sposoby weryfikacji generatorów permutacji z punktu widzenia ich konkretnych zastosowań kryptograficznych. Przede wszystkim pokazano, w jaki sposób można zastosować hybrydowe generatory permutacji w szyfrach blokowych. Zaprezentowano ogólnie blokową oraz strumieniową metodę szyfrowania i pokazano sposoby użycia w nich hybrydowych generatorów permutacji. Podano też szereg metod testowania permutacji pod kątem własności decydujących o ich przydatności do zastosowań w systemach kryptograficznych. Większość wyników zilustrowano przykładami dotyczącymi wcześniej zaprojektowanych układów.

7.1 Charakterystyka metod szyfrowania

Projektowanie i konstruowanie urządzeń szyfrująco/deszyfrujących zawsze związane jest z realizacją sprzętową, programową lub hybrydową (sprzętowo - programową) pewnego algorytmu szyfrowania. W celu utrudnienia analizy swoich produktów projektanci tworzą bardzo skomplikowane struktury lub prezentując swoje pomysły publicznie, stosują metodę niedopowiedzeń, przekłamań lub odwrócenia uwagi. Powoduje to, że osoby zainteresowane poznają wiele nieistotnych szczegółów o nowym urządzeniu i nie mają jasnego obrazu na temat działania kryptosystemu¹. Takie zabiegi w wielu przypadkach powodują jednak tylko zniechęcenie lub odstraszenie mniej wprawnych w sztuce kryptoanalizy potencjalnych oponentów natomiast nie poprawiają bezpieczeństwa samego kryptosystemu. Każdy wprawny kryptoanalityk potrafi bowiem zawsze wskazać w systemie kryptograficznym podstawowe elementy, związane z szyfrowaniem i deszyfrowaniem, a następnie, na podstawie analizy sposobu działania całego systemu, poprzez odpowiednie zaklasyfikowanie kryptosystemu może dobrać właściwe metody łamania szyfru.

¹ Dotyczy to także algorytmów działania kryptosystemów ogólnie znanych i prezentowanych przez różne agendy rządowe na całym świecie, jako bezpieczne pod względem kryptograficznym [Sch95].

Podstawą ideą działania szyfru jest modyfikacja informacji przeznaczonej dla odbiorcy informacji w taki sposób, aby jedynie ta osoba, mogła ją poprawnie zinterpretować. Schemat procesu wymiany informacji stosującego szyfrowanie przedstawiono na rys. 7.1.



Rys. 7.1. Schemat wymiany informacji z zastosowaniem szyfru

Centralnym punktem idei szyfrowania jest element, który upoważnia nadawcę i odbiorcę informacji do operowania przekazywaną informacją w taki sposób, że jej treść jest nieczytelna dla osób postronnych. Tym elementem jest klucz kryptograficzny. W definicji tej nie ma znaczenia, czy klucz jest jedno- czy wielopostaciowy (np. ma inną postać dla procesu szyfrowania a inną dla deszyfrowania). Właściwością klucza jest zawsze to, że jest on składnikiem zaszyfrowanej wiadomości, powodującym jej nieczytelność dla osób nieupoważnionych.

Współcześni kryptografowie, na podstawie doświadczeniach z dziedziny wymiany niejawnych informacji, już dawno temu wskazali na najsłabsze ogniwo w systemie bezpieczeństwa, opartym na stosowaniu szyfru, jakim jest człowiek. Z tego powodu użytkownicy kryptosystemów nie wiedzą często jakimi kluczami szyfrowane są ich informacje, a jedynie znają postać elementów, inicjujących proces ich generowania. Prawdziwy klucz kryptograficzny łączony jest tymczasem w końcowym etapie procesu szyfrowania z tekstem jawnym, powodując, że kryptogram ma postać bardzo różniącą się od informacji skierowanej do odbiorcy. Aby więc poprawnie zanalizować algorytm działania jakiegokolwiek kryptosystemu, należy najpierw wskazać w nim moment procesu łączenia tekstu jawnego z kluczem kryptograficznym, a następnie określić metodę generowania klucza.

Istnieją tylko dwa podstawowe sposoby generowania klucza kryptograficznego. Kryterium podziału stanowi zmienność postaci klucza w procesie szyfrowania, dokładniej

mówiąc użycie pamięci stanów kryptosystemu. Zgodnie z tą klasyfikacją szyfry dzieli się na blokowe i strumieniowe, lub inaczej bezpamięciowe i wyposażone w wewnętrzną pamięć stanów.

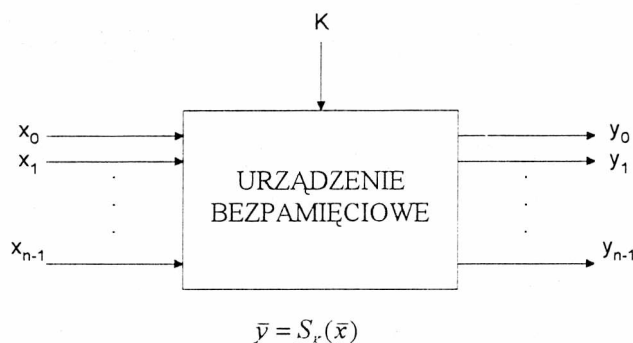
Klasyfikując szyfry w ten sposób mówi się o blokowej (bepamięciowej) lub strumieniowej (pamięciowej) metodzie szyfrowania informacji. Właśnie zależnie od wyboru jednego z tych dwóch sposobów zależy w dużej mierze wybór parametrów składników kryptosystemu, takich jak opisywane w niniejszej pracy generatory permutacji.

Szyfrowanie metodą blokową.

W szyfrach blokowych [Mas93][Mor97][Ole96][Rit98][WWWTR], każdy ustalonej wielkości fragment tekstu jawnego zwany blokiem, szyfrowany jest niezależnie tym samym kluczem szyfrującym. Szyfry blokowe są szyframi podstawieniowymi, których bezpieczeństwo zależy przede wszystkim od rozmiaru bloku. Im większy jest blok, tym większa jest przestrzeń możliwych do użycia kluczy szyfrujących. W metodzie tej, przy ustalonej wartości klucza K , kryptosystem zawsze identycznie przekształci blok tekstu jawnego w odpowiadający mu blok kryptogramu.

Z powodu prostoty podejścia do problemu użycia klucza, szyfry blokowe wymagają zastosowania w procesie szyfrowania/deszyfrowania wielu mechanizmów, chroniących tworzony kryptogram przed zbyt łatwą analizą, poprzez zastosowanie jednej lub wielu skomplikowanych funkcji, realizujących podstawienie². Najczęściej funkcje te powodują mieszanie i rozpraszanie informacji, co sprawia, że kryptogram pozbawiony zostaje pewnych charakterystycznych cech tekstu jawnego i staje się niejako „obojętny” na statystyczne testy weryfikacyjne.

² Taką pojedynczą funkcją może być jakiekolwiek przekształcenie, dla którego zadanie znalezienia przekształcenia odwracającego jest bez znajomości wartości odpowiednich parametrów obliczeniowo niewykonalne lub bardzo trudne [Den83][Sch95].



Rys. 7.2. Schemat szyfrowania metodą blokową przy n - bitowej wielkości bloku.

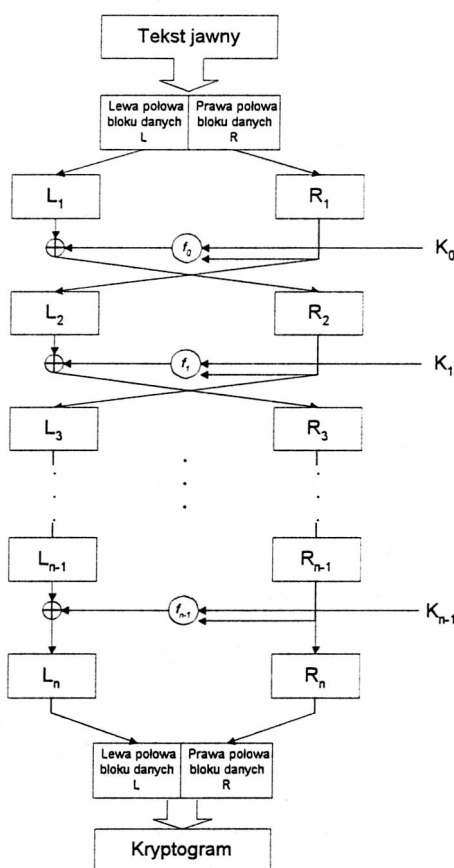
S funkcja realizująca podstawienie w zależności od wartości K

Podstawowymi elementami szyfratora szyfru blokowego są zawsze układy realizujące przekształcenia podstawieniowo-permutacyjne. Elementy realizujące podstawienia, zwane s-skrzynkami (substitution box), zastępują m -bitowe bloki danych innymi n -bitowymi blokami tak, że dowolna zmiana postaci danych wejściowych powoduje losowo wyglądającą zmianę w bloku wynikowym. Warto przy tym zaznaczyć, że blok przetwarzany przez pojedynczą s-skrzynkę jest zwykle mniejszy od bloku danych szyfrowanego przez cały kryptosystem. Relacja między rozmiarem bloku wejściowego m i bloku wyjściowego n nie jest ściśle określona i zależy od przyjętego rozwiązania. Stosowane są różne warianty, choć najczęściej we współczesnych systemach przyjmuje się, iż najkorzystniejsze parametry wprowadzają skrzynki, dla których $m \leq n$. Struktura wewnętrzna s-skrzynek, oparta na wyrafinowanych funkcjach nieliniowych, kolejność użycia i sposób zastosowania, są często wynikiem długich badań i wielu doświadczeń, polegających na doborze odpowiednich dla danego kryptosystemu cech aplikacyjnych, choć dobór ten realizowany jest jednak także z zastosowaniem pewnych określonych, ogólnie znanych kryteriów.

Pierwszym nowoczesnym i jednocześnie jakby wzorcowym szyfrem blokowym stał się 64-bitowy DES (Data Encryption Standard) opracowany w USA w latach siedemdziesiątych przez firmę IBM przy współpracy z NSA (National Security Agency) oraz NIST (National Institute of Standards and Technology). Jego struktura stanowi, jak do tej pory, wzorzec blokowej metody szyfrowania danych z powodu takiego doboru własności przekształceń permutacyjno - podstawieniowych, że przez niemal dwadzieścia lat opierał się on różnym próbom wielokryterialnej analizy kryptograficznej, a w niektórych przypadkach swoimi

własnościami wyprzedzał nawet odkrycia w dziedzinie kryptoanalizy³ [Sch95] [Den83] [Sto94][Bro93].

Opierając się na publicznie prezentowanych doświadczeniach, zdobytych dzięki DES, wprowadzono obecnie kilka standardów projektowania tego typu szyfrów. Jednym z najnowocześniejszych jest kanadyjski CAST, pozwalający na zaprojektowanie własnego szyfru, działającego podobnie jak DES, o dobrych parametrach kryptograficznych i właściwie nieograniczonej wielkości bloku danych. Materiały dotyczące tych kryteriów projektowych są ogólnie dostępne np. w sieci Internet [Ada97]. Uproszczony schemat procesu szyfrowania, zwanego siecią permutacyjno - podstawieniową, w takim podobnym do DES kryptosystemie przedstawiono na rys. 7.3.



Rys.7.3. Schemat przetwarzania danych w sieci permutacyjno-podstawieniowej

W algorytmie tym, podobnie jak w DES, 32-bitowe s-skrzynki⁴ występują w funkcjach f_i [Ada97], których zadaniem jest generowanie kolejnych sekwencji klucza kryptograficznego

³ Gdy w 1989r odkryto tzw. kryptoanalizę różnicową okazało się, że DES jest na nią odporny, bowiem jej twórcy już dwadzieścia lat wcześniej znali tę metodę łamania szyfrów.

⁴ W DES zastosowano 4-bitowe s-skrzynki.

łączonego następnie z lewą połową bloku danych za pomocą sumowania modulo 2. Przetwarzany przez s-skrzynkę zestaw danych jest połączeniem połowy szyfrowanego bloku danych i odpowiednio przygotowanego fragmentu K_i wejściowego klucza kryptograficznego K . Po wykonanej w s-skrzynce operacji podstawienia bity uzyskanego bloku danych są permutowane⁵.

Warto zauważyć, że mimo iż większość reguł projektowania jest ogólnie znana, spory dotyczące dodatkowych własności s-skrzynek zastosowanych w DES nie ustają. Nie wszystkie opublikowane przez twórców tego szyfru kryteria są wystarczająco jasne z punktu widzenia zastosowania przekształceń permutacyjno - podstawieniowych. Nikt jednak, do tej pory, przekonująco nie udowodnił istnienia tzw. tajnych przejść, pozwalających na łatwe złamanie szyfru przez odpowiednie służby, ukrytych w strukturze bloków podstawieniowych, choć istnieje wiele przesłanek, że takie przejścia istnieją [Dou95][Sim92].

Szczegółowe kryteria projektowe dotyczące parametrów s-skrzynek oraz sposobu ich wykorzystania w szyfrach blokowych zostały zaprezentowane w dalszym ciągu niniejszego rozdziału.

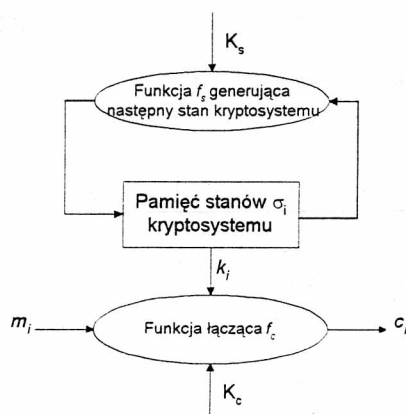
Szyfrowanie metodą strumieniową.

W kryptosystemie szyfrującym metodą⁶ strumieniową [Rue86][Sch95][Den93] każda ustalonej wielkości porcja tekstu jawnego szyfrowana jest kluczem o wartości zmiennej w trakcie szyfrowania. W procesie generowania takiego strumienia klucza (lub kluczy), po zaszyfrowaniu każdego kolejnego bloku tekstu jawnego p_i kluczem k_i , następuje zmiana postaci klucza szyfrującego według pewnych określonych reguł, związanych przeważnie z aktualną wartością parametrów wewnętrznych procesu oraz często postacią uprzednio przetwarzanego fragmentu danych. Strumień klucza, wytworzony w ten sposób, powinien szyfrować tekst jawny tak, aby kryptogram był odporny na analizę statystyczną. Istotne jest także takie działanie generatora kolejnych podkluczy szyfrujących k_i , które zapewniłoby nieprzewidywalność sposobu jego pracy. Osiąga się to przeważnie poprzez nadanie strumieniowi kluczy cech zbliżonych do tych, jakie cechują strumień wartości wytworzonych

⁵Pojęcie permutacji w tym przypadku dotyczy n -elementowego uporządkowanego zbioru elementów o wartościach $\{0,1\}$. W niektórych publikacjach taka permutacja nazywana jest permutacją właściwą.

⁶ W niektórych publikacjach ta metoda szyfrowania nazywana uznawana jest jedynie za tryb pracy szyfru blokowego.

losowo⁷ i zagwarantowanie odpowiednio długiego okresu powtarzalności wartości wytworzonej sekwencji [Rue86][Sch95].



Rys. 7.4 Działanie szyfru strumieniowego.

Jak wynika z rys. 7.4 generator strumienia kluczy k_i , stosownie do wewnętrznego stanu kryptosystemu σ_i i wartości klucza K_s , wytwarza klucz kryptograficzny, którym za pomocą funkcji łączącej f_c oraz zgodnie z wartością klucza K_c szyfrowana jest porcja (blok) tekstu jawnego m_i . Następnie funkcja f_s wytwarza kolejny stan kryptosystemu σ_{i+1} , który służy do wygenerowania kolejnego klucza k_{i+1} .. itd. Występujące parametry K_s i K_c używane są przeważnie jako wartości inicjujące pracę systemu, zaś jako funkcje f_s i f_c wykorzystywane są często szyfry blokowe.

O ile pewne własności dotyczące niektórych parametrów funkcji stosowanych w szyfrach strumieniowych są, z punktu widzenia własności szyfrów blokowych oczywiste, o tyle ta metoda szyfrowania zawiera jednak także inne elementy, które muszą być brane pod uwagę, aby kryptosystem w ten sposób skonstruowany był bezpieczny. Przede wszystkim f_s musi zapewnić *możliwie największą złożoność obliczeniową przekształceń* służących do wygenerowania kolejnego stanu kryptosystemu. Niezbędna jest także taka własność użytych w niej generatorów permutacji, która powoduje deterministyczne wytwarzanie strumienia elementów k_i w taki sposób, aby *maksymalizować okres powtarzalności elementów* tego strumienia, a także, co przy tak sformułowanych warunkach jest szczególnie trudne, *zapewnić łatwą implementację i dużą szybkość działania* systemu szyfrującego.

Projektowanie szyfrów strumieniowych, mimo ich pozornie cennych własności, jest bardzo trudne, lecz dobry szyfr strumieniowy wydaje się być o wiele lepszy od jego blokowego odpowiednika, zwłaszcza że mocne szyfry blokowe mogą być składnikami tak

⁷ Na przykład sekwencja wartości 0 i 1 przyporządkowana wynikom rzucania monetą.

zaprojektowanego szyfru strumieniowego. Warto jednak zauważyć, że nawet zastosowanie w funkcji f_c lub f_s bardzo dobrego szyfru blokowego nie gwarantuje bezpieczeństwa tak skonstruowanego szyfru strumieniowego, bowiem przy źle działającym generatorze strumienia klucza kryptoanalitik może łatwo odgadnąć kolejno wygenerowane klucze k_i .

Najbardziej znanym nowoczesnym szyfrem strumieniowym jest RC4 zaprojektowany przez firmę RSA Laboratories, który jako funkcje wewnętrzne wykorzystuje zmodyfikowane warianty DES. Dane dotyczące algorytmu działania tego systemu, jak zresztą większości szyfrów strumieniowych, nie zostały jednak publicznie udostępnione [Sch95].

7.2 Zacieranie charakterystycznych cech statystycznych tekstu jawnego w kryptogramie

Jedną z podstawowych metod łamania szyfrów jest analiza statystyczna kryptogramu, polegająca na badaniu częstości wystąpień znaków alfabetu języka⁸ (systemu opisu) użytego do jego wykonywania. Ponieważ takie systemy zapisu cechuje redundancja, polegająca na różnicach w częstości wystąpień poszczególnych znaków lub grup znaków, kryptoanalitik, porównując elementy kryptogramu ze znaną sobie statystyczną charakterystyką⁹ zastosowanego opisu potrafi element po elemencie odtworzyć tekst jawny.

Podstawy teoretyczne kryptografii, jako sztuki usuwania naturalnych związków między znakami tekstu, opisał w 1949r w artykule „The Communication Theory of Secrecy Systems” twórca podstaw teorii informacji C.E.Shannon. W swojej pracy przedstawił on dwie podstawowe techniki zacierania tych związków, zwane mieszaniem (konfuzją) i rozpraszaniem (dyfuzją). Konfuzja zmniejsza związki między tekstem jawnym a szyfrogramem tak, że kryptoanalitik nie może złamać szyfru na podstawie badań nadmiarowości lub innych cech statystycznych kryptogramu. Dyfuzja to rozpraszanie kodów znaków występujących w tekście jawnym po całym zakresie kodów użytego alfabetu. Pierwsza z tych technik realizowana jest poprzez stosowanie podstawień, czyli zastępowanie znaków, lub zestawów znaków, występujących w tekście jawnym innymi, zgodnie z odpowiednio przyjętymi regułami. Dyfuzja

⁸ Dotyczy to także plików zawierających inne typy danych jak np. obrazy, dźwięki itp.

