ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY W SZCZECINIE

WYDZIAŁ INFORMATYKI

mgr inż. Michał Apolinarski[+]

Metody oceny jakości algorytmów generowania kluczy rundowych w szyfrach blokowych

Rozprawa doktorska

Promotor: dr hab. inż. Krzysztof Chmiel

Promotor pomocniczy: dr inż. Anna Grocholewska-Czuryło

Szczecin, 2022

Pragnę podziękować Promotorom rozprawy dr. hab. inż. Krzysztofowi Chmielowi oraz dr inż. Annie Grocholewskiej-Czuryło – za ogromne wsparcie, poświęcony czas, cenne uwagi oraz konstruktywną krytykę.

Szczególne podziękowania składam dr. hab. inż. Januszowi Stokłosie – mojemu pierwszemu opiekunowi naukowemu, w latach 2008 - 2012, który zainteresował mnie problematyką rozprawy i wskazał kierunek badań.

Dziękuję Koleżankom i Kolegom, z dawnego Zakładu Bezpieczeństwa Systemów Informatycznych Politechniki Poznańskiej, za przyjazną atmosferę pracy i motywowanie do ukończenia rozprawy.

Dziękuję mojej żonie Joannie oraz moim Rodzicom, Janowi i śp. Krystynie, za stworzenie mi warunków do pracy naukowej, za wyrozumiałość i cierpliwość. Niniejszą rozprawę dedykuję właśnie im oraz moim wspaniałym dzieciom Krzysiowi i Hani.

[+] Politechnika Poznańska, Wydział Informatyki i Telekomunikacji, Zakład Inżynierii Informatycznej
michal.apolinarski[at]put.poznan.pl

Spis treści

OZNACZENIA 5

1. WSTĘP 8

1.1. TEMATYKA ROZPRAWY 8

1.2. CEL I TEZA ROZPRAWY 10

1.3. STRUKTURA ROZPRAWY 11

2. PODSTAWY MECHANIZMÓW KRYPTOGRAFICZNYCH 14

2.1. CHARAKTERYSTYKA OGÓLNA 14

2.2. MECHANIZMY OCHRONY POUFNOŚCI 14

2.2.1. Algorytmy symetryczne 15

2.2.2. Algorytmy asymetryczne 19

2.3. MECHANIZMY KONTROLI INTEGRALNOŚCI 22

2.3.1. Jednokierunkowa funkcja skrótu 22

2.3.2. Podpis cyfrowy 24

3. PRZEGLĄD I ANALIZA WYBRANYCH ALGORYTMÓW GENEROWANIA KLUCZY RUNDOWYCH 27

3.1. OGÓLNA STRUKTURA SZYFRU BLOKOWEGO 27

3.2. SZYFRY BLOKOWE O KONSTRUKCJI SIECI FEISTELA 33

3.2.1. DES (1975) 33

3.2.2. 3DES (TDEA) (1995) 40

3.2.3. LOKI97 (1997) 42

3.2.4. SM4 (2006) 49

3.2.5. KASUMI (1999/2007) 53

3.2.6. FeW (2014/2019) 59

3.2.7. LCB-IoT (2021) 64

3.3. SZYFRY BLOKOWE O KONSTRUKCJI SPN 69

3.3.1. AES (Rijndael) (1998) 69

3.3.2. Serpent (1998) 77

3.3.3. SAFER+ (1998) 83

3.3.4. PRESENT (2007) 88

3.3.5. PP-1 (2010) 94

3.3.6. PP-2 (2013) 101

3.4. SZYFRY BLOKOWE O KONSTRUKCJI INNEJ 109

3.4.1. IDEA (1991) 109

3.4.2. RC6 (1998) 114
3.4.3. Speck (2013) 120
3.5. ANALIZA PORÓWNAWCZA ALGORYTMÓW GENEROWANIA KLUCZY RUNDOWYCH 126
4. TESTOWANIE I OCENA JAKOŚCI ALGORYTMÓW GENEROWANIA KLUCZY RUNDOWYCH 136
4.1. METODOLOGIA TESTÓW 136
4.2. PAKIET TESTÓW STATYSTYCZNYCH NIST SP800-22 138
4.3. BADANIA I EKSPERYMENTY 152
4.3.1. Charakterystyka ogólna 153
4.3.2. Metoda A (próbek) 154
4.3.3. Metoda B (nadpróbek) 167
4.3.4. Metoda C (metapróbek) 171
4.3.5. Metoda D (podpróbek) 175
4.3.6. Metoda E (hybrydowa) 177
5. PODSUMOWANIE 197
6. DODATKI 204
SPISY 210
SPIS RYSUNKÓW 210
SPIS TABEL 214
SPIS ALGORYTMÓW 219
LITERATURA 220

Oznaczenia

Operatory

$\oplus$ – suma wyłączająca (XOR) słów o takiej samej długości

$\boxplus$ – dodawanie modulo 2^n

$\boxminus$ – odejmowanie modulo 2^n

$\odot$ – mnożenie modulo $(2^n + 1)$, gdzie n-bitowa wartość 0 reprezentuje liczbę 2^n

$\sim$ – negacja (NOT)

$\wedge$ – koniunkcja (AND)

$\vee$ – dysjunkcja (OR)

$||$ – konkatenacja

$\ll b$ – przesunięcie w lewo o b bitów

$\lll b$ – cykliczne przesunięcie (rotacja) w lewo o b bitów

$\lll_d b$ – cykliczne przesunięcie (rotacja) w lewo o b bitów, sterowane danymi

0^n – ciąg zer o długości n

1^n – ciąg jedynek o długości n

$\{0,1\}^n$ – n-krotny iloczyn kartezjański zbioru $\{0,1\}$

0xFF – liczba 255 w zapisie szesnastkowym

$\lfloor a \rfloor$ – największa liczba całkowita mniejsza od a lub równa a

$\lceil a \rceil$ – najmniejsza liczba całkowita większa od a lub równa a

$|a|$ – wartość bezwzględna liczby a

$a|b$ – a dzieli b

b – skrót słowa bit

f^{-1} – funkcja odwrotna do funkcji f

$|x|$ – długość słowa x w bitach

Zmienne i funkcje

c – tekst zaszyfrowany, szyfrogram (ang. *ciphertext*)

c_j – blok tekstu zaszyfrowanego o numerze j

c_A – tekst zaszyfrowany użytkownika A

c_B – tekst zaszyfrowany użytkownika B

d – miara podobieństwa – odległość

D – funkcja deszyfrująca (ang. *decryption function*)

DF – funkcja deszyfrująca zależna od kluczy rundowych

D_k – funkcja deszyfrująca z ustaloną wartością klucza *k*

D_v – macierz odległości

E – funkcja szyfrująca (ang. *encryption function*)

EF – funkcja szyfrująca zależna od kluczy rundowych,

E_k – funkcja szyfrująca z ustaloną wartością klucza *k*

f – funkcja bazowa sieci Feistela

h – funkcja skrótu (ang. *hash function*)

H_0 – hipoteza zerowa (ang. *null hypothesis*)

H_a – hipoteza alternatywna (ang. *alternative hypothesis*)

IF – funkcja iteracji (ang. *iteration function*)

k – klucz szyfrujący, klucz tajny / główny (ang. *secret / master key*)

k_i – klucz rundowy (*ang. round key*)

k_A – klucz użytkownika A

k_B – klucz użytkownika B

k_{prv} – klucz prywatny (ang. *private key*)

k_{pub} – klucz publiczny (ang. *public key*)

K – przestrzeń klucza (ang. *key space*)

KA – warstwa dodania klucza (ang. *key addition layer*)

h – funkcja skrótu (ang. *hash function*)

l – długość słowa w bitach

m – wiadomość (ang. *message*), tekst jawny (ang. *plaintext*)

m_j – blok wiadomośći o numerze *j*

MK – klucz główny (ang. *master key*)

n – rozmiar bloku w bitach

nrk – liczba kluczy rundowych (ang. *number of round keys*)

P – warstwa dyfuzji (ang. *diffusion layer*), warstwa permutacji (P-blok)

r – liczba rund szyfru

rk – klucz rundowy (ang. *round key*)

R – liczba iteracji w *RKG*

RF – funkcja rundy (ang. *round function*)

RKG – funkcja generowania kluczy rundowych (ang. *round keys generation*)

rkp – próbka skonkatenowanych kluczy rundowych (ang. *round keys probe*)

S – skrót wiadomości, podpis (ang. *signature*) lub warstwa postawień (ang. *substitution layer*), funkcja podstawień (S-blok)

v – liczba wariantów algorytmu, liczba obiektów

z – liczba kryteriów projektowych, liczba cech

Akronimy

3GPP – *3rd Generation Partnership Project*

AES – *Advanced Encryption Standard*

ARX – *Addition, Rotation, XOR*

BBS – *Blum Blum Shub*, generator liczb pseudolosowych

DES – *Data Encryption Standard*

DSS – *Digital Signature Standard*

ECC – *Elliptic Curve Cryptography*

FIPS – *Federal Information Processing Standard*

IEC – *International Electrotechnical Commission*

IoT – *Internet of Things*, Internet rzeczy

ISO – *International Organization for Standardization*

IV – *Initialization Vector*

LFSR – *Linear Feedbeck Shift Register*

LSB – *Least Significant Bit*

MSB – *Most Significant Bit*

NIST – *National Institute of Standards and Technology*

NSA – *National Security Agency*

PI – poziom istotności, α

PA – przedział akceptacji testu statystycznego

RFID – *Radio Frequency Identification*

RSA – *Rivest–Shamir–Adleman*, asymetryczny algorytm kryptograficzny

SHA-3 – *Secure Hash Algorithm-3*

SPN – *Substitution Permutation Network*

TDEA – *Triple Data Encryption Algorithm*

UMTS – *Universal Mobile Telecommunications System*

WAPI – *WLAN Authentication and Privacy Infrastructure*

WLAN – *Wireless Local Area Network*

1. Wstęp

1.1. Tematyka rozprawy

Niniejsza rozprawa poświęcona jest ocenie algorytmów generowania kluczy rundowych w szyfrach blokowych.

Szyfry blokowe (ang. *block ciphers*) to klasa algorytmów szyfrów symetrycznych czyli szyfrów wykorzystujących w procesie szyfrowania i deszyfrowania ten sam klucz główny (ang. *master key*). Szyfry blokowe przetwarzają iteracyjnie dane, w *n*-bitowych blokach z wykorzystaniem $|k|$-bitowego klucza, dlatego często nazywane są również iteracyjnymi szyframi blokowymi. Pojedyncza iteracja w szyfrze blokowym nazywana jest rundą szyfru.

Algorytmy generowania kluczy rundowych (ang. *key schedule*, *key expansion algorithm*) wykorzystywane są w szyfrach blokowych do wygenerowania z klucza głównego, będącego przeważnie ciągiem o długości pomiędzy 64 a 256 bitów, zbioru kilkunastu lub kilkudziesięciu kluczy rundowych (ang. *round keys*, *subkeys*). Klucze rundowe wykorzystywane są w rundach (iteracjach) właściwego procesu szyfrowania lub deszyfrowania. Zależnie od konstrukcji szyfru, w ramach jednej rundy wykorzystuje się jeden lub kilka kluczy rundowych. Generowanie kluczy rundowych odbywa się zazwyczaj jednokrotnie, przed właściwym szyfrowaniem bloków i jest procesem czasochłonnym. Istotną cechą jest to, aby wygenerowane przez algorytm klucze rundowe były niezależne, czyli o właściwościach statystycznych zbliżonych do ciągów losowych. Niezależność kluczy rundowych wpływa na proces kryptoanalizy szyfru. Jeśli klucze rundowe w szyfrze są niezależne to kryptoanaliza szyfru jest trudniejsza i wymaga więcej zasobów, co wykazali ponad 25 lat temu Eli Biham i Adi Shamir w pracach [15][16][1] oraz inni naukowcy w późniejszych publikacjach, m.in. [66]. W pracy [71], autorstwa Larsa Knudsena oraz Johna Mathiassena, przedstawiono również ciekawy eksperyment wykazujący, że szyfry blokowe z kluczami rundowymi wygenerowanymi przez bardziej skomplikowany algorytm są odporniejsze na kryptoanalizę, a szyfry z kluczami wytworzonymi z wykorzystaniem prostych przekształceń lub posiadającymi defekty statystyczne, są łatwiejsze do złamania.

Do oceny właściwości statystycznych ciągów pseudolosowych (za takie ciągi należy uznać szyfrogram oraz klucze rundowe), w tym do wykrywania defektów statystycznych, zastosowanie mają różnego rodzaju testy statystyczne, m.in. testy NIST SP800-22

[1] Publikacje cytowane łącznie ponad 5300 razy.

wykorzystane w konkursie na standard AES (ang. *Advanced Encryption Standard*) do oceny szyfrów blokowych [103][104].

W rozprawie wykorzystano testy NIST SP800-22 do oceny algorytmów generowania kluczy rundowych, na podstawie sekwencji kluczy rundowych, o długości kilkuset lub kilku tysięcy bitów dla pojedynczej wartości klucza głównego.

Projektując algorytm generowania kluczy rundowych należy znaleźć kompromis pomiędzy szybkością generowania kluczy, a bezpieczeństwem, w szczególności kiedy projektowany szyfr ma być „lekki" (ang. *lightweight*) czyli stosowany w środowisku o małych, ograniczonych zasobach, np. karta RFID [32]. Przy takich ograniczeniach problematyczne staje się określenie optymalnych struktur spełniających zadane kryteria.

Wielu autorów szyfrów blokowych skupia się na tym, by szyfr blokowy posiadał odpowiednią liczbę rund, długość klucza głównego (przestrzeń klucza) [8], a szyfrogram jak najlepiej rozpraszał (ang. *diffusion*) i zaciemniał (ang. *confusion*) bity tekstu jawnego. Zgodnie z przyjętymi w świecie kryptografii zasadami, nadrzędnym celem jest to, aby szyfr blokowy, jako całość, był odporny na znane ataki kryptograficzne [43][79]. Do najważniejszych ataków należą: kryptoanaliza liniowa (ang. *linear cryptanalysis*), opracowana w 1993 roku przez Mitsuru Matsui [80][81] i kryptoanaliza różnicowa (ang. *differential cryptanalysis*), opublikowana w 1991 roku, autorstwa wspomnianych wcześniej Eli Bihama i Adi Shamira [15] oraz ich odmiany, m.in. kryptoanaliza z powiązanymi kluczami (ang. *related-key cryptanalysis*) [17][28][29][108][109], atak z poślizgiem (ang. *slide attack*) [24][25], atak bumerangowy (ang. *boomerang attack*) [56][69] [105], atak prostokątny (ang. *rectangle attack*) [20][22][59][69] czy kryptoanaliza z niemożliwymi różnicami (ang. *impossible differential cryptanalysis*) [18][23][27][2], a także kryptoanaliza różnicowo-liniowa (ang. *differential-linear cryptanalysis*) [7][33][57], będąca połączeniem obu technik, zaprezentowana po raz pierwszy przez Susan Langford i Martina Hellmana w 1994 roku w pracy [75].

Zdecydowanie mniej uwagi poświęca się algorytmom generowania kluczy rundowych, często ograniczając się w opisie szyfru wyłącznie do przedstawienia zastosowanych operacji, niezbędnych do wytworzenia zestawu tych kluczy. Z reguły twórcy szyfrów blokowych pomijają szczegóły dotyczące kryteriów projektowych algorytmów generowania kluczy

[2] Autor rozprawy pragnie zaznaczyć, że kryptoanaliza jest bardzo rozległą i złożoną dziedziną nauki, sama w sobie nie jest przedmiotem niniejszej rozprawy, a szczegółowy opis i analiza ataków kryptoanalitycznych wykracza poza jej zakres.

rundowych czy też motywacji zastosowania danego rodzaju struktur zakładając, że dobrze zaprojektowana runda szyfru niweluje ewentualne słabości kluczy rundowych. Niemniej można spotkać prace, takie jak wspomniana wcześniej [71] czy też prace [40][41] oraz [60][61][66][72][84][93], w których dyskutowane są kwestie pożądanych własności algorytmów generowania kluczy rundowych, m.in. maksymalizacja efektu lawinowości czy też stosowanie struktur nieliniowych.

Podczas projektowania algorytmu generowania kluczy rundowych warto rozważyć i przetestować różne warianty zastosowanych operacji, np. rotacji (cyklicznego przesunięcia), permutacji, podstawień, stałych lub innych przekształceń.

Wiele wariantów (1, 2, …, *v*) algorytmu generowania kluczy rundowych może spełniać zadane kryteria projektowe (1, 2, …, *z*) w różnym stopniu. Problemem jest znalezienie wśród nich wariantu optymalnego, najbliższego hipotetycznie najlepszemu rozwiązaniu dla rozważanych i testowanych wariantów.

W szczególności dotyczy to przypadku łączenia wariantów algorytmu, spełniających kryteria projektowe, w nowy wariant (punkt 4.3.6).

1.2. Cel i teza rozprawy

Celem rozprawy jest zaproponowanie wykorzystania testów statystycznych oraz analizy skupień (ang. *cluster analysis*), jako elementu wspomagającego i wzbogacającego ocenę konstrukcji algorytmów generowania kluczy rundowych w szyfrach blokowych.

Ogólnie sformułowaną tezę rozprawy można przedstawić następująco:

Proponowana metoda „hybrydowa" oceny jakości algorytmów generowania kluczy rundowych szyfrów blokowych, oparta na testach statystycznych i analizie skupień, umożliwia wyznaczenie optymalnego wariantu algorytmu, najbliższego hipotetycznie najlepszemu rozwiązaniu dla zbioru testowanych wariantów, spełniających zadane kryteria projektowe.

Poszczególne zadania cząstkowe zrealizowane w rozprawie to:

- Opracowanie metody testowania i oceny algorytmów generowania kluczy rundowych opartej na testach statystycznych oraz analizie skupień.
- Przegląd i analiza różnych konstrukcji algorytmów generowania kluczy rundowych szyfrów blokowych (m.in. DES, LOKI97, KASUMI, AES (Rijndael), Serpent, PP-1, PP-2, IDEA, RC6, Speck).
- Ocena jakości wybranych algorytmów generowania kluczy rundowych.

1.3. Struktura rozprawy

Rozprawa składa się ze spisu treści, wykazu oznaczeń, pięciu rozdziałów głównych, dodatków, spisu rysunków, spisu tabel, spisu algorytmów oraz spisu literatury.

Pierwszy rozdział zawiera wprowadzenie do tematyki rozprawy, z przedstawieniem problemu badawczego, celu i tezy rozprawy oraz struktury rozprawy.

Rozdział drugi poświęcony jest podstawowym aspektom bezpieczeństwa. Przedstawiono w nim mechanizmy ochrony poufności, w tym algorytmy symetryczne ze szczególnym uwzględnieniem szyfrów blokowych i trybów ich pracy oraz algorytmy asymetryczne, a także mechanizmy kontroli integralności, w tym jednokierunkową funkcję skrótu i podpis cyfrowy.

W rozdziale trzecim dokonano obszernego przeglądu i analizy konstrukcji algorytmów generowania kluczy rundowych wybranych szyfrów blokowych (m.in. DES, LOKI97, KASUMI, AES (Rijndael), Serpent, PP-1, PP-2, IDEA, RC6, Speck). Dokonując wyboru omawianych szyfrów kierowano się ich popularnością oraz interesującą strukturą. Część przedstawionych szyfrów zakwalifikowano do konkursu na standard AES. Dla każdego z zaprezentowanych algorytmów, przedstawiono jego konstrukcję i elementy składowe (zastosowane operacje liniowe lub nieliniowe).

W rozdziale czwartym przedstawiono metody testowania i oceny jakości generatorów ciągów losowych oraz ciągów pseudolosowych. Przedstawiono metodologię testów, w tym zasady badania hipotez statystycznych H_0 oraz H_a oraz omówiono pakiet testów statystycznych NIST SP800-22. W tym samym rozdziale zaprezentowano wyniki badań i eksperymentów przeprowadzonych przez autora rozprawy. Przedstawiono sposób doboru kluczy głównych oraz generowania danych wejściowych dla testów statystycznych NIST SP800-22. W ramach rozprawy zaproponowano następujących pięć metod oceny jakości algorytmów generowania kluczy rundowych w szyfrach blokowych:

- **Metoda A (próbek)** – polega na przebadaniu binarnych sekwencji, wytwarzanych przez algorytm generowania kluczy rundowych, wybranymi testami NIST SP800-22. Przykładem zastosowania metody A są badania przeprowadzone nad wybranymi algorytmami generowania kluczy rundowych w szyfrach DES, IDEA, KASUMI, PP-1 i PP-2 [3] oraz nad różnymi wariantami modyfikacji algorytmu generowania kluczy rundowych w szyfrze PP-1 [4].
- **Metoda B (nadpróbek)** – polega na przeprowadzeniu wszystkich rodzajów testów statystycznych pakietu NIST SP800-22, jeśli konstrukcja algorytmu generowania

kluczy rundowych pozwala na wygenerowanie większej liczby kluczy, bez modyfikacji elementów składowych algorytmu, a jedynie przez zwiększenie liczby jego iteracji. Przykładem zastosowania metody B są badania przeprowadzane nad rozszerzonymi (ze zmienionym warunkiem stopu) algorytmami generowania kluczy rundowych szyfrów PP-1 oraz PP-2 [6].

- **Metoda C (metapróbek)** – polega na przeprowadzeniu wszystkich testów statystycznych NIST SP800-22, dla algorytmu generowania kluczy rundowych, niezależnie od konstrukcyjnego ograniczenia jego liczby itearcji. Metodę C zastosowano w odniesieniu do algorytmów generowania kluczy rundowych szyfrów: DES, IDEA, KASUMI, PP-1 oraz PP-2.
- **Metoda D (podpróbek)** – polega na przeprowadzeniu wybranych testów NIST SP800-22, na skróconych próbkach, w celu oceny jakości początkowych kluczy rundowych algorytmu. Metodę D zastosowano w odniesieniu do algorytmów generowania kluczy rundowych szyfrów: DES, IDEA, KASUMI, PP-1 oraz PP-2.
- **Metoda E (hybrydowa)** – polega na przeprowadzeniu testów statystycznych NIST SP800-22, dla różnych wariantów algorytmu generowania kluczy rundowych, w połączeniu z analizą skupień (taksonomią wrocławską), w celu wyznaczenia rozwiązania najbliższego hipotetycznie najlepszemu wariantowi algorytmu. Taksonomia wrocławska to metoda analizy skupień, stosowana do łączenia pewnych obiektów (zmiennych) w grupy jednorodne pod względem *z*-cech (wymiarów). Przykładem zastosowania metody E (hybrydowej) są przeprowadzone przez autora badania nad wyznaczeniem najlepszego cyklicznego przesunięcia w lewo (rotacji w lewo) w algorytmie generowania kluczy rundowych szyfru IDEA [5]. Innym przykładem wykorzystania metody E (hybrydowej) jest zastosowanie jej do wyznaczenia optymalnego wariantu modyfikacji szyfru PP-1. Spośród dwunastu rozważanych wariantów algorytmu generowania kluczy rundowych wybrano te warianty, które spełniają kryterium jakości, jakim są testy statystyczne. W analizie skupień rozważono zatem sześć wariantów algorytmu (oryginalny, bez rotacji, z operacjami XOR, ze zredukowaną liczbą S-bloków do sześciu, z pozytywnymi (prostymi) modyfikacjami, hipotetycznie najlepszy) z uwzględnieniem dziesięciu cech (dziewięć cech stanowią wyniki poszczególnych testów statycznych, natomiast dziesiątą cechą jest czas generowania kluczy).

Rozdział piąty zawiera wnioski z otrzymanych wyników i podsumowanie niniejszej rozprawy, w tym ocenę wartości teoretycznej i praktycznej proponowanych rozwiązań.

2. Podstawy mechanizmów kryptograficznych

Rozdział poświęcony jest podstawowym aspektom bezpieczeństwa w systemach informatycznych oraz mechanizmom kryptograficznym je realizującym, ze szczególnym uwzględnieniem szyfrów blokowych.

2.1. Charakterystyka ogólna

Podstawowymi aspektami bezpieczeństwa są poufność, integralność i dostępność. Wyróżnić należy również niezaprzeczalność oraz dodatkowe procesy, m.in. uwierzytelnienie i kontrola dostępu (autoryzacja). Kryptografia jest dziedziną, która pozwala osiągnąć cele wymienionych aspektów bezpieczeństwa oraz procesów.

Poufność (ang. *confidentiality*) – zdolność do transmisji (lub przechowywania) danych bez ujawnienia jakiejkolwiek ich części nieautoryzowanym jednostkom.

Integralność (ang. *integrity*) – zdolność do transmisji (lub przechowywania) danych w taki sposób, aby jakiekolwiek modyfikacje (nieprawidłowe lub nieautoryzowane) mogły być wykryte.

Dostępność (ang. *availability*) – zdolność do obsługi jednostek upoważnionych do korzystania z chronionych zasobów.

Niezaprzeczalność (ang. *nonrepudiation*) – zdolność do wykazania, że jeśli określona czynność została wykonana przez jednostkę, to nie można w żaden sposób zaprzeczyć faktowi wykonania tej czynności przez tę jednostkę.

Uwierzytelnienie (ang. *authentication*) – proces ustalania tożsamości jednostki.

Kontrola dostępu (ang. *access control*) lub **autoryzacja** (ang. *authorization*) – proces, w ramach którego przydzielane są odpowiednie prawa dostępu jednostce, np. po zakończeniu procesu uwierzytelnienia [101].

2.2. Mechanizmy ochrony poufności

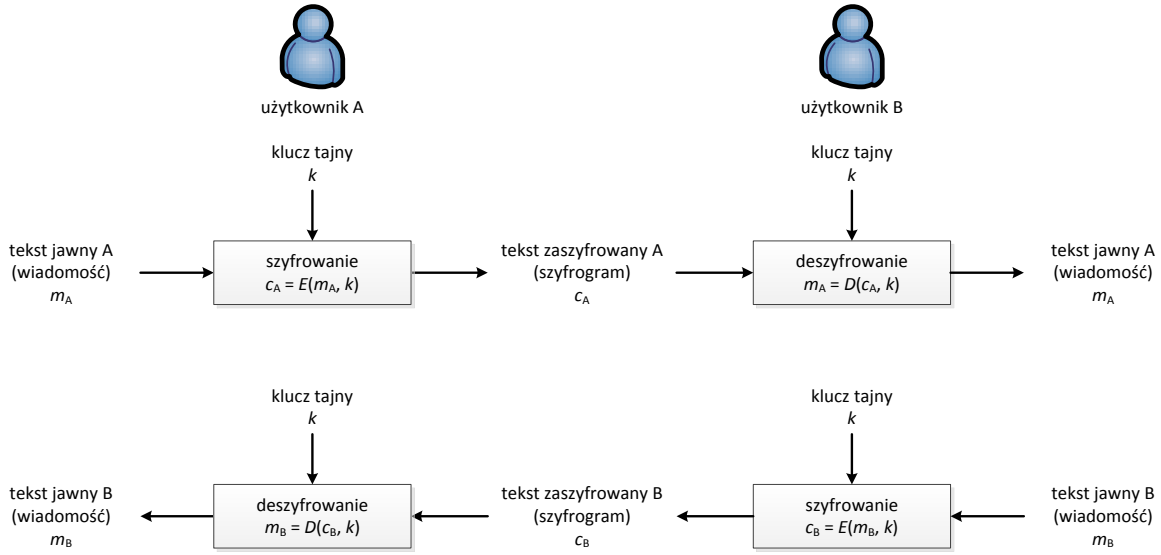
Poufność osiąga się za pomocą szyfrowania danych. Szyfrowanie jest to proces przekształcenia, z wykorzystaniem określonego algorytmu kryptograficznego, tekstu jawnego w tekst zaszyfrowany (szyfrogram) mający nieczytelną postać. Deszyfrowanie jest procesem odwrotnym, polegającym na przywróceniu zaszyfrowanemu tekstowi oryginalnej postaci. Cel ten można osiągnąć z wykorzystaniem symetrycznych lub asymetrycznych algorytmów

kryptograficznych. Szyfry (algorytmy) symetryczne dzielone są dalej na szyfry strumieniowe oraz szyfry blokowe, którym poświęcona jest głównie niniejsza rozprawa.

2.2.1. Algorytmy symetryczne

W algorytmach symetrycznych wykorzystuje się ten sam tajny klucz do szyfrowania oraz deszyfrowania wiadomości. Ponieważ nadawca i odbiorca posiadają wspólny klucz to algorytmy symetryczne nazywane są również algorytmami z kluczem współdzielonym.

Proces szyfrowania i deszyfrowania przy użyciu algorytmu symetrycznego jest pod względem matematycznym znacznie prostszy, aniżeli z użyciem algorytmu asymetrycznego (punkt 2.2.2), dlatego też algorytmy symetryczne cechują się znacznie większą wydajnością w porównaniu z algorytmami asymetrycznymi.



Rys. 2.1. Zapewnienie poufności z wykorzystaniem kryptografii symetrycznej

Rysunek (rys. 2.1) przedstawia użytkownika A oraz użytkownika B, którzy komunikują się ze sobą z wykorzystaniem algorytmu symetrycznego. Zarówno użytkownik A jak i użytkownik B korzystają z tego samego algorytmu kryptograficznego, przy użyciu którego dane są szyfrowane i deszyfrowane. Dysponują również tym samym wspólnym, tajnym kluczem $k \in K$, gdzie K to przestrzeń klucza, wykorzystywanym przez algorytm szyfrowania i deszyfrowania. Uzgodnienia dotyczące klucza i algorytmu dokonywane są przed właściwym procesem kryptograficznym, z wykorzystaniem protokołów negocjacyjnych lub bezpiecznego kanału komunikacyjnego.