⁹ Element tej metody w postaci histogramu rozkładu znaków w pliku zaprezentowano w poprzednich rozdziałach niniejszej pracy.

to przede wszystkim przestawienia znaków, lub bloków znaków, w tekście jawnym (w przypadku zestawu bitów-przestawienia bitów) [Bra88].

Jako wzorzec kryptosystemu rozpraszająco-mieszającego Shannon zaproponował szyfr blokowy, złożony z pewnej liczby małych układów podstawieniowych zwanych s-skrzynkami, oraz układów dużych, realizujących permutację bitów, zwanych p-skrzynkami. Zestaw przekształceń permutacyjno-podstawieniowych, w którym kolejne s-skrzynki mieszają zawartość danych wejściowych a następujące po nich p-skrzynki działają rozpraszająco, Shannon nazwał siecią permutacyjno-podstawieniową (*ang. S-P network*).

Ideę sieci S-P rozwinął w 1973r Feistel oraz Kam i Davida w 1979r, którzy określili także podstawowe cechy takich sieci. Są nimi **lawinowość** i **kompletność (zupełność)** (*ang. avalanche, completeness*).

Pierwsza z tych własności mówi, że zmiana wartości jednego bitu wejściowego powinna prowadzić do zmiany wartości około połowy bitów na wyjściu kryptosystemu. Cecha kompletności wymaga zaś, aby każdy bit wyjściowy był złożoną funkcją wszystkich bitów wejściowych.

Tak sformułowane własności kryptosystemów stały się podstawą opracowania kryteriów projektowych elementów składowych szyfrów, wzorowanych na idei sieci S-P, będących podstawowymi elementami szyfrów blokowych.

W ramach dalszych badań ustalono także, że zasadnicze znaczenie dla ich własności rozpraszająco-mieszających mają przede wszystkim s-skrzynki i to one muszą być niezwykle precyzyjnie dobierane zgodnie z ustalonymi kryteriami. Jednocześnie powstała także precyzyjna definicja s-skrzynki, jako przekształcenia binarnego.

DEFINICJA 7.1 Skrzynką podstawieniową S (s-skrzynką) o wymiarach $n \times m$ ($n \geq m$) nazywa się odwzorowanie $S: \{0,1\}^n \rightarrow \{0,1\}^m$ [Ole96].

Jak łatwo zauważyć przy założeniu, że $n = m$, definicja powyższa może być odniesiona także do przekształcenia realizowanego przez typowy hybrydowy generator permutacji HGP rozpatrywany w niniejszej pracy.

7.2.1 Kryteria weryfikacyjne s-skrzynek

Opierając się na doświadczeniach [Bro93][Mis96][Ole96][Ser96][Ada97] uzyskanych w trakcie prac nad DES oraz kryptoanalizach tego systemu podczas jego eksploatacji określono szereg praktycznych kryteriów weryfikacji s-skrzynek projektowanych dla sieci S-P

zapewniających dobre własności mieszająco-rozpraszające, dając w efekcie kryptogram zupełnie pozbawiony cech tekstu jawnego.

Oto cechy, które powinny posiadać s-skrzynki wchodzące w skład typowej sieci S-P:

- lawinowość,
- kompletność,
- nieliniowość.

Wielu specjalistów podaje, zależnie od praktycznych zastosowań, podaje jeszcze kilka cech s-skrzynek, jednak wymienione powyżej są podstawowe [Rit98:1].

Tak określone własności pozwalają na weryfikację s-skrzynek już na etapie ich projektowania poprzez ustalenie zgodności ich cech z kryteriami.

Kryterium lawinowości (ang. *Avalanche Criterion AC* lub *Strict Avalanche Criterion SAC*)
[Hir94][Hey96][Ser96][Rit98]

DEFINICJA 7.2 Odwzorowanie wzajemnie jednoznaczne $S: \{0,1\}^n \rightarrow \{0,1\}^n$ spełnia kryterium lawinowości, jeżeli dla $\forall f_i \in S: f_i: \{0,1\}^n \rightarrow \{0,1\}$ oraz dla $\forall a: a \in f_i$ i $W_H(a) = 1$ [Ole96][Hir94][Hir95] (W_H - funkcja, której wartością jest ilość jedynek w wektora binarnego)

funkcja $f_i(x) \oplus f_i(x \oplus a)$ jest zrównoważona, czyli $W_H(\sum_{i=0}^{n-1} f_i) = 2^{n-1}$.

W praktyce [Ada97][Hey96][Rit98] kryterium to rozumiane jest w ten sposób, że zmiana wartości dowolnego pojedynczego bitu na wejściu s-skrzynki powinna powodować zmianę co najmniej połowy bitów na wyjściu układu (2 bity dla DES [Sch95][Bro93]), co określane jest wskaźnikiem AC danego przekształcenia podającym (w procentach przetwarzanego zakresu bitów) ile bitów jest zmienianych na wyjściu na skutek zmiany jednego bitu wejściowego. Za wartość AC przyjmuje się średnią dla wszystkich wektorów binarnych w permutacji. Przedstawia to zależność

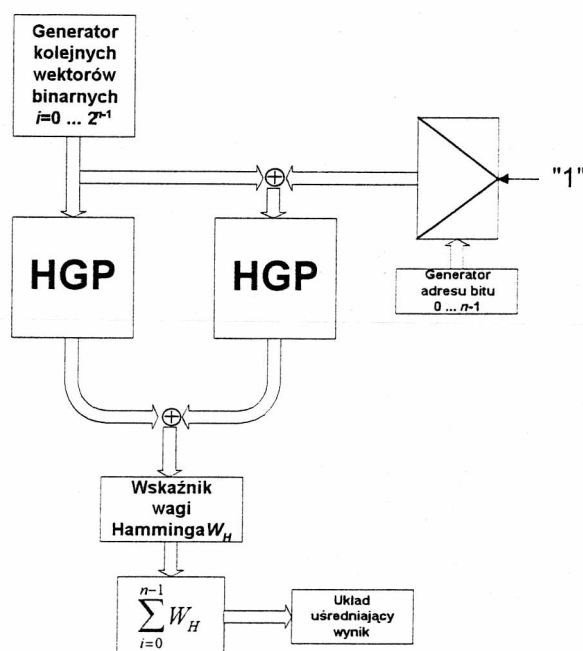
$$AC = \frac{\sum_{i=0}^{2^m-1} \sum_{j=0}^{m-1} I_{ij}}{2^m m^2} \cdot 100\% \quad (7.1)$$

gdzie

I_{ij} - liczba bitów zmienionych na wyjściu układu przy zmianie j -tego bitu wejściowego

dla wartości wektora wejściowego równej 2^i .

Badanie wartości AC w generatorach permutacji można zrealizować stosując układ o strukturze przedstawionej na rysunku 7.5.



Rys. 7.5. Schemat urządzenia do badania lawinowości w układach

Programowa realizacja takiego urządzenia posłużyła do otrzymania wyników pomiaru AC dla generatorów permutacji zaprojektowanych w poprzednich rozdziałach.

Tabela 7.1.

$\overset{2}{A}_{0:7}$	$\overset{8}{A}_{0:5}$	$\overset{2}{I}_{0:5}$	$\overset{4}{I}_{0:5}$	$\overset{2}{K^4}_{0:7}$	$\overset{2}{K}_{0:7}$
50%	16%	50%	50%	12%	12%

Jak widać z tabeli 7.1 kryterium lawinowości spełnia jedynie układ generujący 8-bitowe przekształcenie afiniczne nad $GF(2)$ oraz układy wytwarzające permutacje samoodwracalne oparte na inwersji realizowane na 6 bitach.

Wartość AC związana ściśle jest ze złożonością równań opisujących zastosowane przekształcenia, zatem przy konstruowaniu generatorów hybrydowych, przeważnie nie dochodzi do kompensacji cechy układów o dużej lawinowości. Daje to wartość współczynnika AC układu hybrydowego na poziomie zbliżonym do tego, jaki posiada permutacyjny blok składowy o jego największej wartości AC. Wyjątkiem jest sytuacja, gdy równania opisujące kolejne bloki spowodują zredukowanie złożoności formuły opisującej układ hybrydowy do postaci prostszej niż równania opisujące kolejne przekształcenia składowe. Oto przykłady wartości AC dla niektórych układów hybrydowych.

Tabela 7.2.

$\overset{2}{A} \rightarrow \overset{2}{K}^4$ 0:7 0:7	$\overset{8}{A} \rightarrow \overset{2}{I}$ 0:5 0:5	$\overset{2}{I} \rightarrow \overset{4}{I}$ 0:5 0:5	$\overset{4}{I} \rightarrow \overset{8}{A}$ 0:5 0:5	$\overset{2}{K} \rightarrow \overset{2}{K}^4$ 0:7 0:7	$\overset{2}{A} \rightarrow \overset{2}{K}$ 0:7 0:7
50%	47%	44%	46%	12%	12%

Można zauważyć, że w przypadku układu hybrydowego $\overset{2}{A} \rightarrow \overset{2}{K}$ nastąpiła kompensacja cechy do poziomu posiadanego przez $\overset{2}{K}$. Jednak przy zmianie kolejności bloków składowych AC osiąga wartość 50%.

Kryterium kompletności (ang. Completeness Criterion CC)

DEFINICJA 7.3 Odwzorowanie wzajemnie jednoznaczne $S: \{0,1\}^n \rightarrow \{0,1\}^n$ spełnia kryterium kompletności, jeżeli dla $\forall i, j: i, j \in \{0, \dots, n-1\}$ istnieją takie dwa n -bitowe wektory X_1, X_2 , dla których X_1 i X_2 różnią się tylko wartością bitu i oraz $f(X_1)$ różni się od $f(X_2)$ co najmniej wartością bitu j .

Kryterium to pozwala na projektowanie szyfrów, które byłyby odporne na ataki z tekstem jawnym badające charakterystyki odwzorowań binarnych. W atakach tych bada się wpływ zmian wartości grup bitów wejściowych na wartość pozycji wyjściowych układu przekształcającego.

Praktycznie kryterium kompletności stosowane do projektowania s-skrzynek mówi, że przekształcenie realizowane przez s-skrzynkę jest kompletne wtedy, gdy wartość każdego bitu wyjściowego j zależy od wartości wszystkich bitów wejściowych (jest funkcją wszystkich bitów wejściowych).

W tabeli 7.3 podano wartości CC dla zaprojektowanych pojedynczych generatorów permutacji. Struktura układu badającego tę cechę jest zbliżona do zaprezentowanej dla kryterium lawinowości. Jako miarę kompletności układu CC, podobnie jak dla lawinowości, przyjęto średnią wartość procentową ogólnej liczby bitów wejściowych układu, wpływających na wartość dowolnego bitu wyjściowego.

Tabela 7.3.

$\overset{2}{A}_{0:7}$	$\overset{8}{A}_{0:5}$	$\overset{2}{I}_{0:5}$	$\overset{4}{I}_{0:5}$	$\overset{2}{K^4}_{0:7}$	$\overset{2}{K}_{0:7}$
51%	33%	48%	48%	53%	31%

A oto wartości CC dla przykładowych generatorów hybrydowych:

Tabela 7.4.

$\overset{2}{A}_{0:7} \rightarrow \overset{2}{K^4}_{0:7}$	$\overset{8}{A}_{0:5} \rightarrow \overset{2}{I}_{0:5}$	$\overset{2}{I}_{0:5} \rightarrow \overset{4}{I}_{0:5}$	$\overset{4}{I}_{0:5} \rightarrow \overset{8}{A}_{0:5}$	$\overset{2}{K}_{0:7} \rightarrow \overset{2}{K^4}_{0:7}$	$\overset{2}{A}_{0:7} \rightarrow \overset{2}{K}_{0:7}$
46%	50%	50%	48%	46%	46%

Jak widać z tabeli 7.4 po połączeniu bloków permutacyjnych nastąpiła kompensacja cechy, co spowodowało że wartość CC dla układu hybrydowego zbliżona jest do średniej z wartości poszczególnych bloków składowych.

Ponieważ w praktyce osiągnięcie 100% wartości CC jest niemożliwe, stosuje się jeszcze jedno kryterium weryfikacyjne s-skrzynek, będące modyfikacją kryterium kompletności, zwane **kryterium niezależności bitów** (*ang. Bit Independence Criterion*) [Ada97]. Kryterium niezależności bitów mówi, że wyjściowe bity j i k s-skrzynki powinny zmieniać się niezależnie, kiedy zmieniana jest wartości dowolnego pojedynczego bitu wejściowego i . Własność ta powinna być zachowana dla danych wartości i, j, k ponad całym zbiorem par wektorów wejściowych różniących się między sobą wartości i . Spełnienie tego kryterium jest warunkiem odporności kryptosystemu na dwie najsukuteczniejsze metody kryptoanalityczne: kryptoanalizę różnicową i liniową [Can97] [Kan97] [Ada97].

Weryfikację niezależności zmian bitów wyjściowych od układów zmian na wejściu układu można przeprowadzić na przykład przy pomocy testu χ^2 [Rit98], z hipotezą zerową zakładającą niezależność zmian wartości bitowych. W ramach realizacji pracy dokonano sprawdzenia tej cechy na zaprojektowanych wcześniej generatorach permutacji. Do weryfikacji hipotezy zerowej o niezależności zmian na pozycjach bitowych użyto testu χ^2 dla

- 8 bitów - 49 stopni swobody, 256 wyników;
- 6 bitów - 25 stopni swobody, 64 wyniki.

Oto wyniki testu:

Tabela 7.5.

$\overset{2}{A}$ 0:7		$\overset{8}{A}$ 0:5		$\overset{2}{I}$ 0:5		$\overset{4}{I}$ 0:5		$\overset{2}{K}^4$ 0:7		$\overset{2}{K}$ 0:7	
χ^2	p	χ^2	p	χ^2	p	χ^2	p	χ^2	p	χ^2	p.
7790	0	1333	0	4,58	1	4,58	1	6980	0	11200	0

Jak widać z tabeli 7.5 jedynie układy oparte na inwersji [Mor97] spełniają podane kryterium. Jednocześnie testy układów hybrydowych pokazały, że jedynie HGP, wyposażone w bloki nieliniowe, wykazują cechy spełniające kryterium niezależności bitów

Nieliniowość

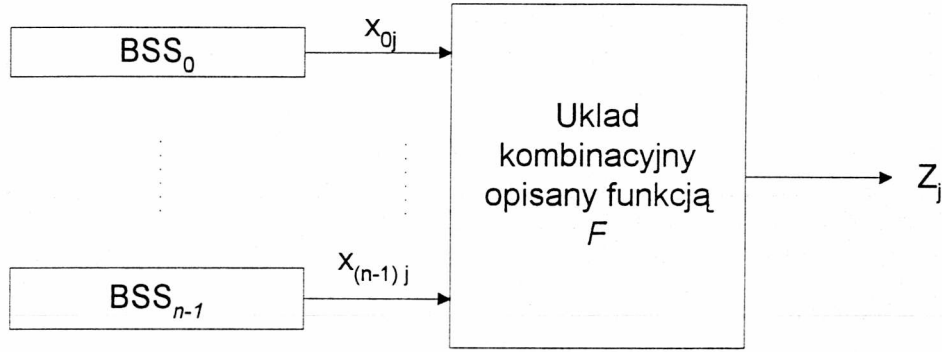
Nieliniowość s-skrzynek, zwana także odpornością na korelację, związana jest przede wszystkim z możliwością opisanego przekształcenia przez nie realizowanego za pomocą równań liniowych. Miarą nieliniowości układu jest dokładność odwzorowania wyznaczonego przez afiniczne przybliżenie układu.

Zagadnieniu wyznaczania takiego przybliżenia poświęcony jest następny podrozdział.

7.3 Odporność na korelację

Cecha ta dotycząca badania możliwości odwzorowania nieliniowych funkcji kombinacyjnych za pomocą układów równań liniowych pierwszy raz została zbadana w 1979r przez Blasera i Heinzmanna. Jednak teorię na ten temat w pełni rozwinął Siegenthaler w 1984r. W celu dokładnego zbadania statystycznych zależności wprowadzanych przez nieliniowe układy kombinacyjne¹⁰ zaproponował on informacyjno-teoretyczny model układu, przedstawiony na rysunku 7.6.

¹⁰ Każdy generator permutacji rozpatrywany w niniejszej pracy jest takim układem.



Rys. 7.6 Model informacyjno-teoretyczny użyty do zdefiniowania odporności na korelację

Źródłem sygnału dla układu kombinacyjnego jest w modelu n Źródeł Sekwencji Binarnych BSS_i (ang. *Binary Sequences Sources BSS*) generujących sekwencje niezależnych losowych zmiennych binarnych o rozkładzie równomiernym x_{ij} .

TWIERDZENIE 7.1. Dowolna nieliniowa funkcja F o równomiernym rozkładzie jest odporna na korelację m - tego rzędu, jeżeli wartość wyjściowa F (na rysunku jako Z_j) i wartość dowolnej zmiennej m - wejściowej są statystycznie niezależne [Rue86].

W praktyce problem weryfikacji odporności na korelację sprowadzany jest do znalezienia najlepszego liniowego (afinicznego) przybliżenia danej funkcji boolowskiej F .

Bardzo dobrym narzędziem matematycznym, pozwalającym na wyznaczenie takiego przybliżenia, jest transformata Walsh. W celu jej wyznaczenia definiuje się dwa n - wymiarowe wektory opisane nad $GF(2)$, $\bar{x} = (x_0, x_1, \dots, x_{n-1})$, odpowiadający n - bitowej dodatniej liczbie całkowitej, oraz $\bar{\omega} = (\omega_0, \omega_1, \dots, \omega_{n-1})$, który opisuje związaną z nią n - bitową dodatnią liczbę całkowitą. Transformata Walsh S_F dowolnej funkcji F o n zmiennych nad $GF(2)$, którą możemy traktować jako funkcję boolowską, zdefiniowana jest następująco

$$S_F(\omega) = \sum_{x=0}^{2^n-1} F(x)(-1)^{\bar{x}\bar{\omega}}, \quad (7.2)$$

gdzie iloczyn $\bar{x}\bar{\omega} = x_0\omega_0 + x_1\omega_1 + \dots + x_{n-1}\omega_{n-1}$ interpretowany jest jako liczba całkowita, odpowiadająca iloczynowi dwóch binarnych n - wymiarowych wektorów nad $GF(2)$.

Znając wartość transformaty Walsh można obliczyć przybliżoną postać funkcji $F(x)$ posługując się wzorem

$$F(x) = 2^{-n} \sum_{\omega=0}^{2^n-1} S_F(\omega)(-1)^{\bar{x}\bar{\omega}}. \quad (7.3)$$

Mając tak zdefiniowany problem oraz określone wzory służące do obliczenia transformaty Walsh, można wyznaczyć afiniczne przybliżenia każdej funkcji F , opisującej generator permutacji. Algorytm wyznaczania takiego przybliżenia jest następujący

- 1° Wyznaczyć transformatę Walsh S_F dla każdej składowej funkcji F_i układu generatora. W praktyce oznacza to wyznaczenie transformaty dla każdej wynikowej pozycji bitowej.
- 2° Znaleźć niezerowe wartości ω oznaczane jako ω_{m_i} , dla których $|S_{F_i}|$ jest maksymalna.
- 3° Jeśli $S_{F_i}(\omega_{m_i}) < 0$, to $\overline{\omega_{m_i}} \bar{x}$ jest najlepszą liniową aproksymacją, jeżeli zaś $S_{F_i}(\omega_{m_i}) > 0$, to najlepszą liniową aproksymacją jest $1 + \overline{\omega_{m_i}} \bar{x}$.