Na rysunku (rys. 2.1) stosowane są następujące oznaczenia:

- m – wiadomość (ang. *message*), inna nazwa: tekst jawny (ang. *plaintext*),

- c – szyfrogram (ang. *cryptogram*), inna nazwa: tekst zaszyfrowany (ang. *ciphertext*),
- k – klucz tajny (ang. *secret key*), inne nazwy: klucz główny (ang. *master key*), klucz szyfru (ang. *cipher key*),
- E – funkcja szyfrująca (ang. *encryption function*),
- D – funkcja deszyfrująca (ang. *decryption function*).

Algorytm 2.1 przedstawia proces bezpiecznej komunikacji między użytkownikami A i B, z wykorzystaniem kryptografii symetrycznej.

Algorytm 2.1. Zapewnienie poufności z wykorzystaniem kryptografii symetrycznej

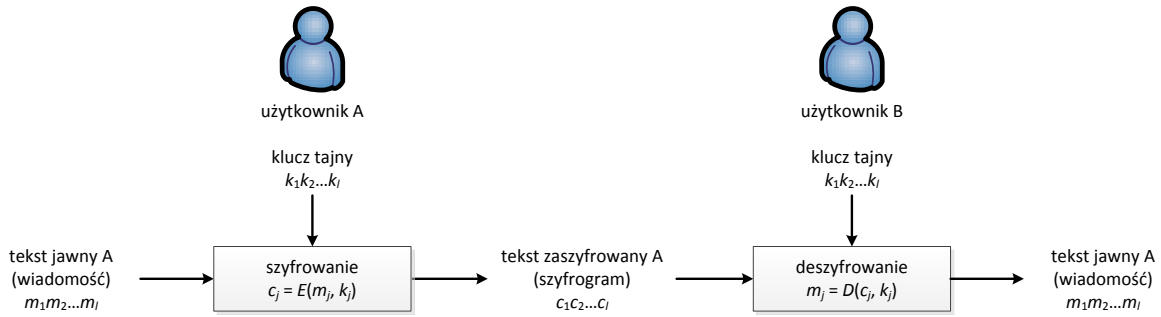
1. Użytkownik A przygotowuje wiadomość m_A, szyfruje ją z wykorzystaniem klucza tajnego, k i wysyła do użytkownika B zaszyfrowaną wiadomość c_A.
2. Użytkownik B otrzymuje wiadomość c_A, deszyfruje ją z wykorzystaniem klucza tajnego, k, do jawnej postaci m_A.
3. Użytkownik B przygotowuje odpowiedź m_B, szyfruje ją z wykorzystaniem klucza tajnego, k i wysyła do użytkownika A zaszyfrowaną odpowiedź c_B.
4. Użytkownik A otrzymuje odpowiedź c_B, deszyfruje ją z wykorzystaniem klucza tajnego, k, do jawnej postaci m_B.

Warunkiem poprawnego działania całego procesu komunikacji z wykorzystaniem szyfru symetrycznego, jest spełnienie zależności:

$$m = D(E(m,k),k). \tag{2.1}$$

Oznacza to, że szyfrogram c, otrzymany w wyniku działania funkcji szyfrującej E z wykorzystaniem klucza k, możliwy jest do przekształcenia, za pomocą funkcji deszyfrującej D z wykorzystaniem klucza k, do postaci tekstu jawnego m. Przetwarzane informacje pozostają bezpieczne do momentu, aż klucz k zostanie skompromitowany (ujawniony), jednak bezpieczeństwo nie zależy wyłącznie od tajności klucza k, ale również od jego długości.

W ramach szyfrów symetrycznych można wyróżnić szyfry strumieniowe (ang. *stream ciphers*), wykorzystujące klucz k o długości niemniejszej od długości wiadomość m ($|k| \geq |m|$) i szyfry blokowe (ang. *block ciphers*), wykorzystujące klucz o stałej długości – niezależnej od długości wiadomości.

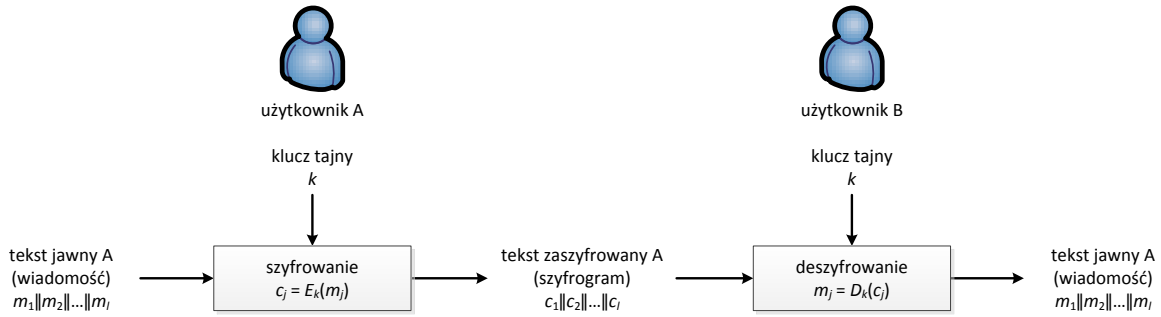


Rys. 2.2. Szyfr strumieniowy – szyfrowanie i deszyfrowanie

Działanie szyfru strumieniowego zilustrowano na rysunku (rys. 2.2). Tekst jawny m i tekst zaszyfrowany c traktowane są jako strumienie znaków $m_1m_2…m_l$ i $c_1c_2…c_l$, gdzie l oznacza długość strumieni. Klucz tajny k także traktowany jest jako strumień znaków $k_1k_2…k_l$. Szyfrowanie i deszyfrowanie tekstu odbywa się znak po znaku, z wykorzystaniem odpowiedniego znaku klucza. Dla $j = 1, 2,…, l$, podczas szyfrowania wykonywane jest obliczenie $c_j = E(m_j, k_j)$, a podczas deszyfrowania – obliczenie $m_j = D(c_j, k_j)$.

Jako parę wzajemnie odwrotnych funkcji E i D wykorzystać można operacje $\boxplus$ i $\boxminus$, tj. dodawanie i odejmowanie modulo 2^n (n jest długością znaku w bitach) lub operacje $\oplus$ i $\oplus$, tj. XOR znaków o długość n bitów (dla $n = 1$ obie pary operacji są sobie równe).

Nadawca i odbiorca zamiast uzgadniać długi strumień klucza mogą uzgodnić generator strumienia klucza (np. generator BBS [101]) i jego parametry. Od jakości generatora zależy bezpieczeństwo szyfru. Zastosowanie generatora wytwarzającego powtarzający się (cykliczny) strumień znaków klucza jest niepożądane. Jeśli wytworzony przez generator strumień klucza jest losowy, ma długość niemniejszą niż szyfrowany strumień wiadomości i użyty zostanie jednokrotnie to taki szyfr, nazywany szyfrem jednokrotnym, jest bezwarunkowo bezpieczny [102].



Rys. 2.3. Szyfr blokowy – szyfrowanie i deszyfrowanie

Działanie szyfru blokowego zilustrowano na rysunku (rys. 2.3). Tekst jawny m i tekst zaszyfrowany c traktowane są jako konkatenacja n-bitowych bloków $m_1||m_2||…||m_l$ i

$c_1||c_2||\ldots||c_l$. Szyfrowanie i deszyfrowanie tekstu odbywa się blok po bloku, z wykorzystaniem tego samego klucza tajnego *k*. W celu podkreślenia stałej wartości klucza, podczas szyfrowania lub deszyfrowania wszystkich bloków tekstu, dla funkcji *E* i *D* stosowana jest notacja E_k *i* D_k. Dla $j = 1, 2, \ldots, l$, podczas szyfrowania wykonywane jest obliczenie $c_j = E_k(m_j)$, a podczas deszyfrowania – obliczenie $m_j = D_k(c_j)$.

Dowolny szyfr blokowy może być stosowany w kilku trybach. Do najpopularniejszych trybów pracy szyfrów blokowych należą [96][102]:

- ECB (ang. *Electronic Codebook*),
- CBC (ang. *Cipher Block Chaining*),
- CFB (ang. *Cipher Feedback*),
- OFB (ang. *Output Feedback*).

Tryb elektronicznej książki kodowej ECB szyfruje niezależnie każdy z *n*-bitowych bloków m_j przy użyciu tego samego tajnego klucza *k*, w rezultacie ten sam blok tekstu jawnego zostanie zawsze przekształcony w taki sam blok tekstu zaszyfrowanego c_j. Jest to niewątpliwie słabość tego trybu, która może być wykorzystana w kryptoanalizie. Dla $j = 1, 2, \ldots, l$, szyfrowanie odbywa się zgodnie z formułą:

$$c_j = E_k(m_j), \tag{2.2}$$

a deszyfrowanie wykonywane jest następująco:

$$D_k(c_j) = E_k^{-1}\left(E_k(m_j)\right) = m_j. \tag{2.3}$$

Pozostałe wymienione tryby wyposażone są w mechanizmy wprowadzania do szyfrowanych wiadomości elementów pseudolosowych, w konsekwencji czego szyfrowanie tego samego tekstu jawnego wielokrotnie, spowoduje otrzymanie różnych wyników.

W trybie szyfrowego wiązania blokowego CBC na każdym bloku tekstu jawnego wykonywana jest operacja XOR z poprzednim blokiem szyfrogramu. Pierwszy blok zostaje natomiast poddany operacji XOR z wektorem inicjującym *IV* (ang. *initialization vector*), który ma postać pseudolosowego ciągu bitów. Dla $j = 1, 2, \ldots, l$ oraz $c_0 = IV$ szyfrowanie odbywa się zgodnie z formułą:

$$c_j = E_k(m_j \oplus c_{j-1}), \tag{2.4}$$

a deszyfrowanie wykonywane jest następująco:

$$D_k(c_j) \oplus c_{j-1} = E_k^{-1}\left(E_k(m_j \oplus c_{j-1})\right) \oplus c_{j-1} = (m_j \oplus c_{j-1}) \oplus c_{j-1} = m_j. \tag{2.5}$$

Tryb szyfrowego sprzężenia zwrotnego CFB wykorzystuje zaszyfrowany poprzedni blok szyfrogramu c_{j-1} i blok jawny m_j w operacji XOR. Dla j = 1, 2, …, l oraz $c_0 = IV$ szyfrowanie odbywa się zgodnie z formułą:

$$c_j = m_j \oplus E_k(c_{j-1}), \tag{2.6}$$

a deszyfrowanie wykonywane jest następująco:

$$c_j \oplus E_k(c_{j-1}) = \left(m_j \oplus E_k(c_{j-1})\right) \oplus E_k(c_{j-1}) = m_j. \tag{2.7}$$

Szyfrowanie w tym trybie polega na szyfrowaniu fragmentów, np. o długości 16 bitów, a nie całych bloków o długości n [96].

W trybie wyjściowego sprzężenia zwrotnego OFB blok tekstu jawnego m_j jest poddawany operacji XOR z blokiem wyjściowym x_j, gdzie $x_j = E_k(x_{j-1})$ dla j = 1, 2, …, l i $x_0 = IV$. Szyfrowanie odbywa się zgodnie z formułą:

$$c_j = m_j \oplus x_j, \tag{2.8}$$

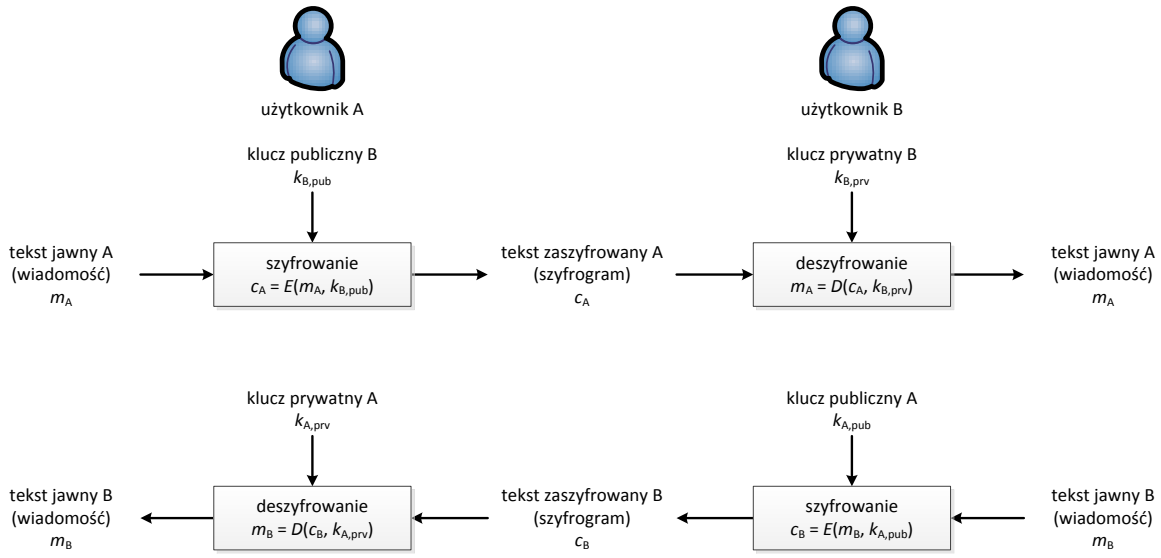
a deszyfrowanie wykonywane jest następująco:

$$c_j \oplus x_j = (m_j \oplus x_j) \oplus x_j = m_j. \tag{2.9}$$

2.2.2. Algorytmy asymetryczne

Szyfrowanie i deszyfrowanie asymetryczne może być stosowane z wykorzystaniem tego samego algorytmu lub algorytmów uzupełniających się. Wymagane jest natomiast stosowanie dwóch różnych, powiązanych ze sobą kluczy: klucza publicznego oraz klucza prywatnego. Klucz prywatny pozostaje tajny i znany jest wyłącznie jego posiadaczowi, natomiast klucz publiczny jest kluczem ogólnodostępnym dla osób trzecich. Choć klucze te są powiązane ze sobą to niemożliwe jest wyznaczenie klucza prywatnego na podstawie znajomości klucza publicznego. Po ewentualnym ujawnieniu klucza prywatnego atakujący może podszywać się pod jego właściciela, wysyłając i odbierając wiadomości w jego imieniu. Kryptografia asymetryczna oparta jest na problemach trudnych obliczeniowo [96].

Tekst jawny zaszyfrowany z wykorzystaniem klucza publicznego k_{pub} może zostać odszyfrowany wyłącznie przy użyciu klucza prywatnego k_{prv}. W przypadku przetworzenia tekstu jawnego z wykorzystaniem klucza prywatnego k_{prv}, odtworzenie tekstu możliwe jest wyłącznie z wykorzystaniem klucza publicznego k_{pub}. Jednak w przypadku przetworzenia tekstu jawnego z wykorzystaniem klucza prywatnego k_{prv} nie zapewnia się poufności danym lecz umożliwia się zweryfikowanie źródła pochodzenia informacji. Do najczęstszych zastosowań algorytmów asymetrycznych należą: zapewnienie poufności danych, uwierzytelnienie nadawcy, zapewnienie niezaprzeczalności oraz kontrola integralności.



Rys. 2.4. Zapewnienie poufności z wykorzystaniem kryptografii asymetrycznej

Rysunek (rys. 2.4) przedstawia sposób zapewnienia poufności za pomocą kryptografii asymetrycznej. Poniżej znajduje się również algorytm 2.2 bezpiecznej wymiany wiadomości między użytkownikami. Zakłada się, iż każdy z użytkowników posiada swoje klucze (prywatny i publiczny) oraz posiada klucz publiczny drugiego użytkownika (założenie to będzie prawdziwe dla każdego z przedstawionych w pracy zastosowań kryptografii asymetrycznej).

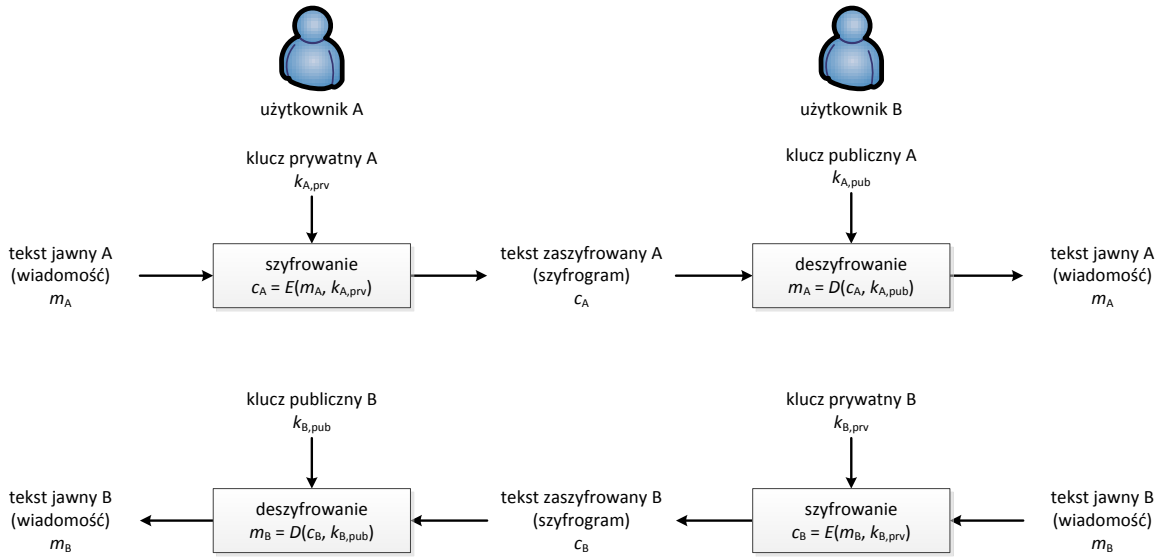
Algorytm 2.2. Zapewnienie poufności z wykorzystaniem kryptografii asymetrycznej

1. Użytkownik A przygotowuje wiadomość m_A, szyfruje ją z wykorzystaniem klucza publicznego użytkownika B, $k_{B,pub}$ i wysyła do użytkownika B zaszyfrowaną wiadomość c_A.
2. Użytkownik B otrzymuje wiadomość c_A, deszyfruje ją z wykorzystaniem własnego klucza prywatnego, $k_{B,prv}$, do jawnej postaci m_A.
3. Użytkownik B przygotowuje odpowiedź m_B, szyfruje ją z wykorzystaniem klucza publicznego użytkownika A, $k_{A,pub}$ i wysyła do użytkownika A zaszyfrowaną odpowiedź c_B.
4. Użytkownik A otrzymuje odpowiedź c_B, deszyfruje ją z wykorzystaniem własnego klucza prywatnego, $k_{A,prv}$, do jawnej postaci m_B.

Poufność jest zapewniona od momentu wysłania pierwszej wiadomości przez użytkownika A, ponieważ użytkownik B może ją odszyfrować tylko przy użyciu swojego własnego klucza prywatnego. Odpowiedź użytkownika B również zostaje poufna, ponieważ

odszyfrować może ją tylko użytkownik A, przy użyciu swojego klucza prywatnego. Warunkiem zapewnienia poufności jest:

$$m = D\left(E\left(m, k_{pub}\right), k_{prv}\right). \tag{2.10}$$



Rys. 2.5. Uwierzytelnienie nadawcy i zapewnienie niezaprzeczalności w kryptografii asymetrycznej

Rysunek (rys. 2.5) oraz algorytm 2.3 przedstawiają sposób wykorzystania kryptografii asymetrycznej w celu uwierzytelnienia nadawcy oraz zapewnienia niezaprzeczalności danych, podczas wymiany informacji między użytkownikami.

Algorytm 2.3. Realizacja uwierzytelnienia nadawcy i zapewnienia niezaprzeczalności w kryptografii asymetrycznej

1. Użytkownik A przygotowuje wiadomość m_A, szyfruje ją z wykorzystaniem własnego klucza prywatnego, $k_{A,prv}$ i wysyła do użytkownika B zaszyfrowaną wiadomość c_A.
2. Użytkownik B otrzymuje wiadomość c_A, deszyfruje ją z wykorzystaniem klucza publicznego użytkownika A, $k_{A,pub}$, do jawnej postaci m_A.
3. Użytkownik B przygotowuje odpowiedź m_B, szyfruje ją z wykorzystaniem własnego klucza prywatnego, $k_{B,prv}$ i wysyła do użytkownika A zaszyfrowaną odpowiedź c_B.
4. Użytkownik A otrzymuje odpowiedź c_B, deszyfruje ją z wykorzystaniem klucza publicznego użytkownika B, $k_{B,pub}$, do jawnej postaci m_B.

Warunkiem uwierzytelnienia i niezaprzeczalności jest:

$$m = D\big(E\big(m, k_{prv}\big), k_{pub}\big). \tag{2.11}$$

W celu zrealizowania poufności oraz uwierzytelnienia i niezaprzeczalności, a ponadto integralności danych za pomocą kryptografii asymetrycznej, należy zastosować podwójne szyfrowanie. Użytkownik A powinien zaszyfrować poufną wiadomość dla użytkownika B, korzystając z jego klucza publicznego $k_{\mathrm{B,pub}}$, a następnie przetworzyć ją ponownie, wykorzystując własny klucz prywatny $k_{\mathrm{A,prv}}$. Użytkownik B stosuje kolejno klucze $k_{\mathrm{A,pub}}$ i $k_{\mathrm{B,prv}}$.

Algorytmy kryptografii asymetrycznej najczęściej wykorzystuje się w aplikacjach wymagających uwierzytelniania przy pomocy podpisów cyfrowych oraz zarządzania kluczami. Klucze publiczne stosuje się również do szyfrowania współdzielonych kluczy sesji (ang. *session keys*) – kluczy wykorzystywanych w ramach pojedynczej sesji, które mogą być w ten sposób wymieniane i przesyłane za pośrednictwem sieci publicznej, bez ryzyka ujawnienia. Najpopularniejszymi algorytmami kryptografii asymetrycznej są: algorytm RSA, algorytm ElGamala oraz algorytmy wykorzystujące krzywe eliptyczne ECC (ang. *Elliptic Curve Cryptography*) [85][96].

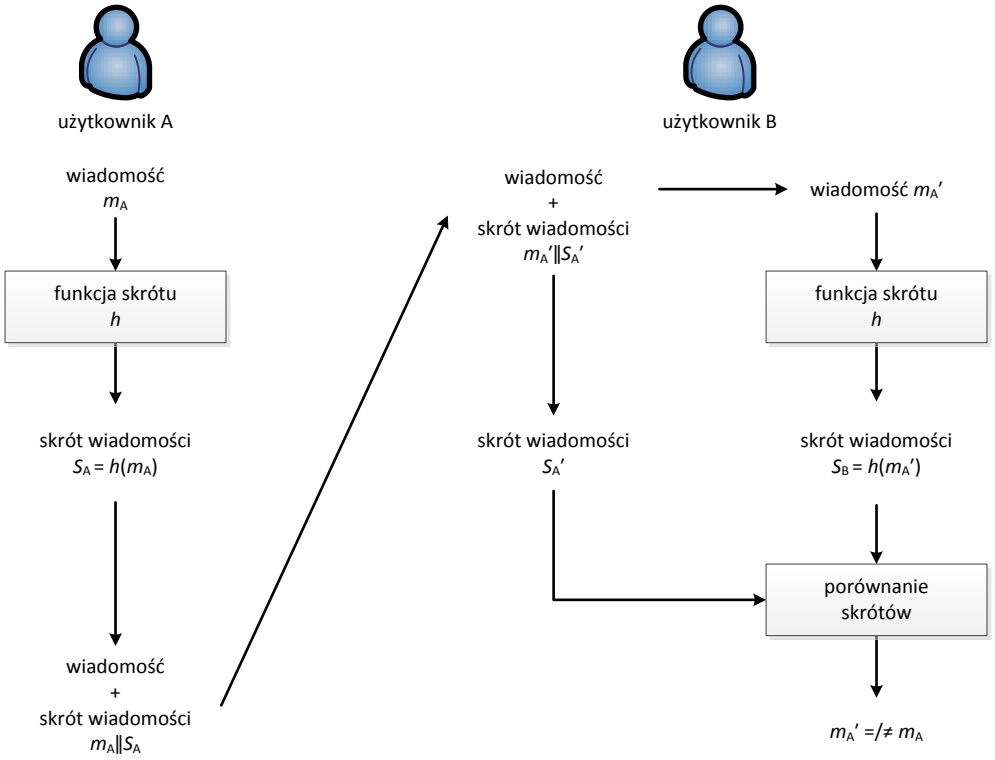
2.3. Mechanizmy kontroli integralności

Mechanizmy kontroli integralności wykorzystywane są w celu wykrywania nieautoryzowanych modyfikacji danych. W niniejszym podrozdziale zostały omówione podstawy funkcji skrótu (ang. *hash function*) oraz podpis cyfrowy (ang. *digital signature*).

2.3.1. Jednokierunkowa funkcja skrótu

Jednokierunkowa funkcja skrótu przetwarza ciąg bitów wejściowych o dowolnej długości w ciąg bitów wyjściowych o stałej długości. Funkcja skrótu powinna posiadać następujące własności, aby uznać ją za bezpieczną i przydatną w kryptografii [85][96]:

- wynik przetwarzania tych samych danych wejściowych musi być zawsze taki sam,
- musi być odporna na kolizje, tzn. by niemal niemożliwe było otrzymanie dwóch identycznych wyników dla dwóch różnych danych wejściowych,
- musi być jednokierunkowa, tzn. że ustalenie treści oryginalnej wiadomości na podstawie wyniku funkcji skrótu musi być bardzo trudne albo niemożliwe,
- musi być łatwo obliczalna, tzn. w czasie wielomianowym.



Rys. 2.6. Zastosowanie funkcji skrótu do weryfikacji integralności danych

Jednokierunkowe funkcje skrótu są zwykle stosowane w mechanizmach kontroli integralności danych. Rysunek (rys. 2.6) oraz algorytm 2.4 przedstawiają jedno z zastosowań funkcji skrótu. W celu weryfikacji integralności danych, komunikacja użytkowników może przebiegać w następujący sposób [101].

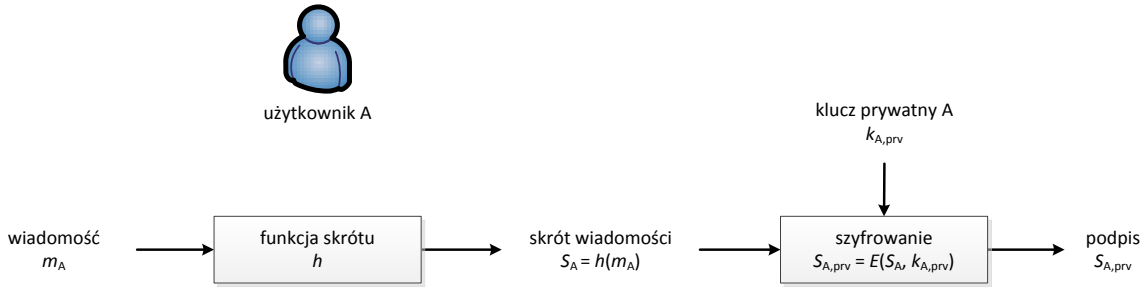
Algorytm 2.4. Weryfikacja integralności danych

1. Użytkownik A przygotowuje wiadomość m_A i oblicza skrót tej wiadomości, $S_A = h(m_A)$, z wykorzystaniem jednokierunkowej funkcji skrótu h.
2. Użytkownik A dołącza skrót S_A do wiadomości m_A i wysyła do użytkownika B wiadomość ze skrótem, $m_A||S_A$.
3. Użytkownik B otrzymuje wiadomość ze skrótem, $m_A'||S_A'$, ewentualnie zmienione podczas transmisji, a następnie, po oddzieleniu wiadomości od skrótu, oblicza skrót tej wiadomości, $S_B = h(m_A')$, z wykorzystaniem tej samej funkcji skrótu h, z jakiej korzystał użytkownik A.
4. Użytkownik B porównuje otrzymany skrót S_A' z obliczonym skrótem S_B. W przypadku $S_A' = S_B$, zachodzi $m_A' = m_A$, a więc oryginalna wiadomość m_A nie została zmieniona podczas transmisji.

Należy zauważyć, iż zastosowanie wyżej wymienionego mechanizmu nie zapobiega atakowi MitM (ang. *Man in the Middle*). Istnieje możliwość podsłuchania i przechwycenia wiadomości użytkownika A. Możliwe jest zmodyfikowanie jej, ponowne obliczenie skrótu wiadomości przez atakującego oraz dalsze przesłanie tak sfałszowanej wiadomości do użytkownika B. Użytkownik B postępując zgodnie z przedstawionym schematem zweryfikuje wiadomość i uzna ją za poprawną, tj. niezmodyfikowaną. Aby zapobiec temu i efektywnie wykorzystać wynik funkcji skrótu jako sygnaturę, można połączyć ją z kryptografią asymetryczną, w rezultacie czego powstaje podpis cyfrowy, omówiony w kolejnym punkcie, lub zastosować funkcję skrótu z kluczem. Przykładem popularnej jednokierunkowej funkcji skrótu jest funkcja Keccak [11], wybrana jako najlepsza w konkursie na standard SHA-3 (*Secure Hash Algorithm-3*) [53].

2.3.2. Podpis cyfrowy

Najczęściej stosowaną metodą generowania podpisu cyfrowego jest szyfrowanie, z wykorzystaniem klucza prywatnego, wyniku jednokierunkowej funkcji skrótu, uzyskanego dla podpisywanej wiadomości. Alternatywną metodą generowania podpisu może być zastosowanie kryptografii symetrycznej oraz zaufanej strony trzeciej. W poniższych rozważaniach ograniczono się do przypadku podpisywania wiadomości z wykorzystaniem kryptografii asymetrycznej.