PRZYKŁAD 7.1. Niech dana będzie permutacja wygenerowana przez 4 bitowy hybrydowy generator permutacji o strukturze przetwarzania $\mathbb{I}_{0.3}^2 \langle x^4 + x + 1 \rangle \mapsto \mathbb{X}_{0.3} \langle 9 \rangle$. Wytworzona permutacja ma postać

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 9 & 8 & 0 & 7 & 4 & 2 & 14 & 15 & 6 & 11 & 5 & 12 & 3 & 13 & 10 & 1 \end{pmatrix}.$$

Niech y_i oznacza wartość i -tego bitu liczby uzyskanej po przekształceniu. Zgodnie z teorią istnieją więc 4 funkcje F , dla których należy obliczyć transformaty Walsh. Wartości y_i dla kolejnych x podano w tabeli 7.6.

Tabela 7.6.

$x =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
y_0	1	0	0	1	0	0	0	1	0	1	1	0	1	1	0	1
y_1	0	0	0	1	0	1	1	1	1	1	0	0	1	0	1	0
y_2	0	0	0	1	1	0	1	1	1	0	1	1	0	1	0	0
y_3	1	1	0	0	0	0	1	1	0	1	0	1	0	1	1	0

Wobec tego zgodnie z wzorem (7.1), postaci transformat Walsh S_{F_i} dla każdej z czterech funkcji są następujące

$$\begin{aligned} S_{F_0} &= 1 + (-1)^{\omega_0 + \omega_1} + (-1)^{\omega_0 + \omega_1 + \omega_2} + (-1)^{\omega_0 + \omega_3} + (-1)^{\omega_1 + \omega_3} + (-1)^{\omega_2 + \omega_3} + (-1)^{\omega_0 + \omega_1 + \omega_2 + \omega_3} , \\ S_{F_1} &= (-1)^{\omega_0 + \omega_1} + (-1)^{\omega_0 + \omega_2} + (-1)^{\omega_1 + \omega_2} + (-1)^{\omega_0 + \omega_1 + \omega_2} + (-1)^{\omega_3} + (-1)^{\omega_0 + \omega_3} + (-1)^{\omega_2 + \omega_3} + (-1)^{\omega_1 + \omega_2 + \omega_3} , \\ S_{F_2} &= (-1)^{\omega_0 + \omega_1} + (-1)^{\omega_2} + (-1)^{\omega_1 + \omega_2} + (-1)^{\omega_0 + \omega_1 + \omega_2} + (-1)^{\omega_3} + (-1)^{\omega_1 + \omega_3} + (-1)^{\omega_0 + \omega_1 + \omega_3} + (-1)^{\omega_0 + \omega_2 + \omega_3} , \end{aligned} \quad (7.4)$$

$$S_{F_3} = 1 + (-1)^{\omega_0} + (-1)^{\omega_1 + \omega_2} + (-1)^{\omega_0 + \omega_1 + \omega_2} + (-1)^{\omega_0 + \omega_3} + (-1)^{\omega_0 + \omega_1 + \omega_3} + (-1)^{\omega_0 + \omega_2 + \omega_3} + (-1)^{\omega_1 + \omega_2 + \omega_3},$$

co pozwala na wyznaczenie wartości transformaty dla dowolnego ω_i .

$\omega =$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_{F0}	-2	0	2	0	2	0	-2	-2	0	-2	4	2	0	2	4
S_{F1}	0	0	0	-2	-2	2	2	0	-4	-4	0	-2	2	-2	2
S_{F2}	0	-2	2	0	0	-2	2	0	0	-2	2	-4	-4	2	2
S_{F3}	-2	0	-2	0	-2	4	2	0	2	0	2	0	2	4	-2

skąd otrzymuje się $\omega_{m_0} = 11$, $\omega_{m_1} = 9$, $\omega_{m_2} = 12$, $\omega_{m_3} = 6$. Czyli najlepsze afiniczne przybliżenie stanowi układ równań o postaci

$$\begin{aligned} y_0 &= x_0 + x_1 + x_3 + 1, \\ y_1 &= x_0 + x_3, \\ y_2 &= x_2 + x_3, \\ y_3 &= x_1 + x_2 + 1, \end{aligned} \tag{7.5}$$

co po konwersji z wektorów nad $GF(2)$ do postaci dziesiętnej liczb całkowitych daje

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
y	9	10	0	3	5	6	12	15	14	13	7	4	2	1	11	8

Liczba elementów permutacji (podana jako procent liczby elementów pełnej permutacji) wygenerowanych identycznie przez afiniczne przybliżenie generatora oraz badany generator została przyjęta w pracy jako miara odporności układów na korelację. Stuprocentowa zgodność oznacza, że afiniczne przybliżenie generatora działa dokładnie tak samo jak generator, co wskazuje na zupełny brak odporności na korelację.

W przykładzie 7.1 uzyskano ok. 19% zgodność z permutacją wygenerowaną za pomocą badanego generatora. Czyli trzy z szesnastu elementów permutacji z przykładu 7.1 są przekształcane identycznie jak w oryginalnym generatorze.

Przedstawiona powyżej metoda dość dobrze pozwala na zbadanie odporności na korelację układów permutacyjnych, a przebadane przy jej pomocy układy permutacyjne używane w s-skrzynkach metody DES wykazują w niektórych przypadkach nawet ok. 90% zgodność z uzyskanymi przybliżeniami [Rue86].

W ramach realizacji niniejszej pracy napisano w języku C++ procedurę programową, realizującą badanie podatności na korelację dowolnego generatora permutacji.

Jest oczywiste, że w przypadku permutacji uzyskanych za pomocą układów opisanych równaniami liniowymi nad $GF(2)$ metoda daje 100% zgodność równań otrzymanych w wyniku aproksymacji z badanymi.

A oto tabela 7.7, zawierająca wyniki zgodności procentowej elementów permutacji wygenerowanych za pomocą afinicznych przybliżeń funkcji opisujących generatory zaprojektowane w poprzednich rozdziałach.

Tabela 7.7.

$\overset{2}{A}_{0:7}$	$\overset{8}{A}_{0:5}$	$\overset{2}{I}_{0:5}$	$\overset{4}{I}_{0:5}$	$\overset{2}{S^4}_{0:7}$	$\overset{2}{S}_{0:7}$
100%	100%	3%	3%	100%	100%

W tabeli 7.8 podane są wartości odpowiadające odporności na korelację układów hybrydowych.

Tabela 7.8.

$\overset{2}{A}_{0:7} \rightarrow \overset{2}{I}_{0:5} \rightarrow \overset{2}{I}_{2:7}$	$\overset{8}{A}_{0:5} \rightarrow \overset{2}{I}_{0:5}$	$\overset{2}{I}_{0:5} \rightarrow \overset{4}{I}_{0:5}$	$\overset{4}{I}_{0:5} \rightarrow \overset{8}{A}_{0:5}$	$\overset{2}{I}_{0:5} \rightarrow \overset{2}{S^4}_{0:7} \rightarrow \overset{2}{I}_{2:7}$	$\overset{2}{I}_{0:5} \rightarrow \overset{2}{S}_{0:7}$
1%	10%	7%	7%	3%	4%

Połączenie układów o dużej odporności na korelację z układem o liniowym charakterze przekształcenia daje w efekcie układ hybrydowy o odporności na korelację zbliżonej do posiadanej przez blok o charakterze nieliniowym.

7.4 Generowanie ciągów losowych i pseudolosowych

Wytwarzanie ciągów losowych lub pseudolosowych jest nieodłączną czynnością nieomal każdego systemu kryptograficznego. Różnica między generatorami pierwszego i drugiego typu polega na tym, że generator liczb losowych (*ang. Random Number Generator RNG*) wytwarza sekwencję statystycznie niezależnych i symetrycznie rozłożonych liczbowych zmiennych losowych¹¹. Generator liczb pseudolosowych (*ang. Pseudo-Random Number*

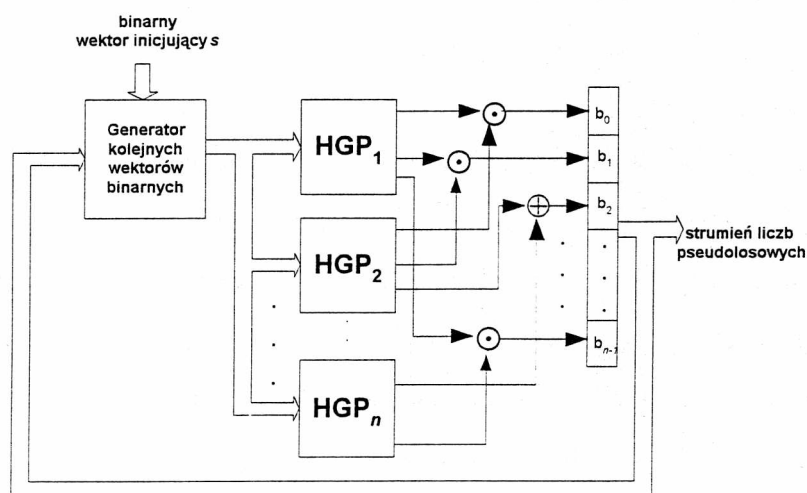
¹¹ Takim generatorem jest na przykład BSS przedstawiany w rozdziale o odporności na korelację

Generator PRNG) to urządzenie, które ma za zadanie deterministycznie wytwarzać ciąg liczb, możliwie najwierniej przypominający ciąg losowy.

W kryptografii generatory typu RN używane są do wytwarzania tajnych kluczy w symetrycznych systemach szyfrujących (takich jak np. DES) lub do generowania parametrów stosowanych w systemach z kluczem publicznym (np. RSA) [Mau92]. Generatory ciągów pseudolosowych są wykorzystywane do uzyskiwania klucza szyfrującego dla szyfrów strumieniowych.

Konstruowanie generatorów obydwu typów polega przeważnie na łączeniu ze sobą różnych generatorów permutacji tak, aby skonstruowane w ten sposób urządzenie mogło wytworzyć ciąg (sekwencję liczb, bitów) odpowiadający danym warunkom losowości lub pseudolosowości. Działanie konstruktora takiego urządzenia ma charakter empiryczny, bowiem nie sposób precyzyjnie powiedzieć, jak łączyć generatory permutacji, aby uzyskać zadany efekt. Szczególnie, że powstałe w wyniku takiego działania urządzenie musi nie tylko generować ciąg o zadanych parametrach, ale także być w dużym stopniu tworem oryginalnym i niepowtarzalnym. Każde bowiem podobieństwo do innych już znanych generatorów może spowodować jego jednoznaczna identyfikację dokonaną przez kryptoanalitików. [Sch95]

Jedyną właściwie zasadą, którą można podać jako ogólny sposób konstruowania RNG lub PRNG jest to, aby łączyć ze sobą różne generatory permutacji (operujące także w różnych systemach algebraicznych) używając do tego różnych działań arytmetycznych (także innych niż użyte w generatorach permutacji) tak, aby konstrukcja w ten sposób powstała miała możliwie najtrudniejszą do analizy strukturę [Rue86].



Rys. 7.7 Przykład generatora liczb pseudolosowych z zastosowaniem hybrydowych generatorów permutacji HGP

Oto podstawowe wymagania dotyczące generatora binarnych sekwencji pseudolosowych :

- bardzo długi okres sekwencji (obecnie przyjmuje się nawet co najmniej 2^{35}), brak powtórzeń;
- możliwie największa złożoność liniowa¹²;
- równomierny rozkład częstości występowania wszystkich możliwych ciągów k - bitowych;
- każdy bit ciągu musi być w możliwie najbardziej złożony sposób zależny od wszystkich lub większości bitów;
- brak nadmiarowości w podciągach;
- odporność na korelację, spełnianie kryterium lawiny itp.;
- żaden atak polegający na podziale generatora i analizie składowych elementów (*ang. divide and conquer*) nie powinien wpływać na bezpieczeństwo całego układu.

Weryfikacji cech losowości lub pseudolosowości sekwencji wygenerowanych przez RNG lub PRNG można dokonać stosując specjalistyczne testy statystyczne. Podstawą tych testów jest hipoteza zerowa, w ramach której zakłada się, iż obserwowana sekwencja ma rozkład losowy.

Oto kilka podstawowych sposobów pozwalających na praktyczną weryfikację jakości sekwencji wygenerowanych przez RNP lub PRNG¹³

- obliczenie entropii ciągu H np. entropia kodu źródłowego programu w języku C wynosi ok. 4,9 bita /znak 8-bitowy;
- określenie podatności wygenerowanego ciągu na kompresję. Ciąg o rozkładzie równomiernym kodów ASCII nie powinien być podatny na kompresję. Podatność typowego pliku tekstowego wynosi ok..50%;
- obliczenie wartości χ^2 dla wygenerowanego ciągu. Pozwala to na określenie różnic między wartościami obserwowanymi w ciągu i oczekiwanymi w ciągu losowym;
- obliczenie wartości średniej oraz innych parametrów statystycznych dla sekwencji;
- test π Monte Carlo określający losowość na podstawie sekwencji 32 bitowych. Sekwencja jest losowa, gdy wartość testu jest zbliżona do wartości π ;

¹²Złożoność liniowa to długość najkrótszego LFSR, którym można wytworzyć taki sam ciąg jak badany.

¹³ Więcej na ten temat można znaleźć [Mau92], [Gro71], [Mrs84].

- pomiar wartości współczynnika korelacji szeregowej. Sprawdzana jest zależność wartości kolejnych sekwencji bitów od wartości sekwencji je poprzedzających. Dla sekwencji losowych wartość ta powinna być zbliżona do zera.

Oto przykłady wartości parametrów podanych testów wyznaczone dla generatora o dobrych własnościach pseudolosowych oraz dla pliku zawierającego tekst w języku angielskim.

Tabela 7.9.

	Entropia	Podatność na kompresję	χ^2	Wartość średnia	Test π	Korelacja szeregową
Dobry generator	7,993	0	368,35	127,65	3,1491	-0,000236
Plik tekstowy	4,28	46%	249455	74,22	4	0,532

Warto zaznaczyć, że weryfikacji cech strumienia bitów należy dokonywać zawsze analizując możliwie najdłuższe sekwencje wygenerowane. Okresy powtarzalności przeciętnych generatorów są rzędu gigabajtów i więcej, dlatego też nie ma sensu wnioskowanie o cechach na podstawie krótszych sekwencji. Na przykład w zestawie testów o nazwie „DIEHARD” dostępnym w sieci Internet sugerowana długość sekwencji wynosi ok. 11MB. [WWWTR]

Powyżej przedstawione metody oraz wiele innych wraz z opisami stosowania, a także często w postaci programów gotowych do użycia, można znaleźć w różnych materiałach dostępnych w Internecie dotyczących testowania własności generatorów sekwencji losowych i pseudolosowych oraz w pracach [Mau92], [Mrs84], [Mrs93] i wielu innych.

8. Przykłady zastosowań - (symulacja)

W rozdziale niniejszym zaprezentowano opis i wyniki testów przykładowego kryptosystemu możliwego do realizacji w wersji sprzętowej. Do jego realizacji posłużą wcześniej opisane bloki permutacyjne połączone i zastosowane zgodnie z regułami przedstawionymi w rozdziałach 6 i 7. Z kolei przedstawione zostaną wyniki działania symulacji komputerowej działania tak zaprojektowanego kryptosystemu.

8.1 Założenia

W celu weryfikacji tez zawartych w pracy dotyczących przydatności zaprojektowanych generatorów dla potrzeb kryptograficznych zaprojektowano kryptosystem, który w całości może być zrealizowanych w postaci sprzętowej z zastosowaniem układów cyfrowych o architekturze równoległej. W celach porównawczych struktura zaprojektowanego systemu kryptograficznego zbliżona jest do budowy DES [Sch95].

Aby przetestować możliwości generowania sekwencji pseudolosowych za pomocą hybrydowych generatorów permutacji, zaprojektowano jednocześnie generator liczb pseudolosowych dla potrzeb sieci przekształceń szyfrująco - deszyfrujących.

8.2 Kryptosystem

8.2.1 Budowa i działanie

Budowa

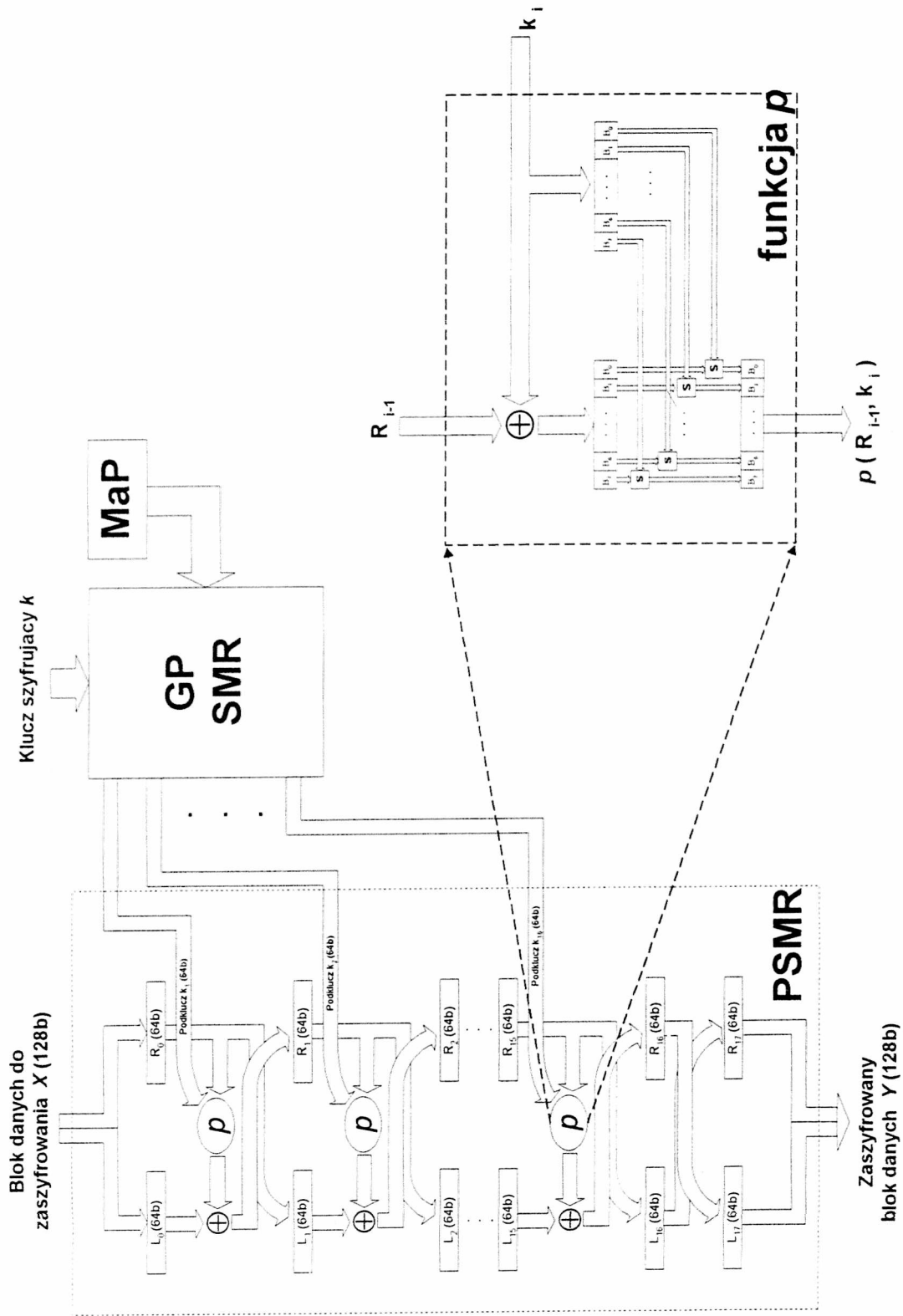
Ogólny schemat struktury zaprojektowanego systemu przedstawiono na rys.8.1. Przyjęto następujące oznaczenia :

L_i , R_i - odpowiednio lewa i prawa połowa 128-bitowego bloku danych ,

p - funkcja realizująca operacje podstawienia z zastosowaniem zaprojektowanych generatorów permutacji ,

GPSMR - Generator Podkluczy Sieci Mieszająco-Rozpraszającej ,

MaP - Matryca Permutacyjna (256-elementowa permutacja).



Rys. 8.1. Schemat systemu kryptograficznego zaprojektowanego do symulacyjnej weryfikacji zaprojektowanych generatorów permutacji wraz ze schematem funkcji p

Szyfrowanie / deszyfrowanie realizowane jest metodą blokową w 16 identycznych cyklach. Każdy blok danych X ma 128 bitów. Klucz szyfrujący k jest również 128-bitowy. Głównym elementem kryptosystemu jest sieć przekształceń bitowych podobna do prezentowanej w poprzednim rozdziale sieci permutacyjno-podstawieniowej. Główna różnica polega na tym, że efekt mieszania i rozpraszania informacji uzyskuje się w prezentowanym systemie przy pomocy wykorzystanych w funkcji p generatorów permutacji działających jako s-skrzynki o odpowiednich właściwościach¹. Z tego powodu struktura ta w dalszym ciągu pracy określana jest jako Permutacyjna Sieć Miesząco-Rozpraszająca PSMR.

Dodatkowo w kryptosystemie zastosowano generator 64-bitowych podkluczy k_i dla poszczególnych faz przetwarzania bloku danych. Elementami inicjującymi działanie tego generatora są: klucz szyfrujący k oraz tzw. Matryca Permutacyjna MaP będąca swego rodzaju dodatkowym kluczem wbudowanym w kryptosystem², o wartości niezmienniej lub zmienianej co jakiś czas.

Zarówno szyfrowanie jak i deszyfrowanie odbywa się za pomocą tego samego algorytmu, z tym, że jak dla DES, podczas deszyfrowania kolejność podkluczy jest odwrotna czyli k_{16} do k_1 .

Działanie

System działa w pięciu fazach. W pierwszej - przygotowawczej, na podstawie wartości klucza k i MaP wygenerowane zostają podklucze k_i . W zależności od typu procesu (szyfrowanie / deszyfrowanie) ustalana jest także kolejność ich występowania.

Druga faza to podział bloku danych X na dwie równe połowy, każda po 64 bity. Trzecia faza składa się z 16 identycznych cykli. W każdym cyklu aktualna lewa połowa bloku L_{i-1} sumowana jest mod 2 z wynikiem działania funkcji $p(R_{i-1}, k_i)$ oraz wstawiana w miejsce prawej połowy bloku, czyli jako $\underline{R_i}$, $R_i = L_{i-1} \oplus p(R_{i-1}, k_i)$. Z kolei miejsce lewej połowy bloku L_i zajmuje wartość z R_{i-1} , to znaczy $L_i = R_{i-1}$. W ramach funkcji p sumowane są poelementowo mod 2 wartości R_{i-1} i k_i . Następnie wynik tego sumowania jest przetwarzany przy pomocy zestawu s-skrzynek. Parametrami sterującymi dla s-skrzynek są wartości poszczególnych bajtów $B_j, j=0 \dots 7$ podklucza k_i .

¹ Takie działanie sugeruje Kryterium Lawiny, bowiem s-skrzynki spełniające ściśle Kryterium Lawiny powinny posiadać odpowiednie własności miesząco-rozpraszające [Rit98].

² Pomysł ten zapożyczony został z projektu profesjonalnego systemu kryptograficznego, w którego realizacji uczestniczył autor niniejszej pracy [Top95][Top95:1][Wol94][Wol94:1].

Ostatnie dwie fazy to zamiana miejscami połówek bloku, czyli $L_{17} = L_{16}$ i połączenie ich w jeden 128-bitowy blok danych Y .

8.2.2 S-skrzynki

Budowa

Aby realizować szyfrowanie i deszyfrowanie za pomocą tego samego algorytmu wykorzystując PSMR, zaprojektowano 8-bitowe s-skrzynki jako samoodwracalne hybrydowe generatory permutacji. W ten sposób dla tych samych parametrów sterujących s-skrzynka naprzemian generuje przekształcenie wzajemnie odwracalne.

Kolejnym założeniem zrealizowanym w opisywanych elementach są ich odpowiednie własności kryptograficzne. Wykorzystując wyniki badań opisane w poprzednim rozdziale zrealizowano s-skrzynka wykorzystywany w PSMR w postaci hybrydowego generatora permutacji, generującego permutację samoodwracalną z dwoma dynamicznie definiowanymi składowymi blokami permutacyjnymi (na schemacie jako Adresowalne Generatory Permutacji AGP) i trzema blokami permutacyjnymi o ustalonej strukturze SHGP. Taki bowiem układ s-skrzynki gwarantuje odpowiednie własności zgodnie z kryteriami prezentowanymi w rozdz. 7 jednocześnie zwiększając nieokreśloność struktury szyfrującej³.

Strukturę przetwarzania s-skrzynki PSMR można przedstawić w postaci

$$\underset{i:j}{F_1^k} \rightarrow \underset{l:m}{F_2^n} \rightarrow \left[\overset{2}{\text{---}} \overset{2}{\text{---}} \overset{2}{\text{---}} \overset{2}{\text{---}} \overset{2}{\text{---}} \right] \rightarrow \underset{0:7}{K^4} \rightarrow \underset{2:7}{\text{---}} \rightarrow \underset{0:3}{\text{---}} \rightarrow \underset{l:m}{F_2^{n-1}} \rightarrow \underset{i:j}{F_1^{k-1}}, \quad (8.1)$$

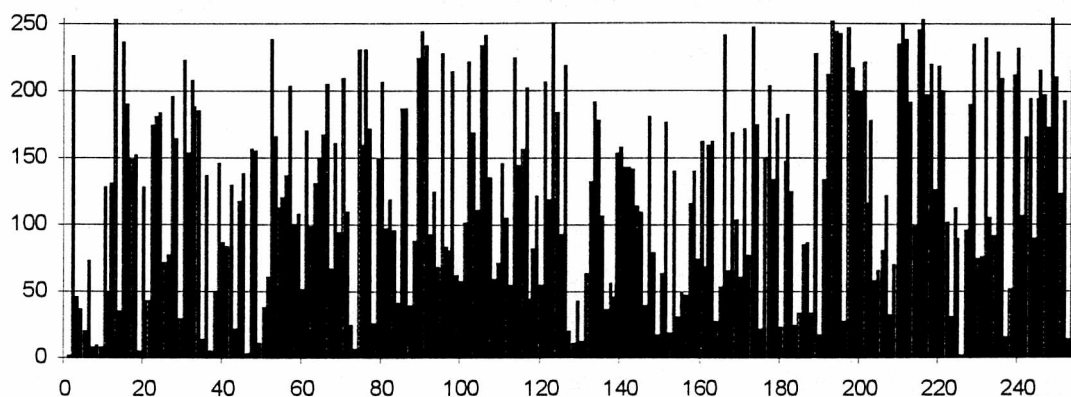
gdzie

$$\underset{i:j}{F_1^k}, \underset{l:m}{F_2^n} - \text{funkcje definiowane dynamicznie,}$$

$$\underset{i:j}{F_2^{n-1}}, \underset{l:m}{F_1^{k-1}} - \text{odwrotności funkcji definiowanych dynamicznie.}$$

Linia przerywaną we wzorze 8.1 oznaczone są przekształcenia stałe dla każdego s-skrzynki SHGP.

O wyborze tej struktury zdecydował histogram permutacji wygenerowanej przez SHGP.



Rys. 8.2. Histogram permutacji wygenerowanej przez SHGP.

Na uwagę w powyższym histogramie zasługuje to, że mimo iż SHGP złożony jest w większości z układów generujących przekształcenie o charakterze nieliniowych, histogram wykazuje duże podobieństwo do histogramów permutacji uzyskanych przy pomocy przekształceń afinicznych. Zgodnie z wynikami poprzednich badań gwarantowało to z jednej strony dobre cechy rozpraszające SHGP, a z drugiej odporność na korelację.

Własności mieszająco-rozpraszające SHGP zmierzone w trakcie badań przedstawiono w tabelce poniżej. Potwierdzają one wnioski wyciągnięte w trakcie analizy histogramu. Szczególnie zwraca uwagę wysoka odporność na korelację zapewniona przez nieliniowy charakter przekształcenia oraz wartości innych parametrów zbliżone do optymalnych.

Tabela 8.1

AC		CC		BIC ⁴ (49 st. swobody)		ODPORNOŚĆ NA KORELACJĘ
il. Bitów	%	il. Bitów	%	χ^2	p	%
3,8	47	3,9	49	37,63	0,88	1

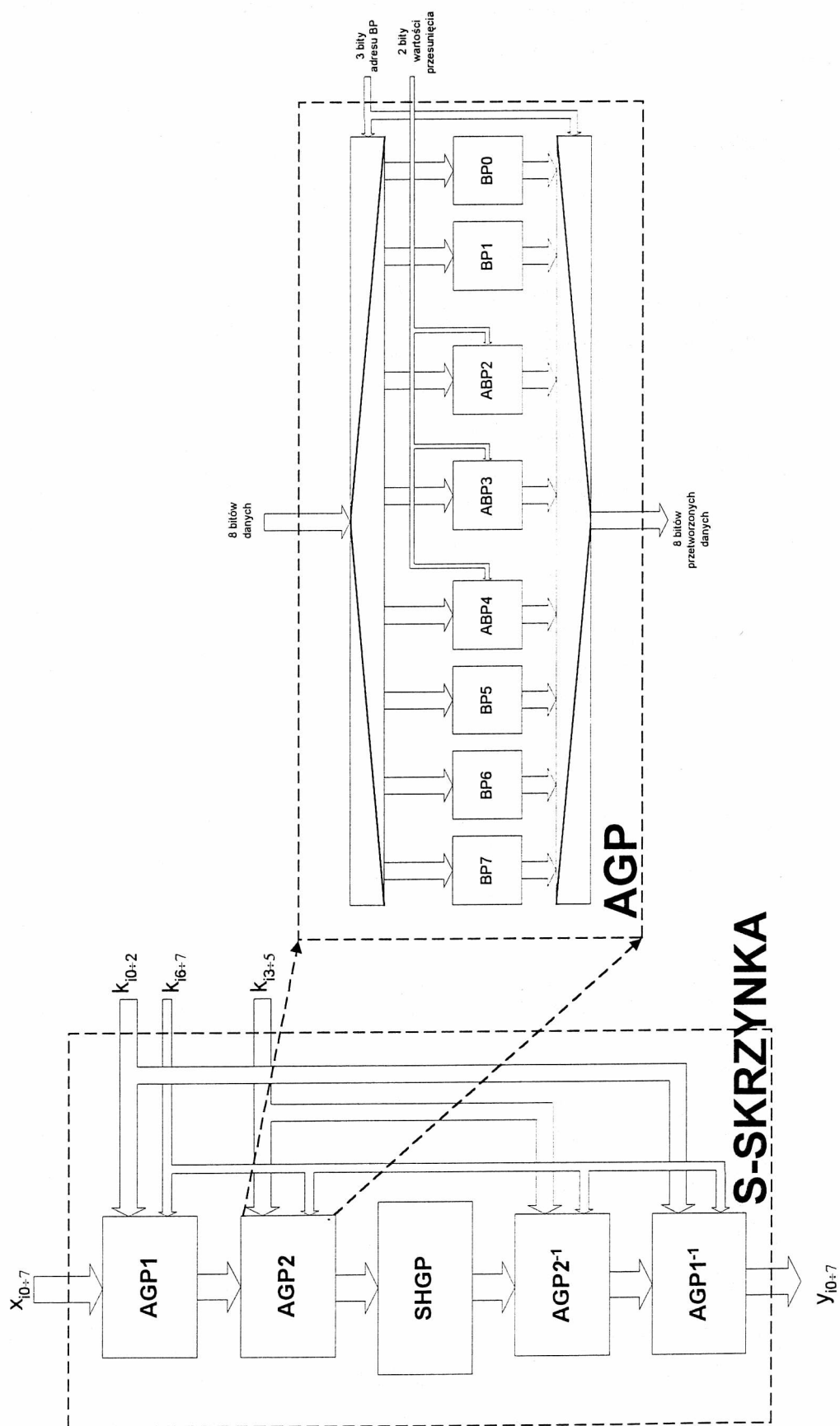
Jak łatwo obliczyć z wzoru 6.8, że dla $m = 8$, $k = 2$, $l_1 = 2$, $l_2 = 4$ w ramach PSMR przekształcenia mogą być realizowane za pomocą $n_p = 8^2 \cdot (3 \cdot 5) = 960$ s-skrzynek o różnych strukturach. Oczywiście w niektórych z nich może dojść do wygenerowania permutacji identycznościowej (np. w przypadku gdy AGP_1 jest 8-bitowym przekształceniem afinicznym,

³ Koncepcję stosowania losowo wybieranych s-skrzynek przedstawiono już w latach 80-tych. [Rit98]. Jednocześnie w ostatnich latach powstał pomysł wzmocnienia sieci permutacyjno-podstawieniowych za pomocą mechanizmów uzależniających strukturę s-skrzynek od wartości klucza szyfrującego [Sch95].

⁴ Wskaźnik dla Kryterium Niezależności Bitów - patrz poprzedni rozdział.

zaś AGP_2 jest odwrotnością tego przekształcenia). Wtedy jednak s-skrzynka generuje permutację w oparciu o przekształcenia SHGP. Schemat s-skrzynki zastosowanej w PSMR znajduje się na rys. 8.3. Przyjęto następujące oznaczenia:

- AGP< i > - Adresowalny Generator Permutacji. Układ z możliwością wyboru jednego z ośmiu bloków permutacyjnych ,
- SHGP - Stały Hybrydowy Generator Permutacji. HGP o strukturze nieziennej w trakcie procesu szyfrowania/deszyfrowania,
- ABP< i > - Adresowalny Blok Permutacyjny. Układ realizujący przełączanie w ramach na kolejne k -bitowe podbloki bitów w ramach m -bitowego bloku danych ($k = \{4, 6\}$, $m = 8$) - np. dla układu realizującego inwersję w $GF(2^6)$).



Rys. 8.3. Schemat s-skrzynki zastosowanej w PSMR

Działanie

W pierwszej kolejności wartości podbloków bitowych k_{0+2} i k_{3+5} ustalają strukturę odpowiednio AGP_1 i AGP_2 . Jednocześnie te same wartości ustalają ich odpowiedniki AGP_1^{-1} i AGP_2^{-1} . Dzięki temu bajt danych X jest kierowany do wejść odpowiednich bloków permutacyjnych.

Oto kody przekształceń występujących w ramach AGP :

Tabela 8.2.

KOD	PRZEKSZTAŁCENIE	KOD	PRZEKSZTAŁCENIE
000	$\overset{2}{A}_{0:7}$ (8 bitów)	100	$\overset{4}{I}_{i:(i+5)}$ (6 bitów)
001	$\overset{8}{A}_{0:5}$ (6 bitów)	101	$\overset{2}{K}_{0:7}^{-4}$ (8 bitów)
010	$\overset{2}{I}_{i:(i+5)}$ (6 bitów)	110	$\overset{2}{K}_{0:7}$ (8 bitów)
011	$\overset{2}{I}_{j:(j+3)}$ (4 bity)	111	$\overset{2}{A}_{0:7}^{-1}$ (8 bitów)

Dobór kodów i funkcji zaprezentowany w powyższej tabelce został zrealizowany doświadczalnie. Czynniki decydującymi o przyporządkowaniu były własności kryptograficzne s-skrzynki w odniesieniu do częstości wystąpień poszczególnych wartości kodów sterujących występujących w podkluczach k_i .

Dodatkowo w przypadku wyboru bloku o zakresie przetwarzania mniejszym niż 8 bitów na podstawie dwóch najbardziej znaczących bitów określany jest zakres przetwarzanych bitów w ramach 8-bitowego bloku. Po przetworzeniu przez AGP_1 , AGP_2 i SHGP blok 8-bitowy jest przetwarzany kolejno przez hybrydowe generatory permutacji, realizujące przekształcenia odwrotne do AGP_1 i AGP_2 .

8.2.3 Generator Podkluczy PSMR

Budowa

Generowanie podkluczy dla procesu szyfrowania jest działaniem wykonywanym raz w ciągu całej sesji szyfrowania/desyfrowania. Dlatego też czynność ta nie jest tak obciążona ograniczeniami czasowymi jak zachodzące w sieci PSMR. Z tego powodu możliwe jest użycie do ich realizacji rejestrów przesuwających, układów mikroprogramowalnych lub jakichkolwiek innych komponentów szeregowo przetwarzających dane. Mimo to w symulowanym

kryptosystemie zaprojektowano taki generator, który nie wykorzystuje ww. czasochłonnych technik.

Generator składa się z jednakowych modułów UGP, które przekształcając elementy MaP zgrupowane w 16-bajtowe bloki oznaczone MaP_{16B} wytwarzają poszczególne podklucze $k_1 \div k_{16}$ dla PSMR. Parametrami dla UGP są bajty klucza szyfrującego k przekształcone przez układ UPK do postaci $B_{k0} \div B_{k15}$. W ramach UGP oraz UPK wykorzystano s-skrzynce o strukturze identycznej jak w PSMR.

Działanie

Działanie generatora rozpoczyna się w chwili wpisania klucza szyfrującego k . Następnie kolejne bajty klucza są przekształcane wg wzorów

$$\begin{aligned} B_{k15} &= S(B_{15}, B_0) \oplus B_{15} \oplus B_0, \\ B_{k14} &= S(B_{14}, B_0) \oplus B_{14} \oplus B_{k15}, \\ &\vdots \\ B_{k15} &= S(B_0, B_{k1}) \oplus B_0 \oplus B_{k1}, \end{aligned} \quad (9.1)$$

gdzie

$S(x, P)$ - funkcja s-skrzynki przekształcająca wartość x na podstawie wartości parametru P .