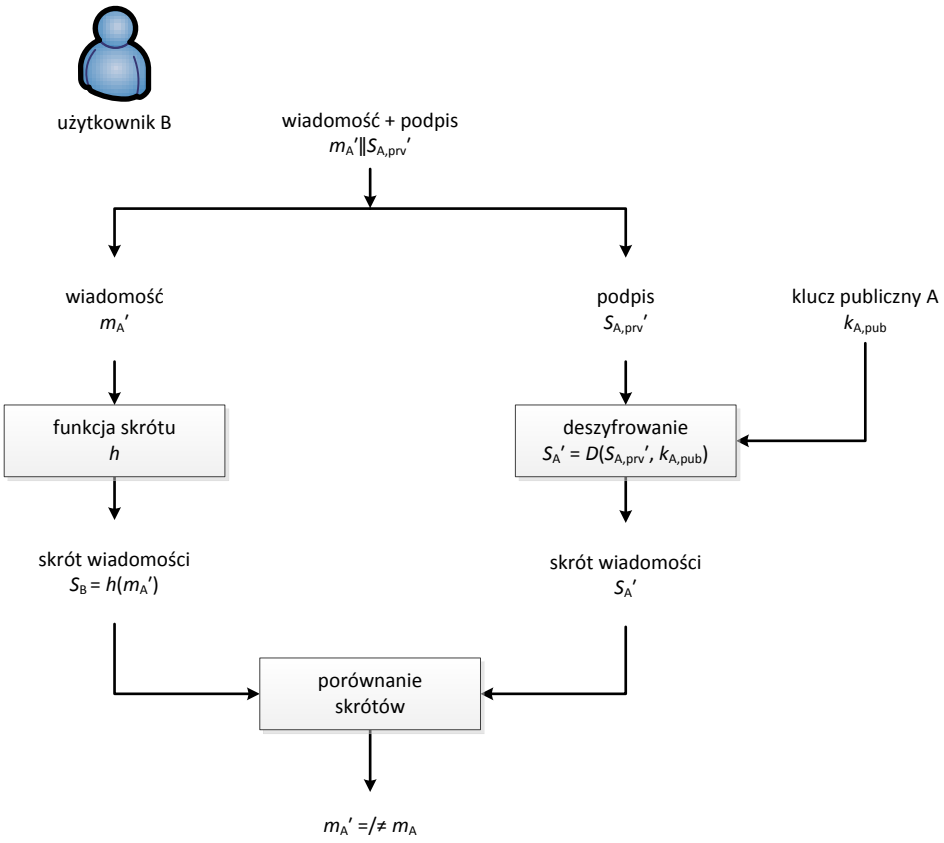
Rys. 2.7. Generowanie podpisu przez użytkownika A

Rysunek (rys. 2.7) i algorytm 2.5 przedstawiają proces generowania podpisu cyfrowego przez użytkownika A [85][96].

Algorytm 2.5. Generowanie podpisu przez użytkownika A

1. Użytkownik A przygotowuje wiadomość m_A i oblicza skrót tej wiadomości, $S_A = h(m_A)$, z wykorzystaniem jednokierunkowej funkcji skrótu h.

2. Użytkownik A szyfruje skrót S_A z wykorzystaniem własnego klucza prywatnego, $k_{A,prv}$, w wyniku czego powstaje podpis cyfrowy, $S_{A,prv}$.
3. Użytkownik A dołącza podpis $S_{A,prv}$ do wiadomości m_A i wysyła do użytkownika B wiadomość z podpisem, $m_A||S_{A,prv}$.



Rys. 2.8. Weryfikacja podpisu i integralności danych przez użytkownika B

Rysunek (rys. 2.8) oraz algorytm 2.6 przedstawiają proces weryfikacji podpisu cyfrowego i integralności danych przez użytkownika B.

Algorytm 2.6. Weryfikacja podpisu i integralności danych przez użytkownika B

1. Użytkownik B otrzymuje wiadomość z podpisem, $m_A'||S_{A,prv}'$, ewentualnie zmienione podczas transmisji, a następnie, po oddzieleniu wiadomości od podpisu, deszyfruje podpis $S_{A,prv}'$, z wykorzystaniem klucza publicznego użytkownika A, $k_{A,pub}$, w wyniku czego uzyskuje skrót S_A'.
2. Użytkownik B oblicza skrót wiadomości m_A', $S_B = h(m_A')$, z wykorzystaniem tej samej funkcji skrótu h, z jakiej korzystał użytkownik A.
3. Użytkownik B porównuje otrzymany skrót S_A' z obliczonym skrótem S_B. W przypadku $S_A' = S_B$, zachodzi $m_A' = m_A$.

Jeśli wynik obliczonego przez użytkownika B skrótu wiadomości S_B jest równy wartości skrótu wiadomości użytkownika A (otrzymanego w procesie deszyfrowania podpisu) S_A', można mieć pewność co do tożsamości nadawcy, a także co do integralności danych. Natomiast jeśli wynik obliczonego przez użytkownika B skrótu wiadomości S_B nie jest zgodny ze skrótem wiadomości użytkownika A (otrzymanego w procesie deszyfrowania podpisu) S_A', świadczy to o sfałszowanym podpisie lub naruszeniu integralności. Należy zauważyć, iż podpis cyfrowy nie zapewnia danym poufności. Najczęściej używanymi algorytmami generowania podpisu cyfrowego z kluczem publicznym są: RSA oraz DSS (ang. *Digital Signature Standard*).

3. Przegląd i analiza wybranych algorytmów generowania kluczy rundowych

W rozdziale przedstawiono ogólną charakterystykę algorytmów generowania kluczy rundowych oraz dokonano przeglądu tych algorytmów w wybranych szyfrach blokowych z podziałem ze względu na budowę szyfru, tj.:

- sieć Feistela: DES, 3DES, LOKI97, SM4, KASUMI, FeW, LCB-IoT,
- sieć SPN: Rijndael (AES), Serpent, SAFER+, PRESENT, PP-1, PP-2,
- inne sieci: IDEA, RC6, Speck.

Dokonując wyboru szyfrów kierowano się ich popularnością, interesującą konstrukcją, ale również ich znanymi słabościami. Szyfry DES (punkt 3.2.1), Triple DES (punkt 3.2.2) i AES (punkt 3.3.1) są standardowymi szyframi blokowymi Narodowego Instytutu Standardów i Technologii USA (NIST).

Do konkursu na szyfr AES o bloku 128 bitów i kluczu o długości 128, 192 i 256 bitów, ogłoszonego przez NIST w 1997 roku, zakwalifikowano szyfry CAST-256, CRYPTON, DEAL, DFC, E2, FROG, HPC, LOKI97 (punkt 3.2.3), MAGENTA, MARS, RC6 (punkt 3.4.2), Rijndael (punkt 3.3.1), SAFER+ (punkt 3.3.3), Serpent (punkt 3.3.2) i Twofish. W finalne tego konkursu znalazły się MARS, RC6, Rijndael, Serpent i Twofish, a jako najlepszy szyfr uznano Rijndael.

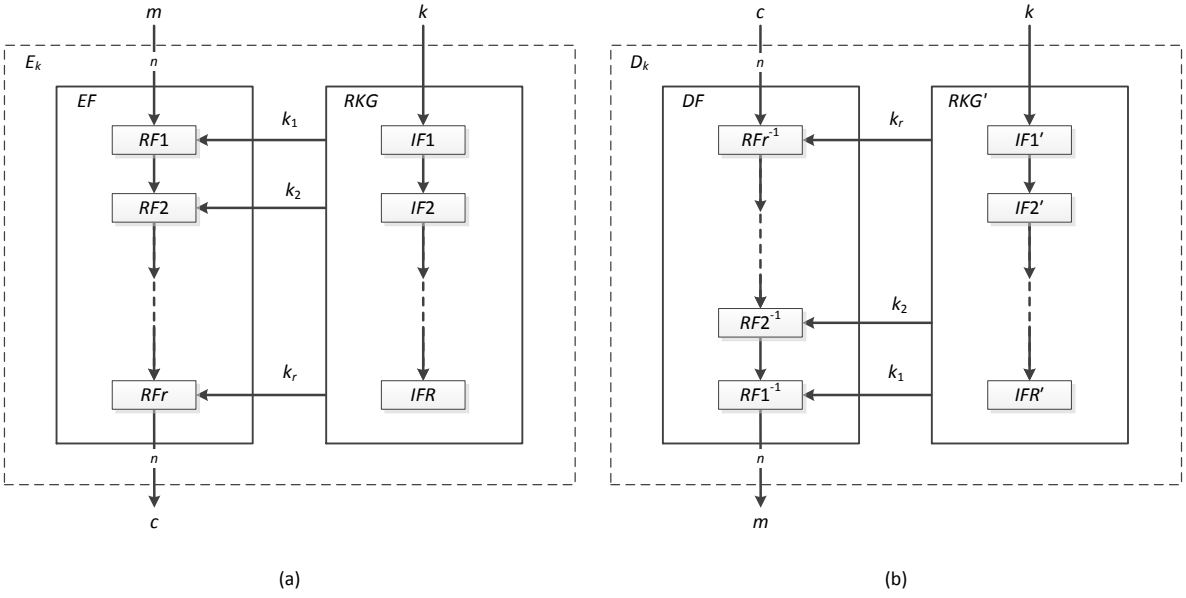
3.1. Ogólna struktura szyfru blokowego

Jak wspomniano we wstępie rozprawy, celem stosowania algorytmów generowania kluczy rundowych w szyfrach blokowych jest wytworzenie, ze stosunkowo krótkiego klucza głównego, zestawu kluczy rundowych. To właśnie klucze rundowe są wykorzystywane, podczas szyfrowania oraz deszyfrowania, w poszczególnych rundach. Algorytmy generowania kluczy rundowych, podobnie jak sam szyfr, wykorzystują proste operacje liniowe lub nieliniowe. Zależnie od zastosowanych operacji, zarówno czas generowania jak i jakość wygenerowanych kluczy rundowych są różne.

Na rysunku (rys. 3.1) przedstawiono wewnętrzną strukturę funkcji szyfrującej E_k i deszyfrującej D_k. Przyjęto następujące oznaczenia:

- c – blok szyfrogramu (ang. *cryptogram*), inna nazwa: blok tekstu zaszyfrowanego (ang. *ciphertext*),

- *DF* – funkcja deszyfrująca (ang. *decryption function*) zależna od kluczy rundowych,
- D_k – funkcja deszyfrująca (ang. *decryption function*) z ustaloną wartością klucza *k*,
- *EF* – funkcja szyfrująca (ang. *encryption function*) zależna od kluczy rundowych,
- E_k – funkcja szyfrująca (ang. *encryption function*) z ustaloną wartością klucza *k*,
- *IF* – funkcja iteracji (ang. *iteration function*),
- *IF′* – wariant *IF*,
- *k* – klucz tajny (ang. *secret key*), inne nazwy: klucz główny (ang. *master key*), klucz szyfru (ang. *cipher key*),
- k_i – klucz rundowy (*ang. round key*),
- *m* – blok wiadomości (ang. *message*), inna nazwa: blok tekstu jawnego (ang. *plaintext*),
- *n* – rozmiar bloku w bitach,
- *R* – liczba iteracji *RKG*,
- *RF* – funkcja rundy (ang. *round function*),
- RF^{-1} – funkcja odwrotna do *RF*,
- *RKG* – funkcja generowania kluczy rundowych (ang. *round keys generation*),
- *RKG′* – wariant *RKG*,
- *r* – liczba rund szyfru.



Rys. 3.1. Wewnętrzna struktura funkcji: (a) szyfrującej E_k, (b) deszyfrującej D_k

Funkcje E_k i D_k są wzajemnie odwrotne. Podczas szyfrowania mamy:

$$c = E_k(m), \tag{3.1}$$

a podczas deszyfrowania zachodzi:

$$D_k(c) = E_k^{-1}\big(E_k(m)\big) = m. \tag{3.2}$$

Przy założeniu stałej wartości klucza *k*, podczas szyfrowania i deszyfrowania wszystkich bloków wiadomości, również stałą wartość mają klucze rundowe k_i, dla i = 1, 2, …, *r*. Wówczas funkcja szyfrująca *EF* może być zapisana w postaci:

$$EF = RFr_{kr} \circ \ldots \circ RF2_{k2} \circ RF1_{k1}, \tag{3.3}$$

gdzie symbol ∘ oznacza złożenie funkcji. Analogicznie funkcję deszyfrującą *DF* zapisujemy w postaci:

$$DF = RF1_{k1}^{-1} \circ RF2_{k2}^{-1} \circ \ldots \circ RFr_{kr}^{-1}. \tag{3.4}$$

W funkcji DF odwrotna jest kolejność funkcji rundowych i stosowane są funkcje odwrotne.

Reprezentatywnym przykładem szyfru SPN jest szyfr PRESENT (punkt 3.3.4). W szyfrach SPN zazwyczaj *RF*1 = *RF*2 = … = *RFr* = *RF* i wówczas:

$$EF = RF_{kr} \circ \ldots \circ RF_{k2} \circ RF_{k1}, \tag{3.5}$$

oraz

$$DF = RF_{k1}^{-1} \circ RF_{k2}^{-1} \circ \ldots \circ RF_{kr}^{-1}. \tag{3.6}$$

W szyfrze SPN dodatkowo inwolucyjnym, gdzie $RF_{ki}^{-1} = RF_{ki}$ dla i = 1, 2, …, *r*, zachodzi:

$$EF = RF_{kr} \circ \ldots \circ RF_{k2} \circ RF_{k1}, \tag{3.7}$$

oraz

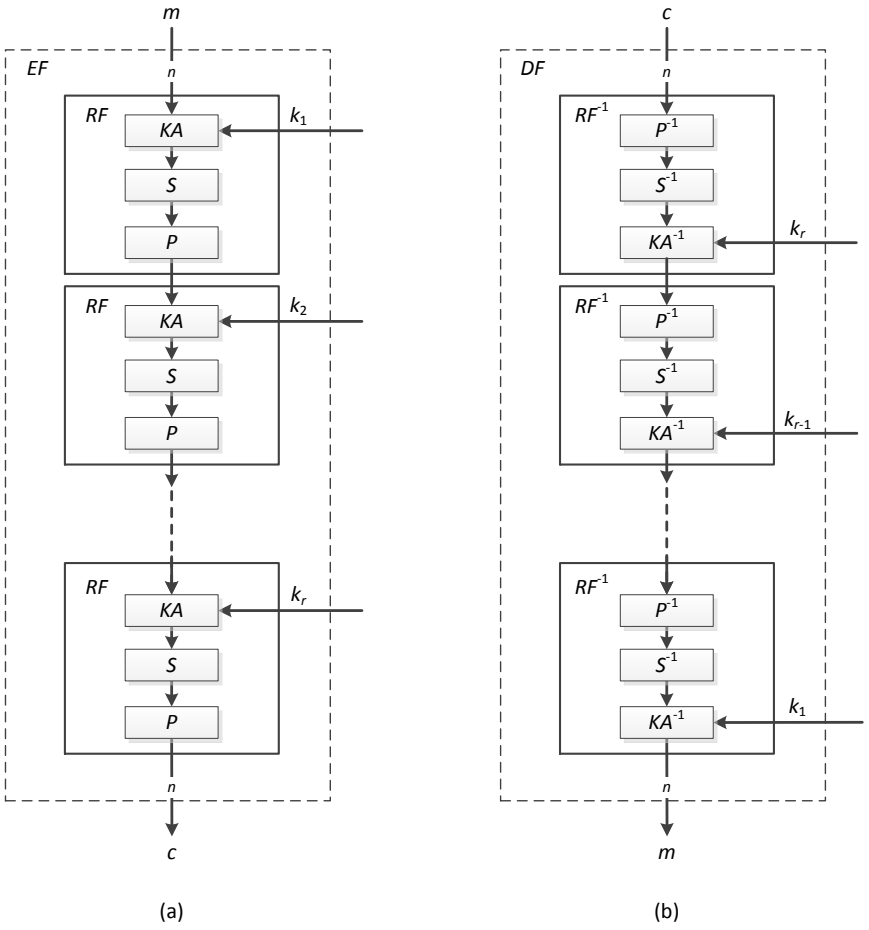
$$DF = RF_{k1} \circ RF_{k2} \circ \ldots \circ RF_{kr}, \tag{3.8}$$

a zatem do deszyfrowania może być wykorzystana struktura funkcji szyfrującej *EF* z odwróceniem kolejności kluczy rundowych. Inwolucyjne szyfry SPN należą do wyjątków. Przykładem takiego szyfru jest szyfr PP-1 (punkt 3.3.5). W typowym szyfrze SPN do deszyfrowania konieczne są funkcje odwrotne i odwrócenie kolejności kluczy rundowych, jak to wynika z formuł (3.5) oraz (3.6).

Na rysunku (rys. 3.2) przedstawiono podstawową strukturę funkcji szyfrującej i deszyfrującej w szyfrze SPN. Funkcja rundy *RF* składa się z trzech warstw:

- *KA* – warstwa dodania klucza (ang. *key addition layer*),
- *S* – warstwa podstawień (ang. *substitution layer*),

- P – warstwa dyfuzji (ang. *diffusion layer*), często realizowana w postaci permutacji bitowej i wówczas nazywana warstwą permutacji (ang. *permutation layer*).



Rys. 3.2. Szyfr SPN – podstawowa struktura funkcji: (a) szyfrującej, (b) deszyfrującej

Określenie SPN (ang. *Substitution Permutation Network*), dla typu szyfru, wynika z konstrukcji funkcji rundy, zawierającej warstwy S i P.

W warstwie dodania klucza, KA, w najprostszym i częstym przypadku, klucz rundowy k_i dodawany jest do bloku danych x_i za pomocą operacji sumy wyłączającej (XOR). Przy założeniu $|k_i| = |x_i| = n$, zachodzi:

$$u_i = KA(x_i, k_i) = x_i \oplus k_i: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n, \tag{3.9}$$

gdzie $i = 1, 2, \ldots, r$.

Warstwa podstawień S konstruowana jest typowo jako równoległe połączenie bloków podstawień – S-bloków. S-blok S_1 o s bitach wejściowych i t bitach wyjściowych realizuje funkcję (podstawienie):

$$v = S_1(u): \{0,1\}^s \rightarrow \{0,1\}^t. \tag{3.10}$$

Dla $s = t$, różnowartościowa funkcja S_1 nazywana jest permutacją zbioru $\{0,1\}^s$ lub krótko permutacją. Niech $s = t$ i niech $s|n$. Przy założeniu $|u_i| = n$, mamy:

$$v_i = v_{i,1}||v_{i,2}||\ldots||v_{i,n/s} = S(u_i = u_{i,1}||u_{i,2}||\ldots||u_{i,n/s}) = S_1(u_{i,1})||S_1(u_{i,2})||\ldots||S_1(u_{i,n/s}), \tag{3.11}$$

gdzie $i = 1, 2, \ldots, r$.

Warstwa dyfuzji P realizowana jest często jako warstwa permutacji, albo inaczej P-blok. P-blok P o n bitach wejściowych i n bitach wyjściowych realizuje funkcję (transpozycję):

$$y_i = P(v_i)\colon \{0,1\}^n \rightarrow \{0,1\}^n, \tag{3.12}$$

taką, że jeśli $y_i = (y_{i,1}, y_{i,2}, \ldots, y_{i,n}) = P(v_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,n}))$, to

$$y_{i,T(j)} = v_{i,j} \text{ dla } j = 1, 2, \ldots, n, \tag{3.13}$$

gdzie $T\colon \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, n\}$ jest permutacją zbioru $\{1, 2, \ldots, n\}$, a $i = 1, 2, \ldots, r$. Funkcja P zwykle jest utożsamiana z funkcją T i nazywana transpozycją, permutacją bitową lub krótko permutacją.

Klasycznym przykładem szyfru Feistela jest szyfr DES (punkt 3.2.1). W szyfrach Feistela zazwyczaj: $RF1 = RF2 = \ldots = RFr{-}1 = RF$, a ostatnia funkcja rundowa $RFr = OT$, gdzie funkcja OT nazywana transformacją wyjściową (ang. *output transformation*), jest różna od pozostałych funkcji rundowych. Funkcja $RF = RFX \circ RFY$, a funkcja $OT = RFY$, gdzie RFX to charakterystyczna dla szyfru Feistela zamiana lewego i prawego słowa. Podczas szyfrowania mamy zatem:

$$EF = RFY_{kr} \circ (RFX \circ RFY_{kr-1}) \circ \ldots \circ (RFX \circ RFY_{k2}) \circ (RFX \circ RFY_{k1}), \tag{3.14}$$

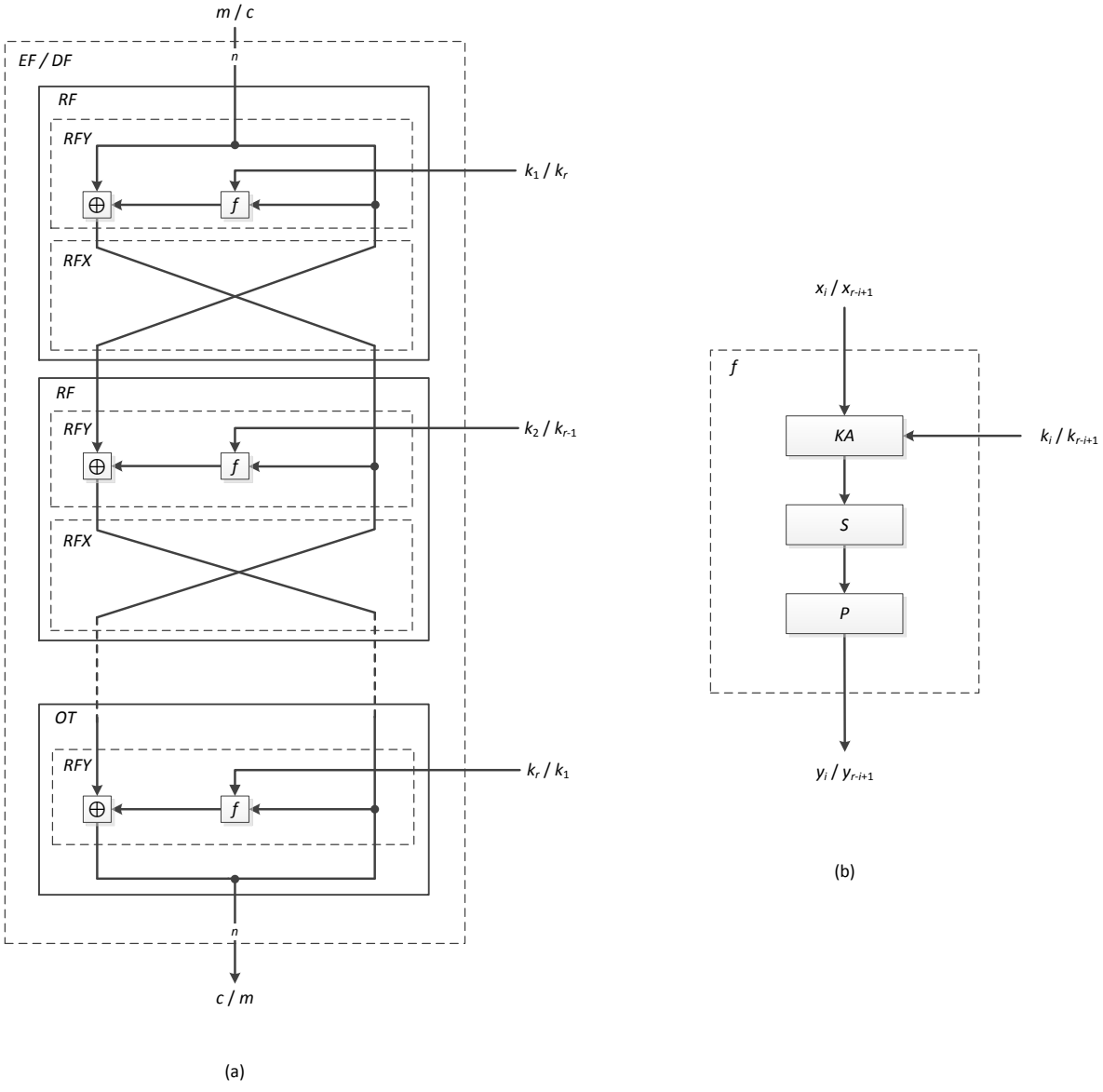
a podczas deszyfrowania zachodzi:

$$DF = (RFY_{k1}^{-1} \circ RFX^{-1}) \circ (RFY_{k2}^{-1} \circ RFX^{-1}) \circ \ldots \circ (RFY_{kr-1}^{-1} \circ RFX^{-1}) \circ RFY_{kr}^{-1}. \tag{3.15}$$

Funkcja RFX jest inwolucją, a funkcja RFY zazwyczaj jest inwolucją. Przykładem szyfru, w którym funkcja RFY nie jest inwolucją jest szyfr LOKI97 (punkt 3.2.2). Przy założeniu, że RFX i RFY są inwolucją oraz przy zmianie nawiasowania w formule (3.15), otrzymujemy:

$$DF = RFY_{k1} \circ (RFX \circ RFY_{k2}) \circ (RFX \circ \ldots \circ RFY_{kr-1}) \circ (RFX \circ RFY_{kr}). \tag{3.16}$$

Z porównania formuł (3.14) i (3.16) wynika, że do deszyfrowania można wykorzystać strukturę funkcji szyfrującej EF z odwróceniem kolejności kluczy rundowych. Taka sytuacja jest typowa dla szyfrów Feistela.



Rys. 3.3. Szyfr Feistela – podstawowa struktura funkcji: (a) szyfrującej/deszyfrującej, (b) bazowej *f*

Na rysunku (rys. 3.3) przedstawiono podstawową strukturę funkcji szyfrującej/deszyfrującej oraz funkcji bazowej f w szyfrze Feistela. Jak już wspomniano, funkcja *RFX* jest inwolucją. Funkcja *RFY* typowo też jest inwolucją, niezależnie od funkcji bazowej *f*, która nawet nie musi być odwracalna, a jeśli jest odwracalna to funkcja odwrotna f^{-1} nie jest wykorzystywana. W konstrukcji funkcji bazowej f występują warstwy: *KA*, *S*, *P*.

W algorytmie generowania kluczy rundowych liczba R iteracji może być równa liczbie r rund szyfru, np. szyfr DES (punkt 3.2.1), lub różna, np. szyfr LOKI97 (punkt 3.2.2), gdzie $R = 3r$ i na każdą rundę przypadają trzy klucze rundowe. Wówczas, na rysunku (rys. 3.1), klucz rundowy k_i ($i = 1, 2, \ldots, r$) może być traktowany jako ciąg trzech podkluczy: $k_i = (k_{i,1}, k_{i,2}, k_{i,3})$.

Funkcja *RKG* generowania kluczy rundowych nie musi być odwracalna, a nawet jeśli jest odwracalna to funkcja odwrotna RKG^{-1} nie jest podczas deszyfrowania potrzebna.

W implementacji programowej $RKG' = RKG$ i funkcja *DF* wywoływana jest z odwróconą kolejnością kluczy rundowych, tj. $k_r, k_{r-1}, \ldots, k_1$ zamiast $k_1, k_2, \ldots, k_r$.

W przypadku implementacji sprzętowej $RKG' = XRK \circ RKG$, gdzie funkcja *XRK* odwraca kolejność kluczy rundowych.

Funkcje iteracji IF_i ($i = 1, 2, \ldots, R$) są zwykle konstrukcyjnie identyczne, ale indywidualizowane są przez różne dla każdej iteracji stałe lub klucze pomocnicze.

W rozprawie, wybrane szyfry Feistela przedstawiono w podrozdziale 3.2, wybrane szyfry SPN – w podrozdziale 3.3, a w podrozdziale 3.4 zawarto przykłady szyfrów o innej konstrukcji, głównie ze względu na odmienną realizację warstwy podstawień *S*.

Algorytm 3.1. Algorytm generowania kluczy rundowych ($R = r$)

WEJŚCIE: klucz główny k o długości n bitów

WYJŚCIE: klucze rundowe $k_1, k_2, \ldots, k_r$ o długości n bitów

PROCEDURA:

1. Jako X_1 podstaw k, tj. $X_1 = k$.
2. Dla $i = 1, 2, \ldots, R = r$ powtarzaj:
   - oblicz $Y_i = IF_i(X_i)$,
   - jako k_i podstaw Y_i, tj. $k_i = Y_i$,
   - jako X_{i+1} podstaw Y_i, tj. $X_{i+1} = Y_i$.