Z kolei na podstawie wartości bajtów klucza B_k przekształcane są bajty MaP za pomocą transformacji

$$B'_i = S(B_i, B_{ki}) \oplus B_i \rightarrow B''_i = S(B'_i, B_{ki}) \oplus B'_i \rightarrow B'''_i = S(B''_i, B_{kj}) \oplus B''_i, \quad (9.2)$$

gdzie

B_i - numer bajtu w 16-bajtowym bloku MaP,

B'_i, B''_i, B'''_i - wartości bajtu B_i po przekształceniu przez kolejne warstwy PBP,

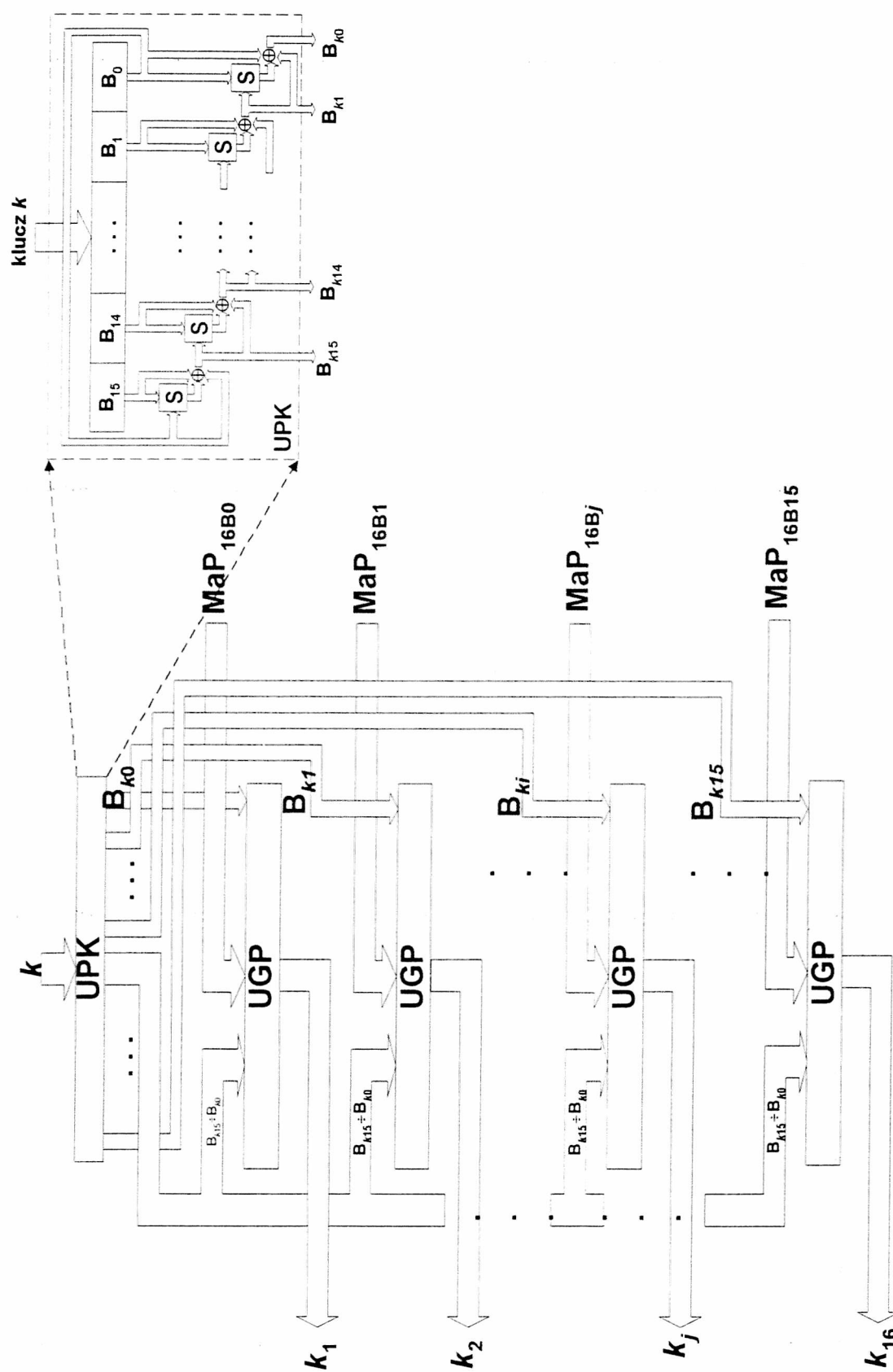
B_{ki} - i -ty bajt klucza,

j - numer 16-bajtowego bloku MaP.

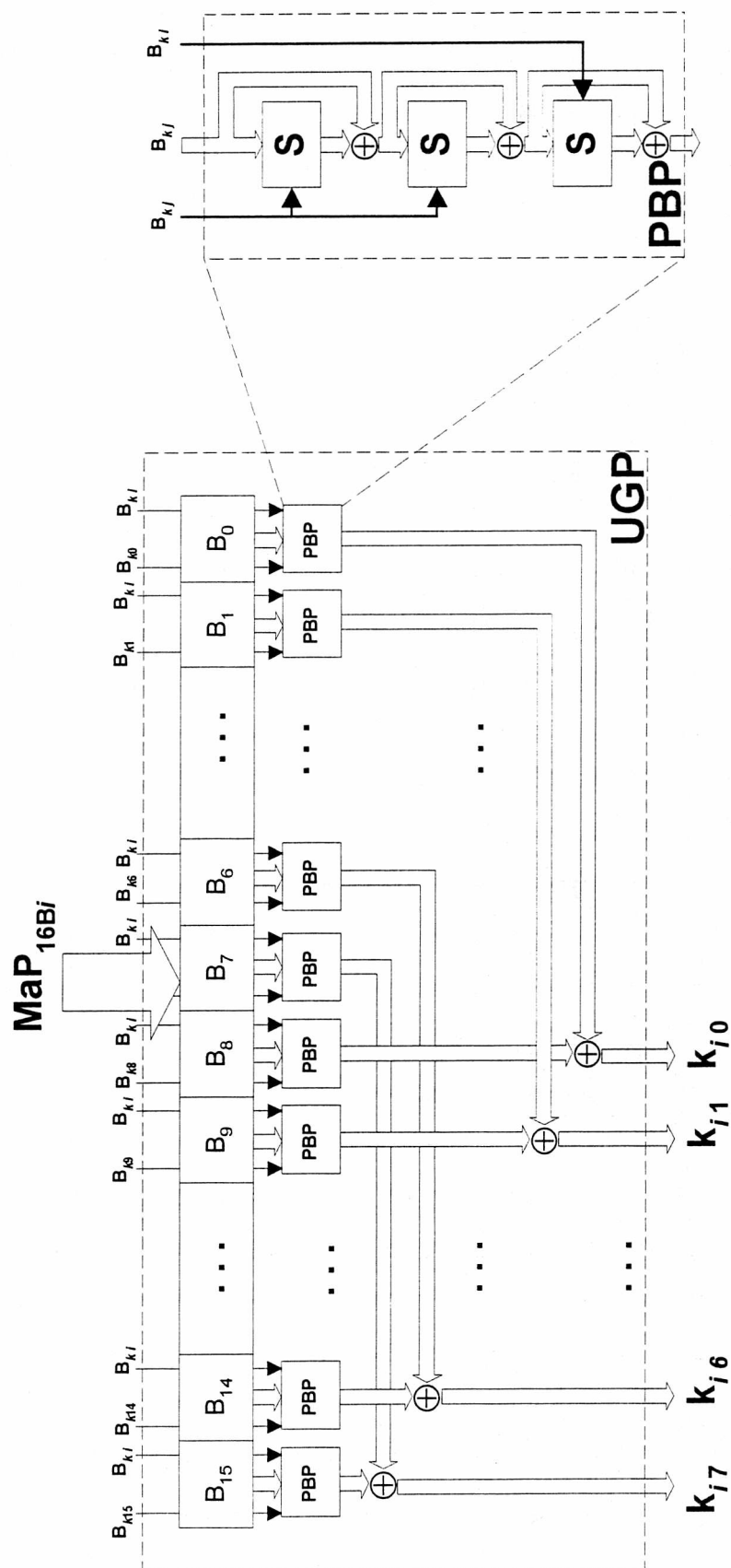
Ostatnią fazą przygotowania zestawu 8-bajtowych podkluczy dla PSMR jest sumowanie bajtów bloku MaP ze sobą wg wzoru

$$k_{jl} = B'''_l \oplus B'''_{l+8} \text{ dla } l = 0 \dots 7, \quad (9.3)$$

gdzie k_{jl} - l -ty bajt j -tego podklucza PSMR.



Rys. 8.4. Budowa Generators Podkluczy dla PSIMR oraz Układu Przekształcającego Klucz UPK



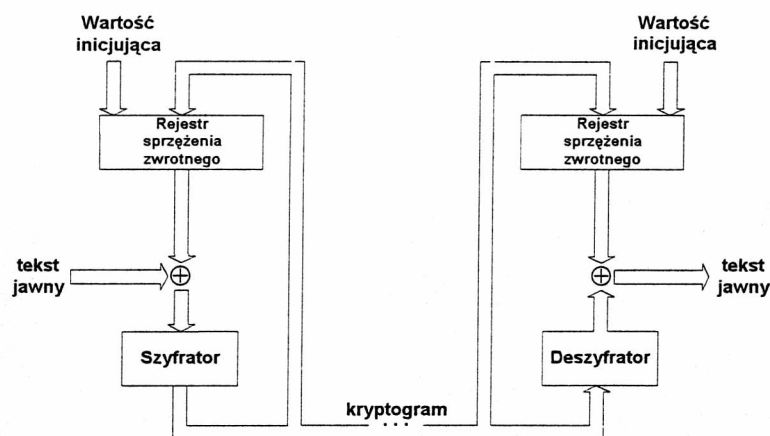
Rys. 8.5. Budowa Układu Generującego Podklucz k_i oraz Potrójnego Bloku Podstawieniowego PBP

8.3 Realizacja programowa

Program symulujący działanie powyżej przedstawionego kryptosystemu zrealizowano w języku C++ w postaci szyfrującej pliki aplikacji systemu DOS. Tabulogram procedury realizującej PSMR, s-skrzynka oraz Generator Podkluczy znajduje się na końcu niniejszej pracy.

8.4 Wyniki testów

Poniżej przedstawiono testowe dane porównawcze kryptogramów uzyskanych przez zaszyfrowanie różnych plików za pomocą różnych kryptosystemów DES, IDEA oraz PSMR. Do testów zastosowano wersje kryptosystemów pracujące w trybie CBC (ze sprzężeniem zwrotnym), co pozwoliło na sprawdzenie jakości generatora podkluczy w każdym z tych systemów kryptograficznych poprzez zaszyfrowanie pliku wypełnionego wzorcem jednego typu (np. tylko litera „b”).



Rys. 8.6. Praca kryptosystemu w trybie ze sprzężeniem zwrotnym CBC.

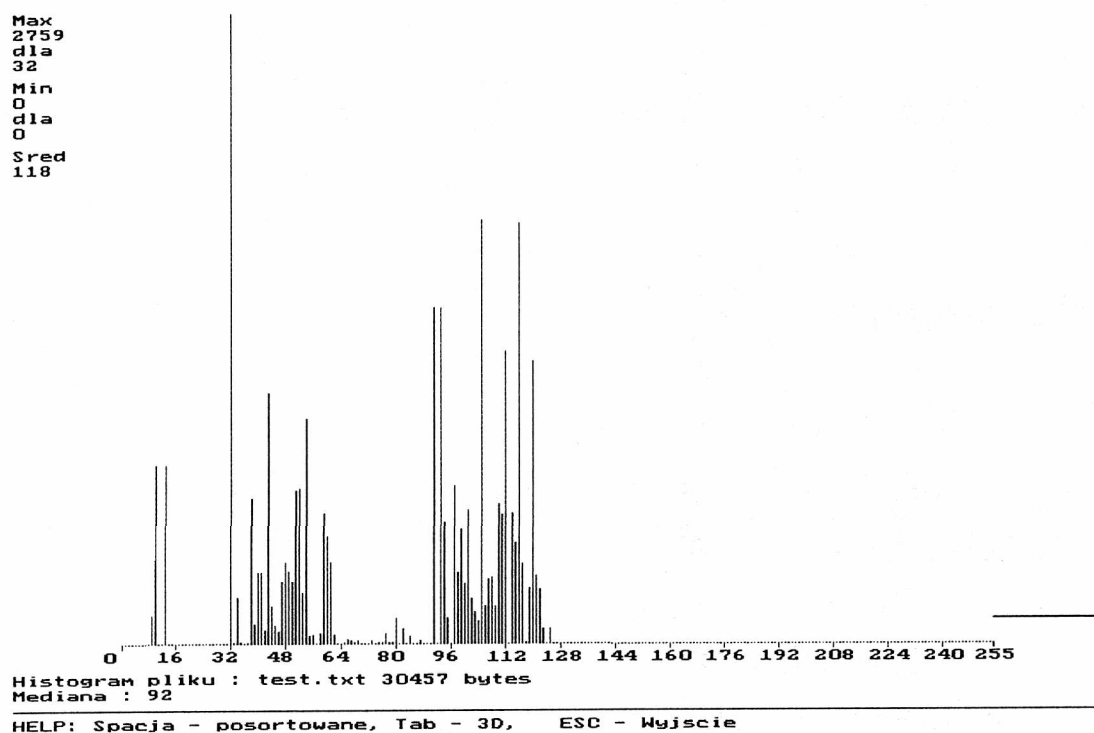
W przypadku IDEA i PSMR szyfrowano kluczem zawierającym 16, zaś przy DES 8 znaków „@”(kod ASCII 64).

Charakterystyki częstości występowania znaków kodów ASCII w plikach (histogram z lewej strony) oraz rozkład par kodów (histogram z prawej).

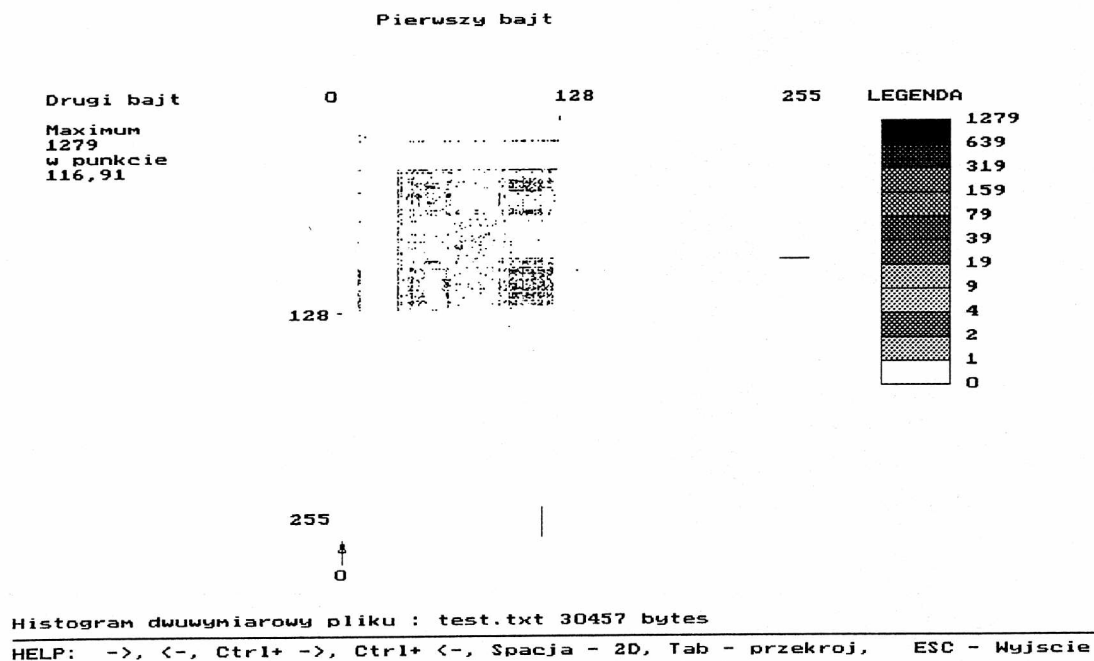
Podczas testów zbadano własności kryptogramów pliku zawierającego dokładnie 1.000.000 liter „b” (kod ASCII 98).

Histogramy

TEKST JAWNY

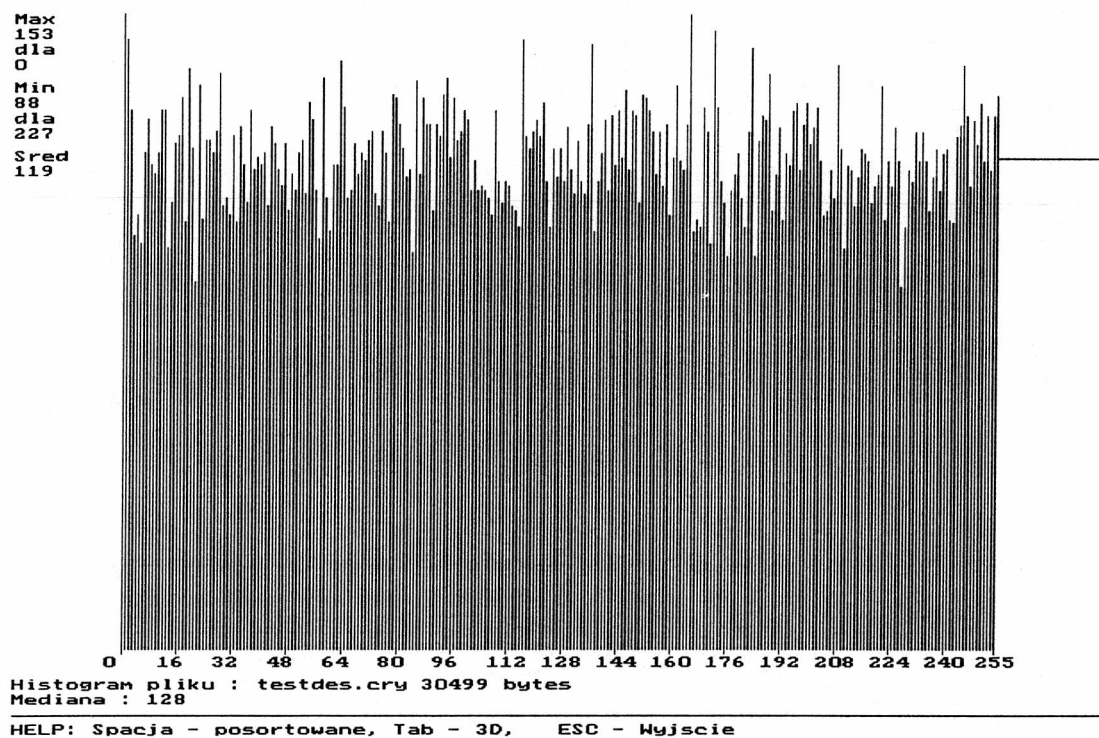


Rys. 8.7. Rozkład kodów ASCII w pliku przed zaszyfrowaniem

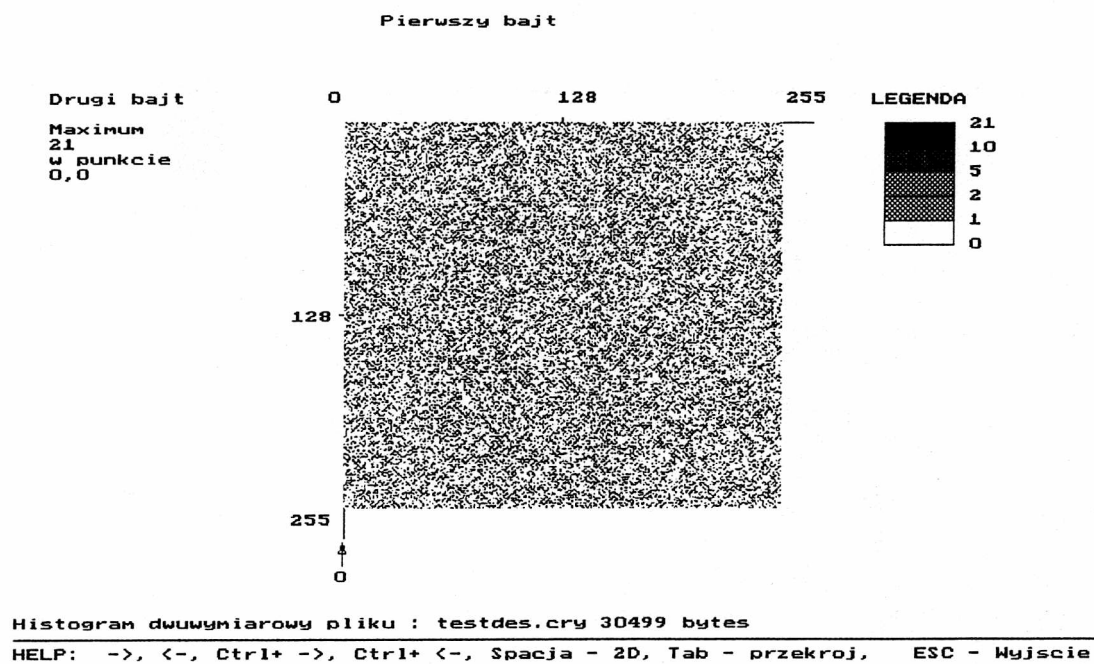


Rys. 8.8. Rozkład par kodów ASCII w pliku przed zaszyfrowaniem

KRYPTOGRAM - DES

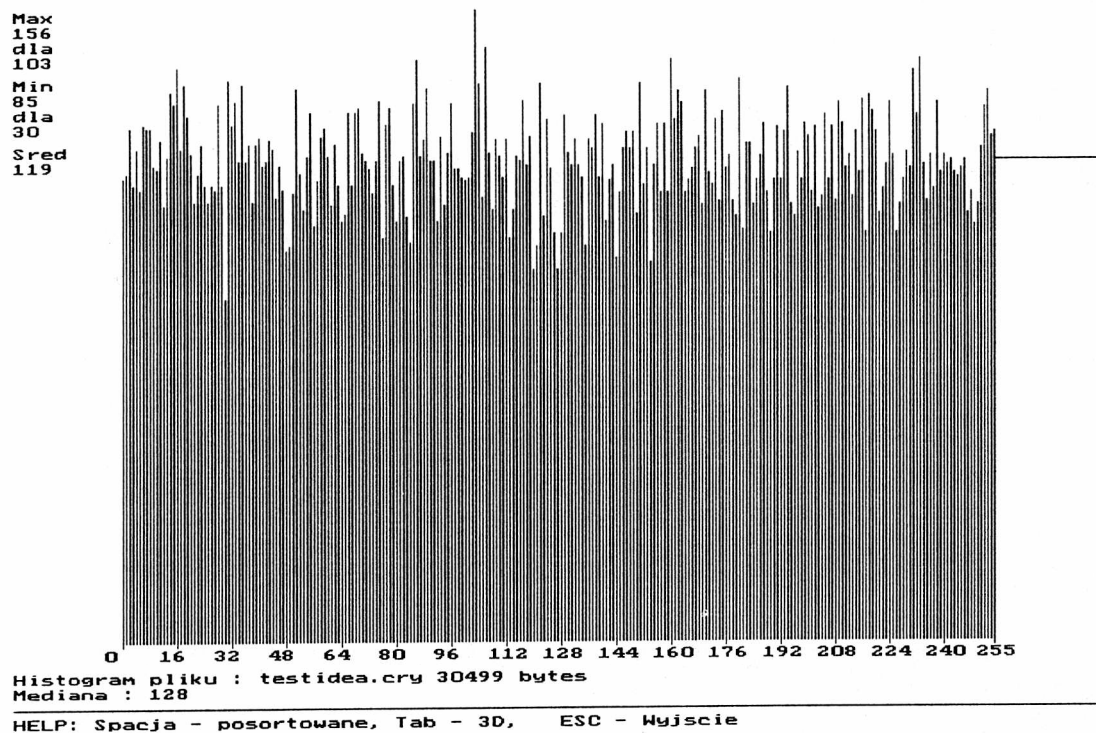


Rys. 8.9. Rozkład kodów ASCII w pliku po zaszyfrowaniu szyfrem DES.

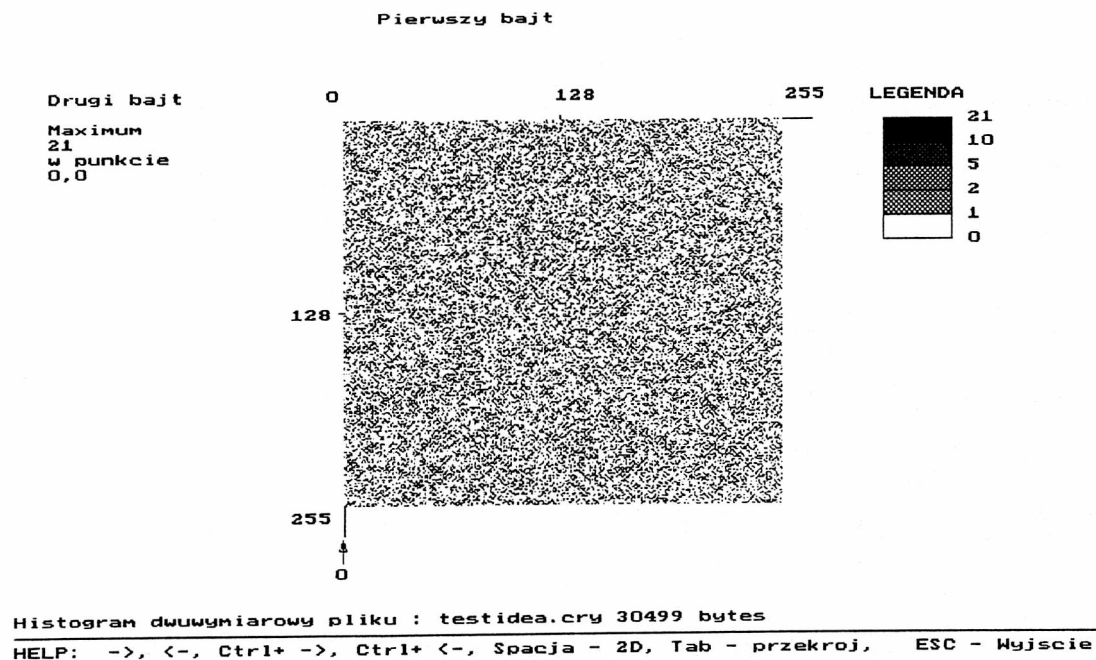


Rys. 8.10. Rozkład par kodów ASCII w pliku po zaszyfrowaniu szyfrem DES

KRYPTOGRAM - IDEA

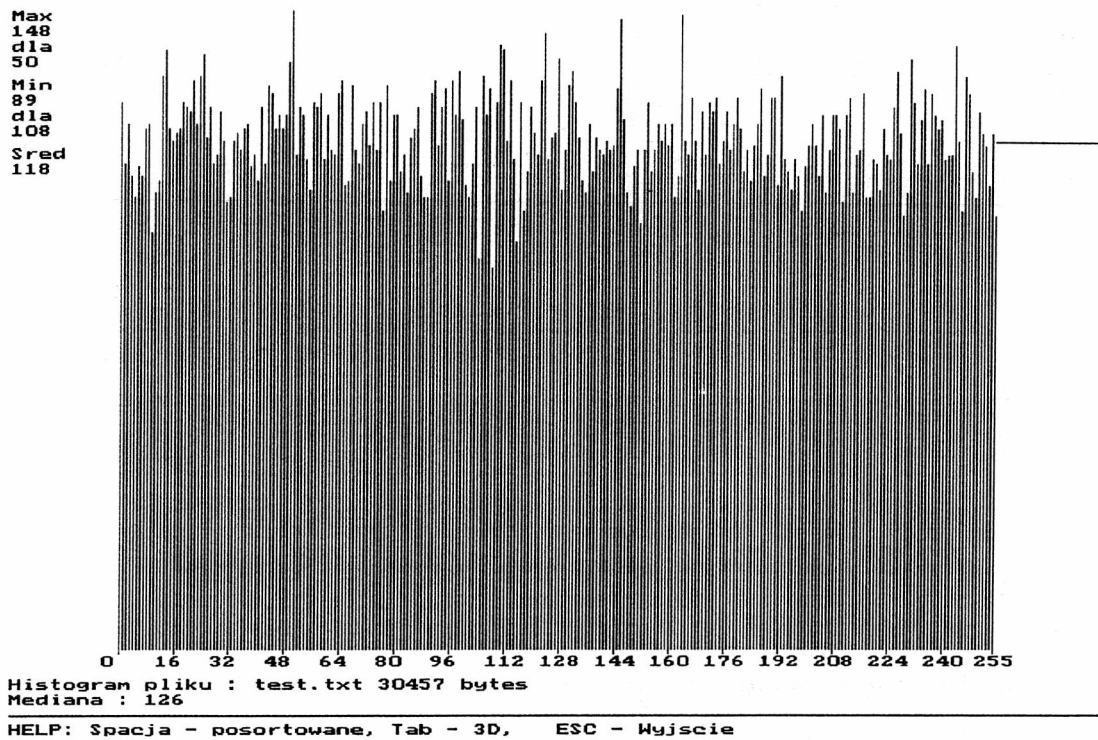


Rys. 8.11. Rozkład par kodów ASCII w pliku po zaszyfrowaniu szyfrem IDEA

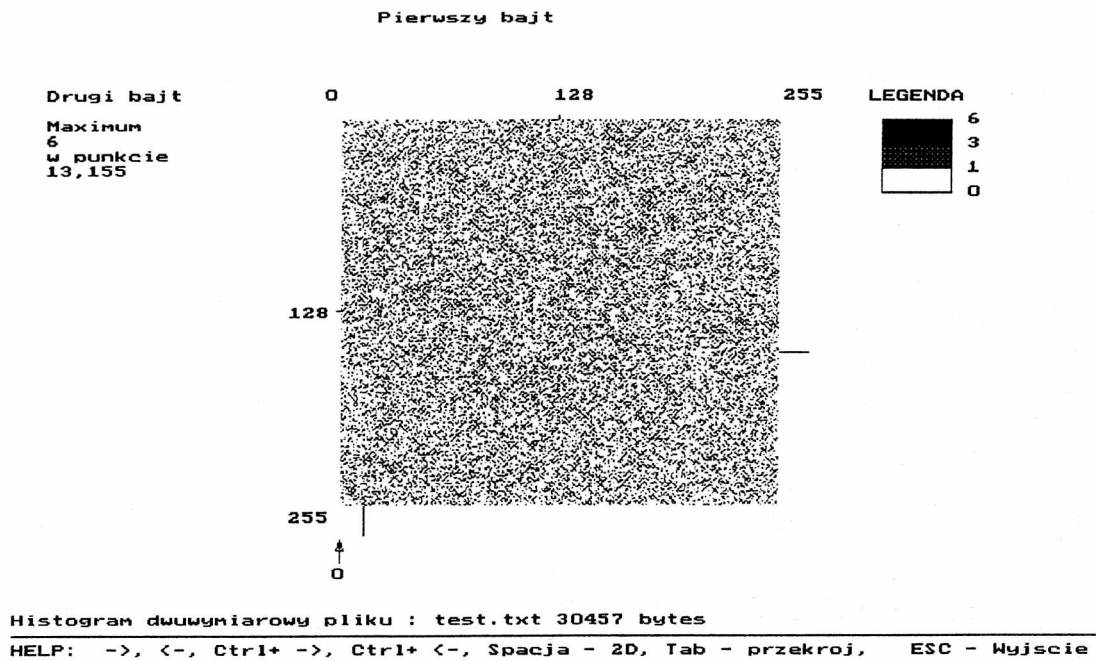


Rys. 8.12. Rozkład par kodów ASCII w pliku po zaszyfrowaniu szyfrem IDEA

KRYPTOGRAM - PSMR



Rys. 8.13. Rozkład kodów ASCII w pliku po zaszyfrowaniu szyfrem PSMR



Rys. 8.14. Rozkład par kodów ASCII w pliku po zaszyfrowaniu szyfrem PSMR

*Testy pseudolosowości***Tabela 8.2.**

	Entropia	Podatność na kompresję	χ^2	Wartość średnia	Test π	Korelacja szeregowa
TEKST JAWNY	0	100	$255 \cdot 10^6$	98	4	-
DES	7,999804	0	271,35	127,55	3,142674	-0,002286
IDEA	7,999803	0	272,64	127,47	3,141970	-0,000608
PSMR	7,999694	0	424,75	127,58	3,138796	0,000995

8.5 Ocena jakości zaprojektowanego systemu kryptograficznego

Jak widać z analizy wyników szyfrowania jakość działania kryptosystemu zbudowanego za pomocą generatorów permutacji zaprojektowanych w poprzednich rozdziałach nie odbiega poziomem działania od uznanych systemów kryptograficznych. Kryptogram uzyskany za pomocą PS MR posiada cechy statystyczne sugerujące odporność na wiele metod współczesnej kryptoanalizy. Dowodzi to słuszności tezy o przydatności przedstawionych w poprzednich rozdziałach metodach projektowania generatorów dla potrzeb kryptograficznych.

9. Zakończenie

W poprzednich rozdziałach niniejszej rozprawy zaprezentowano metody konstruowania, weryfikacji i stosowania generatorów permutacji, realizujących przekształcenia za pomocą układów do wykonywania działań arytmetycznych w ciałach Galois charakterystyki 2. Rozważania te prowadzono pod kątem zastosowań tych generatorów do celów kryptograficznych, a także ich realizacji w wersji sprzętowej w postaci układów o architekturze równoległej. Zaproponowano szereg szczegółowych i ogólnych metod analitycznych dotyczących takich zastosowań.

Omówiono metody projektowania układów generujących permutacje za pomocą przekształceń afinicznych nad $GF(2^n)$ i nad $GF(q)$ charakterystyki 2. Równania opisujące te przekształcenia mają prostą postać, można więc je łatwo zrealizować w wersji sprzętowej. Wygenerowana permutacja rozprasza bardzo dobrze przekształcaną informację po całym zakresie wartości określonych przez wielkość przetwarzanego przez generator bloku. Opisujące ją zależności są proste i stosunkowo łatwo je można zrealizować w postaci sprzętowej za pomocą podstawowych układów techniki cyfrowej XOR. Wadą przekształcenia afinicznego jest jego liniowy charakter, który w dużym zakresie czyni go podatnym na kryptoanalizę. Dodatkowo histogramy prezentujące te permutacje są dość charakterystyczne, co ułatwia ich identyfikację za pomocą analizy wizualnej.

Szczegółowej analizie poddano także metody projektowania generatorów permutacji samoodwracalnych wykorzystujących inwersję w $GF(2^n)$ i $GF(q^c)$, $q = 2^k$, $n = kc$. Uzyskane wyniki pozwalają stwierdzić, że równania, opisujące generatory permutacji samoodwracalnych są bardziej skomplikowane od generatorów opartych na przekształceniu afinicznym dzięki czemu są one bardziej odporne na kryptoanalizę, pomimo małych zdolności rozpraszania. Jednocześnie utrudnia to realizację sprzętową za pomocą bramek AND i XOR. W celu praktycznego zastosowania tych generatorów celowym byłoby uproszczenie opisujących je równań, co jednak jest zagadnieniem dość trudnym i w znacznej mierze wykracza poza zakres niniejszej pracy.

Omówiono także podstawowe cechy i sposób konstruowania generatorów permutacji wzajemnie odwracalnych i samoodwracalnych, realizowanych za pomocą podnoszenia do kwadratu i pierwiastkowania kwadratowego w $GF(2^n)$. Uzyskane wyniki pozwalają ocenić

ten typ przekształceń jako użyteczny do sprzętowych zastosowań kryptograficznych, jednak liniowy charakter oraz nierównomierność rozpraszania informacji sugeruje, że powinny one pełnić w kryptosystemach rolę uzupełniającą, zwłaszcza, że za ich pomocą można wytwarzać zarówno permutacje wzajemnie odwracalne jak i samoodwracalne.

Dokonano wnikliwej analizy aspektów dotyczących sposobów łączenia poszczególnych generatorów w celu generowania wzajemnie odwracalnych lub samoodwracalnych permutacji hybrydowych. Zaproponowano oryginalną metodę opisu tego typu struktur. Przebadano za pomocą programowej symulacji generatorów różne warianty połączeń bloków generujących permutacje. Ustalono, że przekształcanie bloków informacji o dużej ilości bitów musi być realizowane w taki sposób, aby każdy bit był przetwarzany przez kilka układów. Wynika to z faktu, że permutacja wygenerowana przy pomocy przekształceń działających na rozłącznych zbiorach bitów przetwarzanego bloku zawiera zbyt wiele cech umożliwiających kryptoanalitykowi jednoznaczną identyfikację zastosowanych przekształceń. Doświadczenia wykazały także, że cechy permutacji hybrydowej zależą od składu i kolejności występowania przekształceń użytych w generatorze. Na tej podstawie można ogólnie wnioskować o charakterze przekształcenia hybrydowego. W rozdziale dotyczącym permutacji hybrydowych zaprezentowano także zasady łączenia przekształceń w celu uzyskania hybrydowej permutacji samoodwracalnej.

Przedstawiono zasady weryfikacji zaprojektowanych generatorów dla zastosowań kryptograficznych. Stwierdzono, że najlepsze z punktu widzenia zastosowań kryptograficznych są generatory hybrydowe. Spowodowane jest to tym, że pojedynczo żadne z opisywanych wcześniej przekształceń nie posiada pełnego zestawu cech nadających generatorowi odpowiednią odporność kryptoanalityczną. Z wyników badań przeprowadzonych za pomocą symulacji komputerowych wynika także, że aby skonstruować odporny na kryptoanalizę generator trzeba zastosować w nim przekształcenia o charakterze nieliniowym. Dowodzą tego zbliżone do optymalnych wskaźniki lawinowości, kompletności, niezależności bitowej oraz odporności na korelację generatorów realizujących inwersję.

Ostatecznie w celu ustalenia przydatności prezentowanych przekształceń w kryptografii zaprojektowano kryptosystem, który może być w całości zrealizowany w wersji sprzętowej. Sprawdzono też za pomocą symulacji komputerowej jego własności szyfrujące. Symulacja ta udowodniła, że posługując się metodami projektowania oraz kryteriami weryfikacji generatorów permutacji zaproponowanymi w niniejszej pracy, można skonstruować własny

kryptosystem o równie dobrych właściwościach kryptograficznych jak znane systemy szyfrujące wykorzystujące algorytmy DES, IDEA i podobne do nich metody konstruowania szyfrów blokowych.

9.1 Przebieg realizacji pracy

Działania związane z obliczeniami wykonywanymi dla celów projektowych były wykonywane z zastosowaniem pakietu „Mathematica v.2.0”. Najbardziej czasochłonne obliczenia wykonywane w celu uzyskania wzorów dla inwersji w $GF(2^8)$ wykonywane przez komputer klasy IBM PC z pamięcią operacyjną 16 megabajtów oraz zegarem 120MHz zajmowały ok.2 godziny. Jednocześnie zastosowany pakiet pozwalał na napisanie uniwersalnego programu (jego treść znajduje się w dodatkach do niniejszej pracy) obliczającego wzory, opisujące inwersję, operację podnoszenia do kwadratu i mnożenia w ciałach $GF(2^m)$ o dowolnej wartości m .

Badania symulacyjne generatorów przeprowadzone dla celów niniejszej pracy zostały wykonane za pomocą sprzętu komputerowego klasy IBM PC. Wszystkie programy symulacyjne zostały napisane w języku C++ z zastosowaniem standardowych bibliotek. W celach porównawczych przebadano cały szereg generatorów permutacji, także nie wykorzystujących operacji arytmetycznych w ciałach Galois.

Ważnym źródłem informacji weryfikującej prowadzone badania była sieć Internet. Za jej pomocą wymieniano poglądy prowadząc dyskusje w ramach ogólnosieciowych grup dyskusyjnych sci.crypt oraz sci.crypt.research. Za pomocą Internetu pozyskiwano także szereg narzędzi do analizy i wspomagania projektowania generatorów permutacji. Informacji na temat najnowszych osiągnięć w dziedzinie zastosowań ciał Galois do generowania permutacji było w sieci zaledwie kilka [Hir95][Mor97], które dotyczyły przede wszystkim analizy s-skrzynek.

W trakcie realizacji pracy autor zdobywał dodatkowe doświadczenie dotyczące zastosowań generatorów permutacji biorąc aktywny udział w pracach zespołu zajmującego się opracowaniem i wdrożeniem nowego systemu kryptograficznego ZT-UNITAKOD autorstwa prof. dr hab. inż. Zygmunta Topolewskiego. Wynikiem pracy zespołu zostało zgłoszenie patentowe nr 08/775,253 wydane przez Urząd Patentowy USA w 1996r [Top95:1].