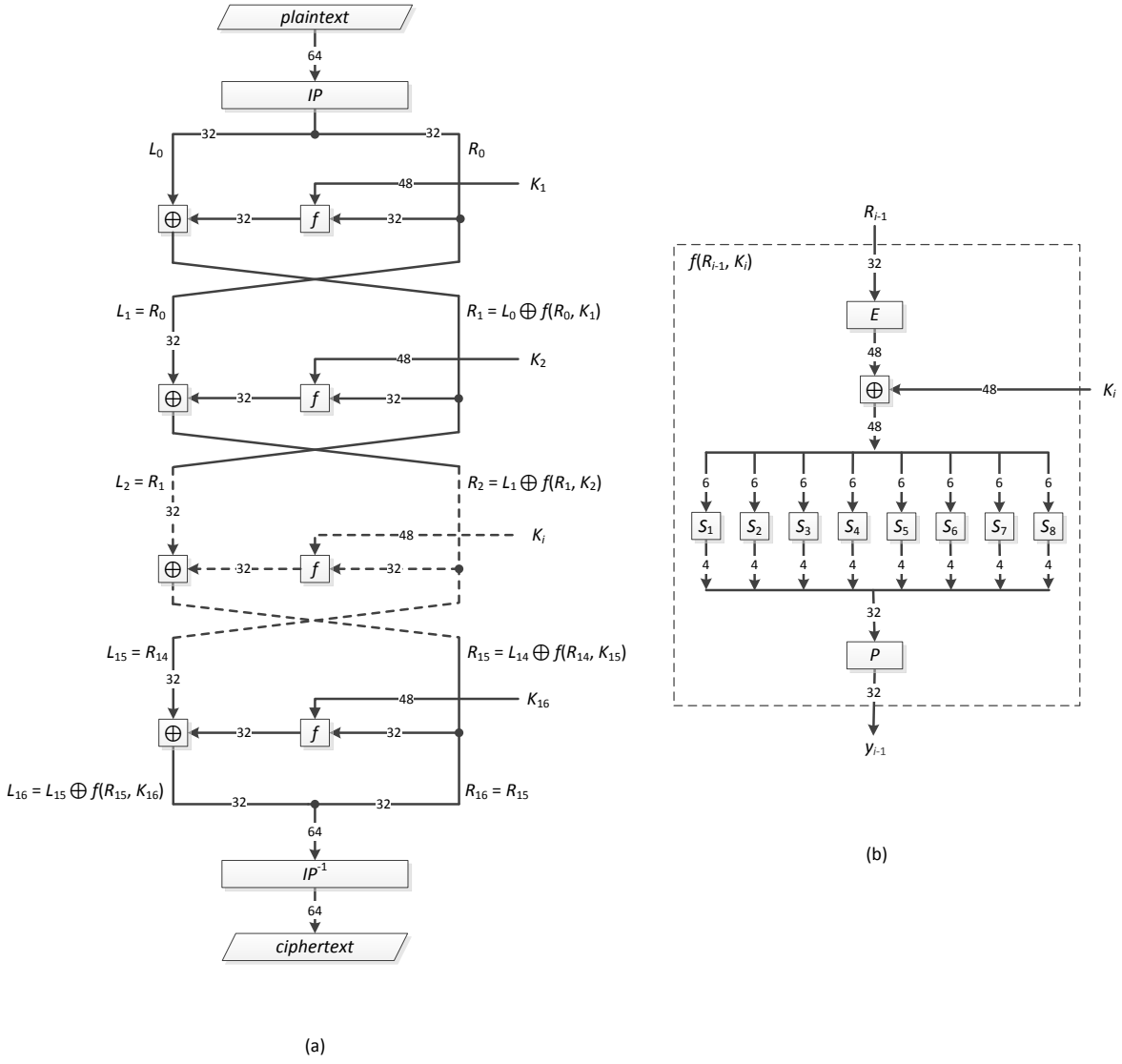
W algorytmie generowania kluczy rundowych, przedstawionym jako algorytm 3.1, przyjęto założenie, że liczba R iteracji jest równa liczbie r rund szyfru, a klucz rundowy k_i ($i = 1, 2, \ldots, r$) jest równy wyjściu funkcji iteracji IF_i.

3.2. Szyfry blokowe o konstrukcji sieci Feistela

3.2.1. DES (1975)

Szyfr blokowy DES (ang. *Data Encryption Standard*) [49] został opracowany przez firmę IBM i stanowił standard federalny USA w latach 1976 – 2005. Szyfr DES jest szyfrem o budowie sieci Feistela i przetwarza 64-bitowy blok tekstu jawnego, z wykorzystaniem 64-bitowego klucza głównego k (co ósmy bit jest bitem parzystości, co w rezultacie daje 56 bitów użytkowych klucza), w 64-bitowy blok tekstu zaszyfrowanego. Przekształcenie odbywa

się w 16 rundach i w rundzie i wykorzystywany jest jeden 48-bitowy klucz rundowy K_i (i = 1, 2, …, 16).



Rys. 3.4. DES – struktura funkcji: (a) szyfrującej/deszyfrującej, (b) bazowej *f*

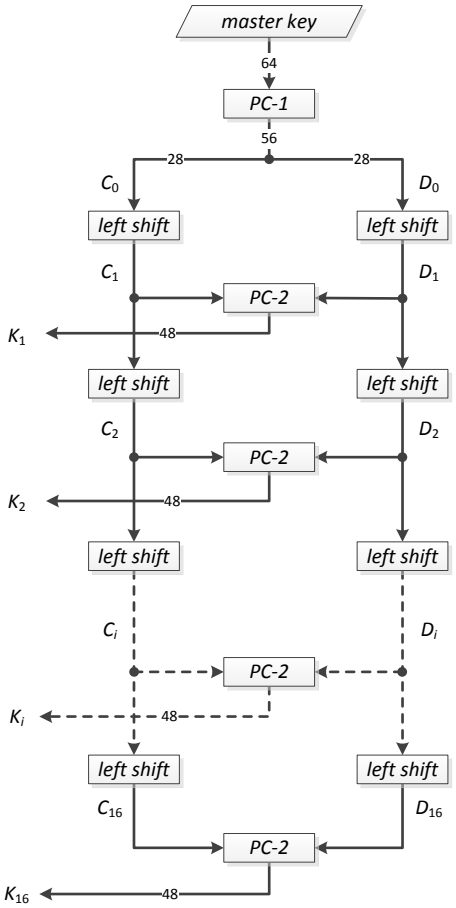
Strukturę funkcji szyfrującej/deszyfrującej szyfru DES przedstawiono na rysunku (rys. 3.4 (a)). Podczas deszyfrowania wykorzystywana jest funkcja szyfrująca z odwrotną kolejnością kluczy rundowych.

W funkcji szyfrującej/deszyfrującej wykonywane są następujące operacje:

- *IP* – permutacja początkowa (ang. *initial permutation*),
- $\oplus$ – suma wyłączająca (XOR) słów 32-bitowych,
- *f* – funkcja bazowa, w skład której wchodzi:
    - *E* – funkcja rozszerzenia 32-bitowego słowa do słowa 48-bitowego,

- $\oplus$ – suma wyłączająca (XOR) słowa 48-bitowego z 48-bitowym kluczem rundowym,
- S_1, S_2, …, S_8 – osiem różnych S-bloków o rozmiarze 6 × 4 bity (podstawienia),
- P – permutacja bitowa,

- IP^{-1} – odwrotna permutacja początkowa (ang. *inverse initial permutation*) lub inaczej permutacja końcowa (ang. *final permutation*).



Rys. 3.5. DES – struktura funkcji generowania kluczy rundowych

W algorytmie generowania kluczy rundowych (algorytm 3.2) i w funkcji przedstawionej na rysunku (rys. 3.5), wykonywane są operacje:

- *PC-1* – permutacja dzieląca 64-bitowe słowo, z pominięciem bitów parzystości, na dwa 28-bitowe słowa (tab. 3.1),
- *left shift* – cykliczne przesunięcie (rotacja) w lewo słowa 28-bitowego o 1 lub 2 bity (tab. 3.2),

- *PC-2* – permutacja łącząca dwa słowa 28-bitowe, z pominięciem 8 bitów, w słowo 48-bitowe (tab. 3.3).

Algorytm 3.2. Algorytm generowania kluczy rundowych szyfru DES

WEJŚCIE: klucz główny k o długości 64 bity

WYJŚCIE: klucze rundowe $K_1, K_2, \ldots, K_{16}$ o długości 48 bitów

PROCEDURA:

1. Oblicz podklucze C_0 i D_0:
   - $C_0 = PC\text{-}1_C(k)$, $D_0 = PC\text{-}1_D(k)$.
2. Dla $i = 1, 2, \ldots, 16$ powtarzaj:
   - $C_i = left\ shift(C_{i-1}, b_i)$, $D_i = left\ shift(D_{i-1}, b_i)$, gdzie b_i to parametr rotacji (tab. 3.2),
   - $K_i = PC\text{-}2(C_i || D_i)$.

Wynikiem działania algorytmu generowania kluczy rundowych jest szesnaście 48-bitowych kluczy rundowych, uzyskanych z klucza głównego o długości 64 bity.

Tab. 3.1. DES – tablica permutacji – operacja *PC-1*

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

Tablica permutacji PC-1 (tab. 3.1) określa, które bity klucza głównego przypisywane są podkluczom C_0 oraz D_0. Pominięte są bity parzystości 8, 16, 24, 32, 40, 48, 56, 64. Do podklucza C_0 przypisywany jest 57, 49, …, 44, 36 bit klucza głównego, natomiast do podklucza D_0 przypisywany jest 63, 55, …, 12, 4 bit klucza głównego. Bity 57 oraz 63 klucza głównego stanowią najbardziej znaczące bity podkluczy C_0, D_0, a bity 36 oraz 4 stanowią najmniej znaczące bity podkluczy C_0, D_0.

Tab. 3.2. DES – tablica rotacji – operacja *left shift*

numer iteracji *i*	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
parametr rotacji b_i	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Tablica rotacji (tab. 3.2) wskazuje, o ile bitów ma być wykonana rotacja w lewo, podkluczy C_{i-1} oraz D_{i-1}, w iteracji i (i = 1, 2, …, 16).

Tab. 3.3. DES – tablica permutacji – operacja *PC-2*

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

Tablica permutacji *PC-2* (tab. 3.3) określa, które bity klucza powstałego z konkatenacji C_i oraz D_i (i = 1, 2, …, 16) przypisywane są do klucza rundowego K_i. Do klucza K_i przypisywany jest 14, 17, …, 29, 32 bit klucza $C_i||D_i$.

Tab. 3.4. DES – zestawienie cech szyfru

rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego |*k*|	liczba kluczy rundowych *nrk*	długość klucza rundowego |*rk*|	długość próbki kluczy rundowych |*rkp*|
64b	16	64b	16	48b	768b

W tabeli (tab. 3.4) przedstawiono zestawienie podstawowych cech szyfru DES, gdzie:

- n – rozmiar bloku w bitach,
- r – liczba rund,
- $|k|$ – długość klucza głównego w bitach (z bitami parzystości),
- nrk – liczba kluczy rundowych,
- $|rk|$ – długość klucza rundowego w bitach,
- $|rkp|$ – długość próbki kluczy rundowych w bitach ($|rkp| = nrk \cdot |rk|$).

Parametr $|rkp|$ wykorzystywany jest w metodzie A, omówionej w punkcie 4.3.2.

Tab. 3.5. DES – lista operacji

rodzaj	operacja	funkcja szyfrująca/deszyfrująca	algorytm generowania kluczy rundowych
permutacja początkowa	IP, IP^{-1}	tak	nie
XOR	$\oplus$	tak	nie
rozszerzenie	E	tak	nie
podstawienie	S_1, S_2, …, S_8	tak	nie
permutacja	P	tak	nie
permutacja z wyborem	*PC-1*, *PC-2*	nie	tak
rotacja w lewo	*left shift*	nie	tak

W tabeli (tab. 3.5) przedstawiono listę operacji funkcji szyfrującej/deszyfrującej i algorytmu generowania kluczy rundowych. Operacje S_1, S_2, …, S_8 są operacjami nieliniowymi, a pozostałe operacje są liniowe. Na przykład – permutacja P jest liniowa ponieważ:

$$P(x \oplus y) = P(x) \oplus P(y), \text{dla } x, y = 0, 1, \dots, 2^l - 1, \tag{3.17}$$

gdzie l to długość słowa w bitach.

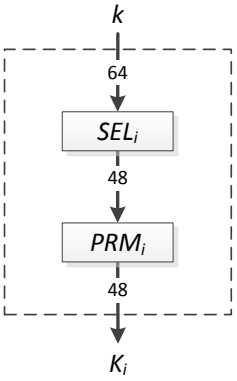
Wniosek 3.1.

W algorytmie generowania kluczy rundowych szyfru DES wykorzystywane są wyłącznie operacje liniowe. Zwiększenie liczby iteracji tego algorytmu nie jest możliwe, ze względu na zdefiniowanie parametru rotacji dla numeru iteracji w zakresie od jeden do szesnaście (tab. 3.2) (metoda B).

■

Zdefiniujmy dla iteracji i (i = 1, 2, …, 16), sumaryczny parametr rotacji, $\beta_i = b_1+b_2+\ldots+b_i$, gdzie b_j (j = 1, 2, …, i) to parametr rotacji z tabeli (tab. 3.2). Klucz rundowy K_i może być określony formułą:

$$K_i = PC\text{-}2(left\ shift(PC\text{-}1_C(k), \beta_i) || left\ shift(PC\text{-}1_D(k), \beta_i)). \tag{3.18}$$



Rys. 3.6. DES – schemat zastępczy funkcji generowania klucza rundowego K_i (i = 1, 2, …, 16)

W schemacie zastępczym funkcji generowania klucza rundowego K_i, przedstawionym na rysunku (rys. 3.6), wykonywane są operacje:

- SEL_i – funkcja wyboru 48 bitów z 64-bitowego klucza k, z pominięciem 8 bitów parzystości w *PC-1*, 4 bitów C_i oraz 4 bitów D_i w *PC-2*,
- PRM_i – permutacja uporządkowania wybranych bitów na właściwych pozycjach klucza K_i, bez istotnych regularności, zgodnie ze złożeniem permutacji *PC-1*, *left shift* i *PC-2*.

Wniosek 3.2.

W szyfrze DES, klucz rundowy K_i (i = 1, 2, …, 16) zależy od 48 bitów 64-bitowego klucza głównego k, przy czym każdy bit klucza K_i jest pewnym bitem klucza k, a tym samym jest zależny od jednego bitu klucza k.

■

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru DES, przedstawiono w postaci charakterystyki (charakterystyka 3.1).

Charakterystyka 3.1. Charakterystyka algorytmu generowania kluczy rundowych szyfru DES

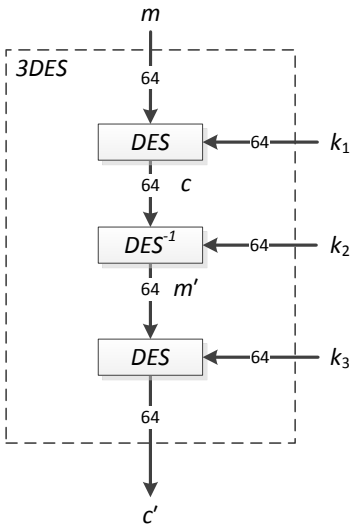
1. Długość klucza głównego: $|\boldsymbol{k}|$ **= 64b**.
2. Długość klucza rundowego: $|\boldsymbol{K_i}|$ **= 48b (*i* = 1, 2, …, 16)**.
3. Liczba iteracji: $\boldsymbol{R}$ **= 16**.
4. Zależność K_i od bitów k:
   - **48b / 64b (75%)**.
5. Zależność bitu K_i od bitów k:
   - **1b / 64b (1.56%)**.
6. Zależność grupy bitów K_i od grupy bitów k: **NIE**.
7. Operacje liniowe: ***PC-1, left shift, PC-2***.
8. Operacje nieliniowe: **BRAK**.
9. Możliwości zwiększenia liczby iteracji: **NIE**.
10. Indywidualizacja iteracji: **przez sumaryczny parametr rotacji β_i.**
11. Teoretyczna ocena jakości algorytmu: **1.56%.**

W punkcie 4 charakterystyki podano, że (cały) klucz rundowy K_i zależy od 48 z 64 bitów klucza głównego k, a w punkcie 5 – że (pojedynczy) bit klucza K_i zależy od 1 z 64 bitów klucza k. Przez grupę bitów w punkcie 6 rozumiemy ciąg sąsiadujących bitów, o długości większej od 1 i mniejszej od długości klucza. W punkcie 6 charakterystyki podano długość grupy bitów klucza K_i zależnej od grupy bitów, o tej samej długości, klucza k.