Szczególnie cenne w ramach prac nad zastosowaniami tej metody były dyskusje i wymiany poglądów z tak znanymi kryptografami jak David Wagner, Paul Kocher czy David Rackoff. To ich uwagi oraz dyskusje i działania realizowane w zespole wdrożeniowym ZT-UNIAKOD potwierdziły trafność doboru tematyki rozprawy, przyczyniły się do odpowiedniego ukierunkowania badań realizowanych w ramach niniejszej pracy oraz praktycznego zweryfikowania zaprojektowanych w trakcie realizacji pracy generatorów permutacji.

9.2 Perspektywy rozwoju tematyki

W trakcie realizacji pracy natrafiono na szereg zagadnień, które nie zostały do końca rozwiązane lub nie znalazły swojego odbicia w treści niniejszej rozprawy mimo że wiążą się z tematyką i mogą być rozwijane w przyszłych badaniach. Oto te problemy :

- problematyka zautomatyzowania sprzętowego lub mikroprogramowego procesu generowania macierzy nieosobliwej do generatorów wykorzystujących przekształcenia afiniczne nad $GF(2)$,
- minimalizacja wyrażeń logicznych opisujących zastosowane przekształcenia,
- badanie zależności długości podcykli występujących w permutacji od typu użytych do jej wygenerowania przekształceń,
- automatyczne generowanie zależności opisujących przekształcenia w ciałach Galois charakterystyki 2 w trakcie procesu szyfrowania lub w jego fazie przygotowawczej,
- łączenie generatorów wykorzystujących działania w ciałach Galois charakterystyki 2 z generatorami wykorzystującymi własności innych struktur algebraicznych i ich sprzętowa realizacja,
- zastosowanie dynamicznych struktur generujących permutacje z wykorzystaniem układów do rachowania w ciałach Galois charakterystyki 2 o architekturze równoległej do generowania strumienia klucza o zadanej złożoności i konkretnych parametrach pracy.

9.3 Podsumowanie i wnioski końcowe

Burzliwy rozwój technik techniki cyfrowej w zastosowaniach związanych niemal każdym typem sprzętu doprowadził do sytuacji, w której ochrona dostępu informacji lub identyfikacja użytkownika jest potrzebna nieomal wszędzie. Wykorzystując powszechnie uznane i dostępne, a jednak nie pozbawione słabości, systemy kryptograficzne można narazić się na sytuację, w której szyfrowanie ukryje bezpiecznie informację przed przeciętnym użytkownikiem Internetu, podczas gdy nawet początkujący hacker komputerowy zna ogólnie dostępne programy do łamania. Warto też wiedzieć, że całe grupy kryptologów - hobbystów wykorzystują swój sprzęt do równoległego przeszukiwania przestrzeni klucza takich systemów jak DES czy RSA. Wydaje się, że w tej sytuacji rozwiązaniem jest używanie kryptosystemów o zindywidualizowanych cechach, będących za każdym razem zagadką dla kryptoanalityków, której często nie opłacałoby się im się rozwiązywać. Do budowy takich systemów można zastosować materiały zawarte w niniejszej pracy.

Podsumowując należy stwierdzić, że założony cel pracy związany z określeniem metod projektowania, weryfikacji i zastosowania generatorów permutacji wykorzystujących układy do rachowania w ciałach Galois charakterystyki 2 został osiągnięty.

Literatura

- [Ada97] Carlise M. Adams, *Constructing Symmetric Ciphers Using the CAST Design Procedure*, Entrust Technologies, Ottawa, Canada, 1997.
- [Arn96] Arnush Craig, *Turbo C++ 4.5 w 21 dni*, Sztuka Programowania, SAMS PUBLISHING, Warszawa, 1996.
- [Bir83] G.Birkhoff, T.C.Bartee, *Współczesna algebra stosowana*, Matematyka dla Politechnik, PWN, Warszawa, 1983.
- [Bra88] G.Brassard, *Modern Cryptology*, Lecture Notes in Computer Science, Springer-Verlag, 1988.
- [Bro93] Lawrence Brown, *Analysis of the DES and Its Implications for the Design of an Extended DES[1]*, Technical Report CS88/32, Australian Defense Academy, Canberra, Australia, 1993.
- [Can97] Anne Canteaut, *Differential Cryptanalysis of Feistel Ciphers and Differentially δ - uniform Mappings*, SAC Workshop 97, Ottawa, 1997.
- [Che93] Chauchin Su, Jyrghong Wang, *ECCSyn - A Synthesis Tool for ECC Circuits*, Departament of Electrical Engineering, National Central University, Taiwan, 1993.
- [Den83] D.Elizabeth Robling Denning, *Kryptografia i ochrona danych*, Biblioteka Inżynierii Oprogramowania, Wydawnictwa Naukowo-Techniczne, Warszawa, 1993.
- [Dou95] John Douglas, *Part Two : Developing „Backdoor” Function for Product Ciphers*, Internet - CryptoMaverick HomePage, 1995.
- [Fah93] Paul Fahn, *Answers to Frequently Asked Questions About Today's Cryptography*, RSA Laboratory, Redwood City, 1993.
- [Gar70] G.A. Garakow, *Tablice nieprivodimych polinomow nad poliem $GF(p)$* , Matematyczieskije woprosy kibiernietiki i wyczislitielnoj techinki, Erewań, 1970.
- [Gre74] D.H. Green and I.S. Taylor, *Irreducible polynomials over composite Galois fields and their applications in coding techniques*, -Proc. IEEE, Vol.121, No.9, pp.935-939, 1974.

-
- [Gro71] Edward J. Groth, *Generation of Binary Sequences With Controllable Complexity*, IEEE Transactions On Information Theory vol.3, 1971.
- [Hey96] Howard M. Heys, *Modelling Avalanche in DES-Like Ciphers*, Third Annual Workshop on Selected Areas in Cryptography SAC'96, Kingston, Ontario, Canada, 1996.
- [Hir94] Shouichi Hirose, Katsuo Ikeda, *A note on the propagation characteristic and the strict avalanche criteria*, SCIS94, Biwako, Japan, 1994
- [Hir95] Shouichi Hirose, Katsuo Ikeda, *Complexity of Boolean functions satisfying the propagation criterion*, SCIS95, Inuyama, Japan, 1994
- [Jan94] Włodzimierz Janiak, *Wstęp do „Mathematica”*, Wydawnictwo PLJ, Warszawa, 1994.
- [Kan97] Yasuyoshi Kaneko, Fumihiko Sano, Kouichi Sakurai, *On Provable Security Against Differential and Linear Cryptanalysis in Generalized Feistel Ciphers with Multiple Random Functions*, SAC Workshop 97, Ottawa, 1997.
- [Kel97] Liam Keller, *Substitution-Permutation Network Cryptosystems Using Key-Dependent S-Boxes*, Queen's University, Kingston, Ontario, Canada, 1997.
- [Kli93] M.Ch.Klin, R.Pöschel, K.Rosenbaum, *Algebra stosowana dla matematyków i informatyków*, Biblioteka Inżynierii Oprogramowania, Wydawnictwa Naukowo-Techniczne, Warszawa, 1993.
- [Koś79] Czesław Kościelny, *Compact Tables of Irreducible Polynomials Over Composite Galois Fields of Characteristic 2*, Raport serii Preprinty nr 167, Wydawnictwo Politechniki Wrocławskiej, Wrocław, 1979.
- [Koś94:1] Czesław Kościelny, *Metoda generowania permutacji zbioru p^m -elementowego dla zastosowań kryptograficznych*, Instytut Cybernetyki Technicznej Politechniki Wrocławskiej, Wrocław, 1994.
- [Koś95] Czesław Kościelny, *A method of Parallel Computing in $GF(2^{6k})$* , Technical University of Zielona Góra, Zielona Góra, 1995.
- [Koś97] Czesław Kościelny, *Computing in $GF(2^m)$ using GAP*, Appl. Math. and Comp. Sci., 1997, Vol.7, No.3, 677-688., 1997.
- [Koś97:1] Czesław Kościelny, *A Design Method of Parallel Finite Field Arithmetic Circuits for Computing in $GF(q^2)$, $q = 2^{2k+1}$, $k \geq 1$* , Proceedings of the 4th

- International Workshop MIXED OF INTEGRATED CIRCUITS AND SYSTEMS MIXDES'97, Poznań 12-14 June 1997, p.323-328.
- [Koś98:1] Czesław Kościelny, *Computing in Composite $GF(q^m)$ of Characteristic 2 Formed by means of Irreducible Binomial*, Appl. Math. And Comp. Sci., Vol.8, No.3, 1998.
- [Koś98:2] Czesław Kościelny, Tomasz Hebisz, *Metoda konstrukcji szyfru blokowego wykrywającego manipulacje na kryptogramach*, Enigma'98 - II Krajowa Konferencja Zastosowań Kryptografii, KII-WOC-29-41, Warszawa 1998.
- [Kob95] Neal Koblitz, *Wykład z teorii liczb i kryptografii*, Wydawnictwa Naukowo-Techniczne, Warszawa, 1995.
- [Lie86] Rudolf Liedl, Harald Niederreiter, *Introduction to finite fields and their applications*, Cambridge University Press, Cambridge 1986.
- [Mar77] Władysław Marcinkowski, *Właściwości cyklicznego kodu Reeda-Solomona przy uproszczonej metodzie dekodowania. Praca doktorska*, Instytut Cybernetyki Technicznej Politechniki Wrocławskiej, Wrocław, 1977.
- [Mas93] James L.Massey, Xuejia Lai, *Device for The Conversion of A Digital Block And Use of Same*, United States Patent no.5,214,703,
- [Mat91] Edoardo D.Mastrovito, *VLSI Architectures for Computations in Galois Fields*, Departament of Electrical Engineering Linköping University, Linköping, Sweden, 1991.
- [Mau92] Ueli M. Maurer, *A Universal Statistical Test for Random Bit Generators*, Journal in Cryptology, vol. 5, no. 2, pp.89-105.
- [Mce87] Robert J.McEliece, *Finite Fields for Computer Scientists and Engineers*, Kluwer Academic Publishers, 1987.
- [Men93] Alfred J.Menezes (editor), *Applications of Finite Fields*, Kluwer Academic Publishers, 1993.
- [Mis96] S. Mister, C. Adams, *Practical S-box Design*, Third Annual Workshop on Selected Areas in Cryptography SAC'96, Kingston, Ontario, Canada, 1996.
- [Moc97] Władysław Mochnacki, *Kody korekcyjne i kryptografia*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 1997.
- [Mor97] Shiho Moriai, Takeshi Shimoyama, *Performance and Security of Block Ciphers Using Operations in $GF(2^n)$* , SAC Workshop 97, Ottawa, 1997.

-
- [Mrs84] George Marsaglia, *A Current View of Random Number Generators*, 16th Symposium on the Interface, Atlanta, 1984.
- [Mrs93] George Marsaglia, *Monkey Tests for Random Number Generators*, Computer & Mathematics with Applications, 9, 1993.
- [Ole96] Daniel Olejar, Martin Stanek, *Design of Substitution Blocks Satisfying Strict Avalanche Criterion*, SAC Workshop 97, Ottawa, 1997.
- [Paa96] Christof Paar, *A New Architecture for a Parallele Finite Field Multiplier with Low Complexity Based on Compostite Fields*, IEEE Transtactions on Computers, vol. 45, no. 7, pp 856-861, July 1996.
- [Pet72] W.W. Peterson and E.J. Weldon, *Error-Correcting Codes*, Cambridge, MA:MIT Press, 1972.
- [Rit98] Terry Ritter, *Measured Distant Avalanche in Large Block Ciphers*, Ritter Software Engineering, [WWWTR], 1998.
- [Rit98:1] Terry Ritter, *S-Box Design and Related Topics*, Ritter Software Engineering, [WWWTR], 1998.
- [Rue86] Reiner A.Rueppel, *Analysis and Design of Stream Ciphers*, Communications and Control Engineering Series, Springer-Verlag, 1986.
- [Sch95] Bruce Schneier, *Kryptografia dla praktyków*, Wydawnictwa Naukowo-Techniczne, Warszawa, 1995.
- [Ser96] Serge Mister, Carlise Adams, *Practical S-box Design*, SAC Workshop 96, Ottawa, 1996.
- [Sim92] G.J. Simmons (editor), *Contemporary Cryptology. The Science of Information Integrity*, IEEE Inc., New York, 1992.
- [Sto94] Janusz Stokłosa, *Kryptograficzna ochrona danych w systemach komputerowych*, Biblioteka Użytkownika Mikrokomputerów, Nakom, Poznań, 1994.
- [Top94] Z. Topolewski, L. Wolaniuk, R. Malewicz, *Komputerowe zabezpieczenie informacji - referat*, Międzynarodowa Konferencja pod auspicjami Rady Europy na temat „Teleinformatyka w wojsku i policji”, Poznań 20-24.04.94r,
- [Top95] Zygmunt Topolewski, *Analiza i synteza szyfrowania informacji*, Wyższa Szkoła Oficerska im.T.Kościuszki, Wrocław, 1995.
- [Top95:1] Z. Topolewski, R. Malewicz, L. Wolaniuk, *System & Method ZT-UNITAKOD*, Patent Application no.08/775,253, Toronto, Canada, 1995.

- [Wil77] F. Jessie MacWilliams, Neil J.A. Sloane, *The Theory of Error-Correcting Codes*, Amsterdam, North Holland, 1977.
- [Wol94] Leszek Wolaniuk, Robert Malewicz, *Szyfr idealny?*, Biuletyn Inżynierii Wojskowej nr 1, Wrocław, 1994.
- [Wol94] Leszek Wolaniuk, Robert Malewicz, *System kryptograficzny TMW-ULTRA-PERFECT*, Biuletyn Inżynierii Wojskowej nr 2, Wrocław, 1994.
- [Wro94] Piotr Wróblewski, *Język C++ dla programistów*, HELION Poland, 1994.

Źródła elektroniczne

Serwis WWW

- [WWWGA] www.math.rwth-aachen.de/mschoene/GAP-Manual/
- [WWWTR] www.io.com/~ritter/

DODATEK A

LICZBA ELEMENTÓW PEŁNEJ GRUPY LINIOWEJ $GL(m,2)$ I LICZBA PRZEKSZTAŁCEŃ
AFINICZNYCH m - WYMIAROWEJ PRZESTRZENI LINIOWEJ NAD $GF(2)$ NA SIEBIE
W FUNKCJI m

m	$ GL(m,2) $	N_T	m	$ GL(m,2) $	N_T
2	6	24	34	$2,8265 \cdot 10^{347}$	$4,8559 \cdot 10^{357}$
3	168	1344	35	$1,6685 \cdot 10^{368}$	$5,7329 \cdot 10^{378}$
4	20160	322560	36	$3,9396 \cdot 10^{389}$	$2,7073 \cdot 10^{400}$
5	$9,9994 \cdot 10^6$	$3,1998 \cdot 10^8$	37	$3,7209 \cdot 10^{411}$	$5,1139 \cdot 10^{422}$
6	$2,0159 \cdot 10^{10}$	$1,2902 \cdot 10^{12}$	38	$1,4057 \cdot 10^{434}$	$3,8640 \cdot 10^{445}$
7	$1,6385 \cdot 10^{14}$	$2,0973 \cdot 10^{16}$	39	$2,1242 \cdot 10^{457}$	$1,1678 \cdot 10^{469}$
8	$5,3481 \cdot 10^{18}$	$1,3691 \cdot 10^{21}$	40	$1,2840 \cdot 10^{481}$	$1,4118 \cdot 10^{493}$
9	$6,9961 \cdot 10^{23}$	$3,5820 \cdot 10^{26}$	41	$3,1046 \cdot 10^{505}$	$6,8270 \cdot 10^{517}$
10	$3,6644 \cdot 10^{29}$	$3,7523 \cdot 10^{32}$	42	$3,0026 \cdot 10^{530}$	$1,3205 \cdot 10^{543}$
11	$7,6811 \cdot 10^{35}$	$1,5731 \cdot 10^{39}$	43	$1,1616 \cdot 10^{556}$	$1,0217 \cdot 10^{569}$
12	$6,4418 \cdot 10^{42}$	$2,6385 \cdot 10^{46}$	44	$1,7974 \cdot 10^{582}$	$3,1621 \cdot 10^{595}$
13	$2,1612 \cdot 10^{50}$	$1,7705 \cdot 10^{54}$	45	$1,1125 \cdot 10^{609}$	$3,9144 \cdot 10^{622}$
14	$2,9006 \cdot 10^{58}$	$4,7523 \cdot 10^{62}$	46	$2,7545 \cdot 10^{636}$	$1,9383 \cdot 10^{650}$
15	$1,5572 \cdot 10^{67}$	$5,1026 \cdot 10^{71}$	47	$2,7280 \cdot 10^{664}$	$3,8393 \cdot 10^{678}$
16	$3,3440 \cdot 10^{76}$	$2,1915 \cdot 10^{81}$	48	$1,0807 \cdot 10^{693}$	$3,0418 \cdot 10^{707}$
17	$2,8724 \cdot 10^{86}$	$3,7650 \cdot 10^{91}$	49	$1,7124 \cdot 10^{722}$	$9,6398 \cdot 10^{736}$
18	$9,8696 \cdot 10^{96}$	$2,5873 \cdot 10^{102}$	50	$1,0853 \cdot 10^{752}$	$1,2220 \cdot 10^{767}$
19	$1,3565 \cdot 10^{108}$	$7,1118 \cdot 10^{113}$	51	$2,7517 \cdot 10^{782}$	$6,1962 \cdot 10^{797}$
20	$7,4572 \cdot 10^{119}$	$7,8195 \cdot 10^{125}$	52	$2,7905 \cdot 10^{813}$	$1,2567 \cdot 10^{829}$
21	$1,6399 \cdot 10^{132}$	$3,4390 \cdot 10^{138}$	53	$1,1320 \cdot 10^{845}$	$1,0196 \cdot 10^{861}$
22	$1,4424 \cdot 10^{145}$	$6,0500 \cdot 10^{151}$	54	$1,8367 \cdot 10^{877}$	$3,3088 \cdot 10^{893}$
23	$5,0751 \cdot 10^{158}$	$4,2573 \cdot 10^{165}$	55	$1,1921 \cdot 10^{910}$	$4,2950 \cdot 10^{926}$
24	$7,1426 \cdot 10^{172}$	$1,1983 \cdot 10^{180}$	56	$3,0949 \cdot 10^{943}$	$2,2301 \cdot 10^{960}$
25	$4,0209 \cdot 10^{187}$	$1,3492 \cdot 10^{195}$	57	$3,2139 \cdot 10^{977}$	$4,6317 \cdot 10^{994}$
26	$9,0543 \cdot 10^{202}$	$6,0763 \cdot 10^{210}$	58	$1,3350 \cdot 10^{1012}$	$3,8479 \cdot 10^{1029}$
27	$8,1554 \cdot 10^{218}$	$1,0946 \cdot 10^{227}$	59	$2,2182 \cdot 10^{1047}$	$1,2787 \cdot 10^{1065}$
28	$2,9383 \cdot 10^{235}$	$7,8874 \cdot 10^{243}$	60	$1,4742 \cdot 10^{1083}$	$1,6996 \cdot 10^{1101}$
29	$4,2345 \cdot 10^{252}$	$2,2734 \cdot 10^{261}$	61	$3,9192 \cdot 10^{1119}$	$9,0370 \cdot 10^{1137}$
30	$2,4410 \cdot 10^{270}$	$2,6210 \cdot 10^{279}$	62	$4,1676 \cdot 10^{1156}$	$1,9219 \cdot 10^{1175}$
31	$5,6287 \cdot 10^{288}$	$1,2087 \cdot 10^{298}$	63	$1,7727 \cdot 10^{1194}$	$1,6350 \cdot 10^{1213}$
32	$5,1915 \cdot 10^{307}$	$2,2297 \cdot 10^{317}$	64	$3,0161 \cdot 10^{1232}$	$5,5637 \cdot 10^{1251}$
33	$1,9153 \cdot 10^{327}$	$1,6453 \cdot 10^{337}$	65	$2,0526 \cdot 10^{1271}$	$7,5729 \cdot 10^{1290}$

DODATEK B

PROGRAM DO OBLICZANIA WZORÓW NA KWADRAT, INWERSJĘ I ILOCZYN DWÓCH ELEMENTÓW DOWOLNEGO $GF(2^n)$ O WIELOMIANIE NIEREDUKOWALNYM $P(x) = 1 + \dots + x^n$ ZA POMOCĄ PAKIETU MATHEMATICA

(* Moduł do obliczania wzorów kwadrat i odwrotność elementu oraz iloczyn dwóch *)
 (* elementów ciała 2^n o wielomianie nieredukowalnym irp. Parametrami *)
 (* wejściowymi są n - najwyższa potęga 'x' w wielomianie nieredukowalnym, irp- *)
 (* wielomian nieredukowalny w postaci $1 + \dots + x^n$. Wyniki obliczeń składowane są *)
 (* na dysku w bieżącym katalogu pod nazwą 'GF' *)

n=4;

irp= x^n+x+1 ;

pa=Sum[a[i]* x^i ,{i,0,n-1}];

pb=Sum[b[i]* x^i ,{i,0,n-1}];

pd=PolynomialRemainder[pa*pb,irp,x];

pd=PolynomialMod[pd,2];

clpd=CoefficientList[pd,x];

prd=Table[0,{n}];

Do[prd[[i]]=clpd[[i]],{i,1,n}];

(* Koniec generowania wzorów na iloczyn dwóch elementów. Zależności *)
 (* znajdują się w tablicy 'prd' *)

m=Table[0,{n}];

ut=IdentityMatrix[n];

Do[m[[i+1]]=Coefficient[clpd,b[i]],{i,0,n-1}];

MatrixForm[m];

d=PolynomialMod[Det[m],2];

Do[Do[d=d/.a[i]^j->a[i],{i,0,n-1}],{j,2,n}];

in=Table[0,{n}];

```
Do[mt=m;mt[[k]]=ut[[1]];in[[k]]=PolynomialMod[Det[mt],2];Do[Do[in[[k]]=in[[k]]/.a[i]^j-
>a[i],{i,0,n-1}],{j,2,n}];in[[k]]=PolynomialMod[in[[k]],2],{k,1,n}];
```

```
(* Koniec generowania wzorów na inwersję elementu. Zależności znajdują się *)
(* w tablicy 'in' *)
```

```
tw=PolynomialRemainder[pa^2,irp,x];
```

```
tw=PolynomialMod[tw,2];
```

```
cltw=CoefficientList[tw,x];
```

```
Do[Do[Do[cltw[[k]]=cltw[[k]]/.a[i]^j->a[i],{i,0,n-1}],{j,2,n}],{k,1,n}];
```

```
(*Koniec generowania wzorów na kwadrat elementu. Zależności znajdują się *)
(* w tablicy 'cltw' *)
```

```
wyd=Table["",{4}];
```

```
wyd[[1]]=" Wzory wyliczeniowe dla ciała GF(2^n);
```

```
wyd[[2]]=n;
```

```
wyd[[3]]=" o wielomianie nieredukowalnym P(X)=";
```

```
wyd[[4]]=irp;
```

```
wyd>>gf;
```

```
nap=" To są wzory na kwadrat :">>>gf;
```

```
Do[wyd[[1]]="sqr";wyd[[2]]=i-1;wyd[[3]]=" ";wyd[[4]]=cltw[[i]];wyd>>>gf,{i,1,n}];
```

```
" To są wzory na inwersje :">>>gf;
```

```
Do[wyd[[1]]="inv";wyd[[2]]=i-1;wyd[[3]]=" ";wyd[[4]]=in[[i]];wyd>>>gf,{i,1,n}];
```

```
" To są wzory na mnożenie dwóch elementów :">>>gf;
```

```
Do[wyd[[1]]="prd";wyd[[2]]=i-1;wyd[[3]]=" ";wyd[[4]]=prd[[i]];wyd>>>gf,{i,1,n}];
```

```
(* Koniec wypisywania wzorów i składowania ich na dysku. Teraz nastąpi *)
```

```
(* obliczenie wartości kwadratu i inwersji dla każdego elementu danego GF(2^n)*)
```

```
!!gf
```

```
invw=in;
```

```
licz=Table[0,{n}];
```

```
xp=IntegerDigits[11,2]
```

```
l=Length[xp];
```

```
Do[a[l-j]=xp[[j]],{j,1,l}];
```

```
Do[licz[[n-h+1]]=Mod[in[[h]],2],{h,1,n}];
```

```
licz
```

```
(*koniec wypisywania wzorów do obliczania inwersji w tym ciele*)
```

DODATEK C

NIEKTÓRE PLIKI ŹRÓDŁOWE PROCEDUR ZASTOSOWANYCH W OPROGRAMOWANIU TESTUJĄCYM HYBRYDOWE GENERATORY PERMUTACJI

Podstawą systemu są procedury w języku programowania C++ [Arn96][Wro94] symulujące działanie poszczególnych bloków permutacyjnych. Aby zapewnić maksymalną czytelność w odniesieniu do matematycznego zapisu występującego wzorów w programie zastosowano notację pozwalającą na szybką konwersję zapisu matematycznego jedynie poprzez zmianę symboli operacji i indeksów np. zapis $y_0 = x_0x_1 + x_3 + x_0x_1x_3$ przekształcony do postaci użytej w programie ma postać $y[0] = x[0]\&x[1] \wedge x[3] \wedge x[0]\&x[1]\&x[3]$. Jak z tego widać znaki operacji mnożenia i dodawania w $GF(2)$ zostały bezpośrednio zastąpione symbolami „&” (AND) i „^” (XOR). Dodatkowo w systemie testującym umieszczono procedurę wypisującą permutacje w postaci cykli (wraz ze zliczaniem długości występujących podcykli). Zintegrowano także z programem podprogram graficzny prezentujący permutację w postaci histogramu. Całość systemu testującego została zrealizowana jako aplikacja o nazwie „hybrydy.exe” działająca w systemie DOS z zastosowaniem biblioteki graficznej BGI. Wprowadzanie danych do systemu jest dwuetapowe. Najpierw, jako parametry uruchomieniowe programu, podaje się symbole kolejnych składowych bloków permutacyjnych¹ a następnie określa zakres ich przetwarzania w 8 bitowym bloku danych (np. dla 6 bitowego bloku inwersji 0, 1 lub 2 jako początek zakresu przetwarzanych bitów) lub wartości parametrów (wartość stałej c dla przekształcenia afinicznego). Wyniki działania programu widoczne są na ekranie i jednocześnie zapisywane do plików o nazwach „hybrydy.txt” i „hybrydsl.txt”.

```
unsigned int af8b(unsigned int xa, unsigned int ca, unsigned int typ=0).
/* Funkcja realizuje przekształcenie afiniczne w GF(2^8). 'xt' - tabl. */
/* wejściowa, 'yt' - tablica wyjściowa, 'ct' - tablica stałej, typ - typ */
/* przekształcenia (0 - proste, 1 - odwrotne) */
{
    unsigned int yaf=0;
    if(typ!=0) xa=xa^ca;
    dotab(xa,xt);
    for(unsigned char j=0;j<16;j++) yt[j]=xt[j];
    if(typ==0)
```

¹ Wykaz symboli można otrzymać po wpisaniu do systemu „?” jako parametru uruchomieniowego.


```

{
    yt[0]=xt[1]^xt[2]^xt[4]^xt[5]^xt[6]^xt[7];/^ct[0];
    yt[1]=xt[1]^xt[2]^xt[3]^xt[5]^xt[6];/^ct[1];
    yt[2]=xt[1]^xt[3]^xt[4]^xt[5];/^ct[2];
    yt[3]=xt[0]^xt[2]^xt[4]^xt[5]^xt[6];/^ct[3];
    yt[4]=xt[0]^xt[1]^xt[4];/^ct[4];
    yt[5]=xt[0]^xt[2]^xt[6]^xt[7];/^ct[5];
    yt[6]=xt[1]^xt[5];/^ct[6];
    yt[7]=xt[0]^xt[3]^xt[6]^xt[7];/^ct[7];
}
else
{
    yt[0]=xt[0]^xt[1]^xt[2]^xt[3]^xt[4]^xt[5];/^ct[0]^ct[1]^ct[2]^ct[3]^ct[4]^ct[5];
    yt[1]=xt[0]^xt[4]^xt[5]^xt[6];/^ct[0]^ct[4]^ct[5]^ct[6];
    yt[2]=xt[1]^xt[3]^xt[4]^xt[5]^xt[7];/^ct[1]^ct[3]^ct[4]^ct[5]^ct[7];
    yt[3]=xt[1]^xt[3]^xt[4];/^ct[1]^ct[3]^ct[4];
    yt[4]=xt[1]^xt[2]^xt[3]^xt[4]^xt[6];/^ct[1]^ct[2]^ct[3]^ct[4]^ct[6];
    yt[5]=xt[0]^xt[4]^xt[5];/^ct[0]^ct[4]^ct[5];
    yt[6]=xt[1]^xt[5]^xt[6]^xt[7];/^ct[1]^ct[5]^ct[6]^ct[7];
    yt[7]=xt[0]^xt[1]^xt[2]^xt[6];/^ct[0]^ct[1]^ct[2]^ct[6];
};
if(typ==0) yaf=ztab(yt)^ca;
else yaf=ztab(yt);
return(yaf);
}

```

unsigned int i4b(unsigned char ox,unsigned int o_p)

/ Funkcja oblicza odwrotność w $GF(2^4)$. */*

/ 'xt' - tabl. wejściowa, 'yt' - tablica wyjściowa, 'tabc' - tablica stałej */*

```

{
    if(o_p+3>7) o_p=0;
    dotab(ox,xt);
    for(unsigned char j=0;j<16;j++) yt[j]=xt[j];
    yt[0+o_p]=xt[o_p+0]^xt[o_p+1]^xt[o_p+2]^xt[o_p+3]^xt[o_p+0]&xt[o_p+2]^xt[o_p+1]&xt[o_p+2]^xt[o_p+0]&xt[o_p+1]&xt[o_p+2]^xt[o_p+1]&xt[o_p+2]&xt[o_p+3];
    yt[1+o_p]=xt[o_p+3]^xt[o_p+0]&xt[o_p+1]^xt[o_p+0]&xt[o_p+2]^xt[o_p+1]&xt[o_p+2]^xt[o_p+1]&xt[o_p+3]^xt[o_p+0]&xt[o_p+1]&xt[o_p+3];
    yt[2+o_p]=xt[o_p+2]^xt[o_p+3]^xt[o_p+0]&xt[o_p+1]^xt[o_p+0]&xt[o_p+2]^xt[o_p+0]&xt[o_p+3]^xt[o_p+0]&xt[o_p+2]&xt[o_p+3];
    yt[3+o_p]=xt[o_p+1]^xt[o_p+2]^xt[o_p+3]^xt[o_p+0]&xt[o_p+3]^xt[o_p+1]&xt[o_p+3]^xt[o_p+2]&xt[o_p+3]^xt[o_p+1]&xt[o_p+2]&xt[o_p+3];
    return(ztab(yt));
}

```

/ Procedura badająca stopień rozpraszania wartości przez układ poprzez */*

/ weryfikacje dla wszystkich wartości wpływu zmiany jednego bitu */*

/ na wejściu na ilość zmian w wartościach pozycji bitowych wyjścia */*

```
void rozprosz(unsigned int *TabRozp)
{
    unsigned long rsum=0;
    long zr,x,jeden;
    int il_bit, zakres=0;
    cout << endl << "Ile bitow?";
    cin >> il_bit;
    zakres=1<<il_bit;
    for(int i=0;i<zakres;i++)
    {
        jeden=1;
        // cout << endl;
        for(int k=0;k<il_bit;k++)
        {
            zr=i^jeden;
            // cout << wagi(TabRozp[i]^TabRozp[zr]) << " ";
            rsum=rsum+wagi(TabRozp[i]^TabRozp[zr]);
            jeden<<1;
        }
    }
    cout << "Wartosc SAC = ( ok." << rsum/(zakres*il_bit) << " " <<
    ((rsum*10)/(zakres*il_bit))%10 << " bita ) tzn. " << (rsum*100)/(zakres*il_bit*il_bit)
    << "% bitow " ;
    komplet(TabRozp,il_bit);
    BIC(TabRozp,il_bit);
    getch();
}
// koniec rozprosz
```

DODATEK D

WYBRANE PROCEDURY SKŁADOWE PROGRAMU SYMULUJACEGO DZIAŁANIE KRYPTOSYSTEMU PSMR

/* Procedura realizująca sieć permutacyjno-przestawieniową */

```
void PSMR(unsigned char *PSMRBufor, char PSMRTryb, unsigned char  
*PSMRKlucz)
```

```
{  
// deklaracja zmiennych unii do przetwarzania bloku
```

```
union
```

```
{  
    unsigned long *PolDanych;  
    unsigned char *BajtyDanych;  
}BlokDanych;
```

```
union
```

```
{  
    unsigned long *PolKlucza;  
    unsigned char *BajtyKlucza;  
}BlokKlucza;
```

```
union
```

```
{  
    unsigned long Polx[2];  
    unsigned char Bajtyx[8];  
}x;
```

```
// Pozyskanie danych z zewnątrz - bufor i klucz  
BlokDanych.BajtyDanych=PSMRBufor;
```

```
// początek pętli rozpoczynającej 16-krokowa sieć PSMR  
for(int i=0;i<16;i++)  
{
```

```
    // wstawienie aktualnego klucza do unii klucza  
    if(PSMRTryb=='e') for(char ikl=0;ikl<16;ikl++)  
BlokKlucza.BajtyKlucza[ikl]=PSMRKlucz[16*i+ikl];  
    if(PSMRTryb=='d') for(char ikl=0;ikl<16;ikl++)  
BlokKlucza.BajtyKlucza[ikl]=PSMRKlucz[16*(15-i)+ikl];
```

```
    // Sumowanie prawych połówek z akt. kluczem
```

```
    // przed s-skrzynkami
```

```
    x.Polx[0]=BlokDanych.PolDanych[2]^BlokKlucza.PolKlucza[0];  
    x.Polx[1]=BlokDanych.PolDanych[3]^BlokKlucza.PolKlucza[1];
```

```
// Sekcja s-skrzynek. Parametry funkcji : 1.BajtDanych, 2.BajtKlucza  
// 3.BajtParametru dla operacji XOR
```

```

    for(byte s=0;s<8;s++)

x.Bajtyx[s]=Sblok(x.Bajtyx[s],BlokKlucza.BajtyKlucza[s],BlokKlucza.BajtyKlucza[s+8])
;

    x.Polx[0]=x.Polx[0]^BlokDanych.PolDanych[0];
    x.Polx[1]=x.Polx[1]^BlokDanych.PolDanych[1];
    BlokDanych.PolDanych[0]=BlokDanych.PolDanych[2];
    BlokDanych.PolDanych[1]=BlokDanych.PolDanych[3];
    BlokDanych.PolDanych[2]=x.Polx[0];
    BlokDanych.PolDanych[3]=x.Polx[1];
    }// koniec petli 16 przekształceń
// końcowa zamiana miejscami połówek bloku
    x.Polx[0]=BlokDanych.PolDanych[0];
    x.Polx[1]=BlokDanych.PolDanych[1];
    BlokDanych.PolDanych[0]=BlokDanych.PolDanych[2];
    BlokDanych.PolDanych[1]=BlokDanych.PolDanych[3];
    BlokDanych.PolDanych[2]=x.Polx[0];
    BlokDanych.PolDanych[3]=x.Polx[1];
} // koniec PSMR

/* Funkcja s-skrzynki przetwarzająca jeden bajt danych */
unsigned char Sblok(unsigned char BajtDanych, unsigned char BajtKlucza, unsigned
char BajtParam)
{
    unsigned char G[2],przes,Wynik;
    G[0]=BajtKlucza&7;
    G[1]=(BajtKlucza>>3)&7;
    przes=BajtKlucza>>6;
    Wynik=BajtDanych;

// stały układ przekształcający. Niezależny od klucza l(0:5)>l(2:7)>Q(0:7)>
// l(2:7)>l(0:5)
    for(int i=0;i<2;i++)
        switch(G[i])
        {
            case 0:Wynik=af8b(Wynik,0,0);break;
            case 1:Wynik=a826b(Wynik,0,0);break;
            case 2:Wynik=i6b(Wynik,przes);break;
            case 3:Wynik=i4b(Wynik,przes);break;
            case 4:Wynik=i436b(Wynik,przes);break;
            case 5:Wynik=si8b(Wynik);break;
            case 6:Wynik=s8b(Wynik,1);break; //^BajtParam;break;
            case 7:Wynik=af8b(Wynik,0,1);break;
        }

Wynik=i6b(Wynik,0);Wynik=i6b(Wynik,2);Wynik=si8b(Wynik);Wynik=i6b(Wynik,2);
Wynik=i6b(Wynik,0);

```

```

for(i=1;i>=0;i--)
switch(G[i])
{
    case 0:Wynik=af8b(Wynik,0,1);break;
    case 1:Wynik=a826b(Wynik,0,1);break;
    case 2:Wynik=i6b(Wynik,przes);break;
    case 3:Wynik=i4b(Wynik,przes);break;
    case 4:Wynik=i436b(Wynik,przes);break;
    case 5:Wynik=si8b(Wynik);break;
    case 6:Wynik=s8b(Wynik,0);break; //^BajtParam;break;
    case 7:Wynik=af8b(Wynik,0,0);break;
}
return(Wynik);
} // koniec procedury s-box'u

```

/* procedura wytwarzająca podklucze dla PSMR */

```

void InicKlucz()
{
    size_t dlugosc;
    cout << endl << "Podaj klucz szyfrujący (max. 16 znaków) : ";
    cin >> Klucz;
    cout << " WERYFIKACJA : (" << strlen(Klucz) << ") znaków = ";
    dlugosc=strlen(Klucz);
    if(dlugosc<16)
        for(int IKw=dlugosc;IKw<16;IKw++)
            Klucz[IKw]=Sblok(Klucz[IKw%dlugosc],Klucz[(IKw+1)%dlugosc]&7,Klucz[(IKw+2)%dlugosc]);
    for(w=0;w<dlugosc;w++) cout << Klucz[w];
    cout << endl;

    // Wygenerowanie tablicy kluczy bieżących dla sieci PSMR
    for(int t1=0;t1<256;t1++)
        TabPerm[t1]=t1^Sblok(t1,Klucz[t1%16],t1);

    for(t1=0;t1<256;t1++)
        TabPerm[t1]=TabPerm[t1]^Sblok(Klucz[t1%16],TabPerm[t1],Klucz[t1%16]);

    for(t1=0;t1<16;t1++)
        for(int t1p=0;t1p<8;t1p++) TabPerm[t1*16+t1p]=Klucz[t1]^TabPerm[(16-
t1)*16+t1p]^TabPerm[t1*16+(t1p+8)];
} // koniec procedury inicjującej podklucze dla PSMR

```



mgr inż. Leszek Wolaniuk
Wyższa Szkoła Oficerska im. Tadeusza Kościuszki we Wrocławiu
ul. Czajkowskiego 109
Wrocław

Niniejszy raport otrzymują:

1. OINT	1 egz.
2. Biblioteka Główna PWr.	1 egz.
3. Z-ca Dyrektora Instytutu	1 egz.
4. Promotor	1 egz.
5. Recenzenci	2 egz.
6. Autor	2 egz.
Razem: 8 egz.	

Raport wpłynął do Redakcji I-6
w grudniu 1998 r.