Jako kryterium teoretycznej oceny jakości algorytmu przyjmujemy zależność bitu klucza rundowego od bitów klucza głównego (punkt 5). Każdy bit klucza rundowego powinien być zależny od wszystkich bitów klucza głównego.

3.2.2. 3DES (TDEA) (1995)

Szyfr blokowy 3DES[3] (ang. *Triple Data Encryption Standard*) [10] został opublikowany w roku 1995, a od roku 1999 stanowi standard kryptograficzny FIPS 46-3 [49]. Szyfr 3DES przetwarza 64-bitowe bloki tekstu jawnego *m* w 64-bitowe bloki tekstu zaszyfrowanego *c′*, z wykorzystaniem 192-bitowego klucza głównego *k* będącego konkatenacją trzech 64-bitowych podkluczy głównych k_1, k_2, k_3.



Rys. 3.7. 3DES –przekształcenie bloku *m* wiadomości w blok *c′* szyfrogramu

Zasada działania szyfru 3DES, przedstawiona na rysunku (rys. 3.7), polega na potrójnym wykorzystaniu standardowego szyfru DES. Przekształcenie odbywa się poprzez szyfrowanie bloku wiadomości *m*, szyfrem DES z kluczem k_1 (*DES*), następnie deszyfrowanie bloku szyfrogramu *c* szyfrem DES z kluczem k_2 (DES^{-1}) i ponowne szyfrowanie, uzyskanego bloku wiadomości *m′* ($m \neq m'$ jeśli $k_1 \neq k_2$), szyfrem DES z kluczem k_3 (*DES*). Wynikiem tych operacji jest blok szyfrogramu *c′* taki, że $c' \neq c$ jeśli $k_2 \neq k_3$. Proces przekształcenia bloku wiadomości *m* w blok szyfrogramu *c′* można zapisać następująco: $c' = DES(DES^{-1}(DES(m, k_1), k_2), k_3)$, a proces odtwarzania bloku wiadomości *m* z bloku szyfrogramu *c′* w następujący sposób: $m = DES^{-1}(DES(DES^{-1}(c', k_3), k_2), k_1)$.

Ponieważ szyfr 3DES wykorzystuje trzykrotnie szyfr DES, to wszystkie operacje w funkcji szyfrującej/deszyfrującej, a także struktura funkcji oraz algorytmu generowania kluczy rundowych są takie same lub analogiczne do opisanych w punkcie 3.2.1.

[3] Nazwa stosowana w standardzie FIPS oraz NIST to TDEA (ang. *Triple Data Encryption Algorithm*).

W roku 2016 opublikowano praktyczny atak na szyfr 3DES [14] i zgodnie z rekomendacją NIST [9], od roku 2017 szyfr ten nie powinien być stosowany w nowych rozwiązaniach, a do roku 2023[4] powinien zostać całkowicie wycofany z użytkowania, również w istniejących rozwiązaniach.

Tab. 3.6. 3DES – zestawienie cech szyfru

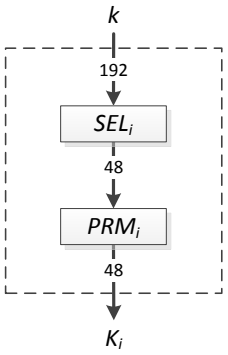
rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego |*k*|	liczba kluczy rundowych *nrk*	długość klucza rundowego |*rk*|	długość próbki kluczy rundowych |*rkp*|
64b	3 ·16 = 48	3 · 64b = 192b	3 ·16 = 48	48b	3 ·768b = 2304b

Zestawienie podstawowych cech szyfru 3DES zawarto w tabeli (tab. 3.6).

Wniosek 3.3.

W algorytmie generowania kluczy rundowych szyfru 3DES wykorzystywane są wyłącznie operacje liniowe (*PC-1*, *left shift*, *PC-2*). Zwiększenie liczby iteracji tego algorytmu nie jest możliwe, ze względu na ograniczenie definicji parametru rotacji b_i (tab. 3.2) (metoda B).

■



Rys. 3.8. 3DES – schemat zastępczy funkcji generowania klucza rundowego K_i (i = 1, 2, …, 48)

W schemacie zastępczym funkcji generowania klucza rundowego K_i, przedstawionym na rysunku (rys. 3.8), wykonywane są operacje:

- SEL_i – funkcja wyboru 48 bitów z klucza k o długości 192 bity,
- PRM_i – permutacja uporządkowania wybranych bitów na właściwych pozycjach klucza K_i.

[4] Pierwotnie planowaną datą całkowitego wycofania szyfru 3DES był rok 2030 [10].

Wniosek 3.4.

W szyfrze 3DES, klucz rundowy K_i (i = 1, 2, …, 48) zależy od 48 bitów 192-bitowego klucza głównego k, przy czym każdy bit klucza K_i jest pewnym bitem klucza k, a tym samym jest zależny od jednego bitu klucza k.

■

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru 3DES, przedstawiono w postaci charakterystyki (charakterystyka 3.2).

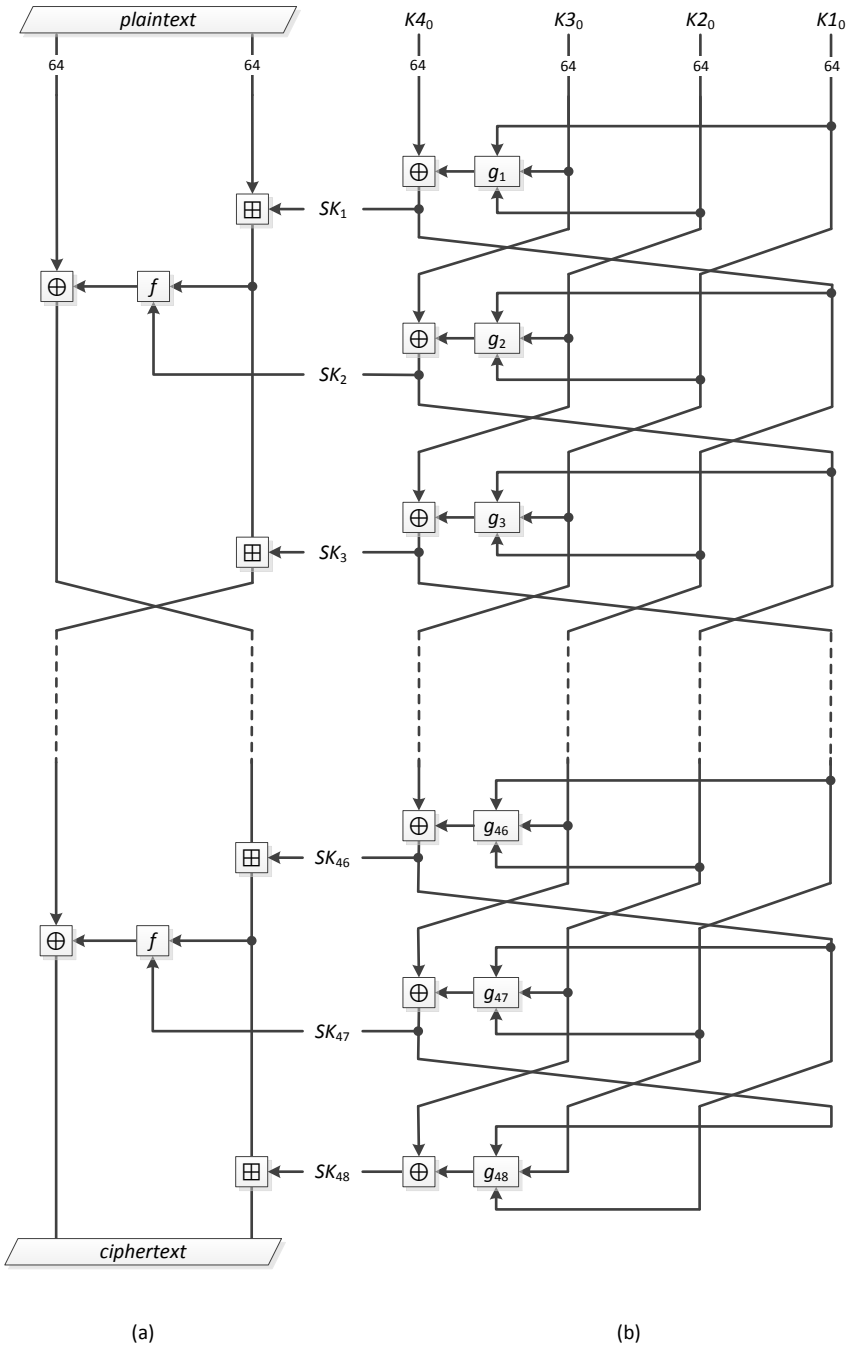
Charakterystyka 3.2. Charakterystyka algorytmu generowania kluczy rundowych szyfru 3DES

1. Długość klucza głównego: $|\boldsymbol{k}| = \mathbf{192b}$.
2. Długość klucza rundowego: $|\boldsymbol{K_i}| = \mathbf{48b}$ $\mathbf{(}\boldsymbol{i} \mathbf{= 1, 2, …, 48)}$.
3. Liczba iteracji: $\boldsymbol{R} = \mathbf{48}$.
4. Zależność K_i od bitów k:
   - **48b / 192b (25%)**.
5. Zależność bitu K_i od bitów k:
   - **1b / 192b (0.52%)**.
6. Zależność grupy bitów K_i od grupy bitów k: **NIE**.
7. Operacje liniowe: ***PC-1, left shift, PC-2***.
8. Operacje nieliniowe: **BRAK**.
9. Możliwości zwiększenia liczby iteracji: **NIE**.
10. Indywidualizacja iteracji: **przez sumaryczny parametr rotacji β_i**.
11. Teoretyczna ocena jakości algorytmu: **0.52%**.

3.2.3. LOKI97 (1997)

Szyfr blokowy LOKI97 [36], zakwalifikowany do konkursu na standard AES, o strukturze Feistela funkcji szyfrującej, przetwarza bloki danych o rozmiarze n = 128 bitów z wykorzystaniem klucza głównego k o długości $|k|$ = 128, 192 lub 256 bitów. Przekształcenie bloku tekstu jawnego w blok tekstu zaszyfrowanego odbywa się w r = 16 rundach i w rundzie i (i = 1, 2, …, r) wykorzystywane są trzy 64-bitowe klucze rundowe SK_{3i-2}, SK_{3i-1}, SK_{3i}.

Ta sama struktura funkcji szyfrującej, przedstawiona na rysunku (rys. 3.9 (a)), wykorzystywana jest podczas deszyfrowania. Klucze rundowe wówczas podawane są w odwrotnej kolejności, a klucze SK_{3i-2}, SK_{3i} zamieniane są na klucze $(-SK_{3i-2})$, $(-SK_{3i})$, co zastępuje w rundzie i zamianę obu operacji $\boxplus$ na operacje $\boxminus$.



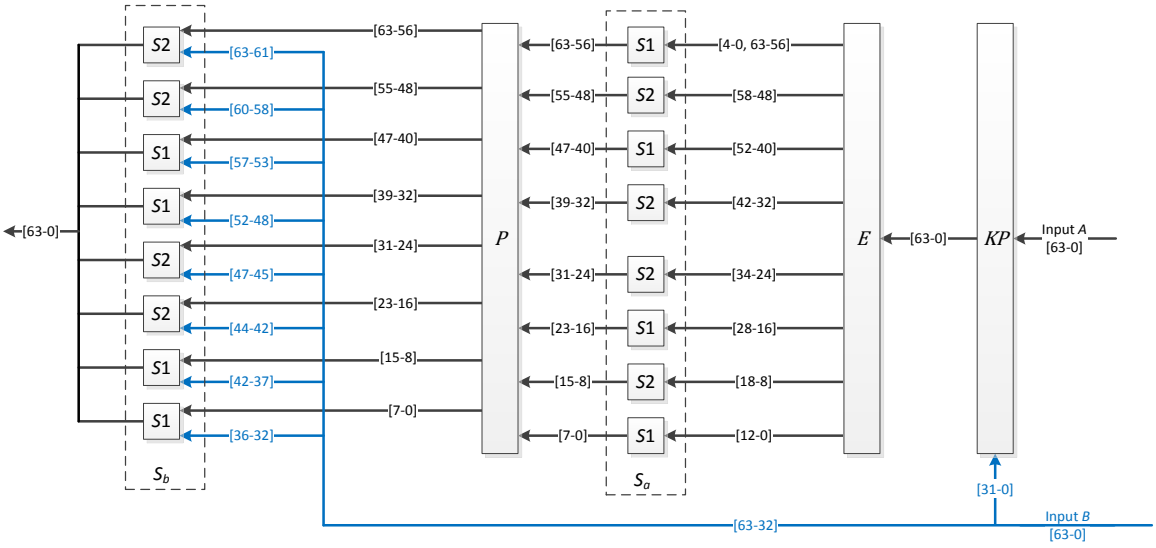
Rys. 3.9. LOKI97 – struktura funkcji: (a) szyfrującej/deszyfrującej, (b) generowania kluczy rundowych

W funkcji szyfrującej/deszyfrującej wykonywane są następujące operacje:

- $\boxplus$ – dodawanie modulo 2^{64},
- $\oplus$ – suma wyłączająca (XOR) słów 64-bitowych,
- f – nieliniowa funkcja bazowa (rys. 3.10) taka, że dla $A = Al||Ar$ i $B = Bl||Br$, gdzie $|A| = |B| = 64$ bity, $f(A, B) = S_b(P(S_a(E(KP(A, Br)))), Bl)$, złożona z funkcji:
  - KP – permutacja 64-bitowa sterowana 32-bitowym kluczem taka, że:

    $$KP(Al||Ar, Br) = [Al \wedge (\sim Br)] \vee [Ar \wedge Br]|||[Ar \wedge (\sim Br)] \vee [Al \wedge Br],$$

- E – funkcja rozszerzenia słowa 64-bitowego do słowa 96-bitowego, przez powielenie 32-bitów,
- S_a – funkcja podstawień, realizowana przez ciąg ($S1$, $S2$, $S1$, $S2$, $S2$, $S1$, $S2$, $S1$) równolegle połączonych S-bloków, gdzie $S1$ ma rozmiar 13 × 8 bitów, a $S2$ – rozmiar 11 × 8 bitów,
- P – permutacja 64-bitowa (tab. 3.7), np. bit 63 wejścia jest bitem 56 wyjścia,
- S_b – funkcja podstawień, realizowana przez ciąg ($S2$, $S2$, $S1$, $S1$, $S2$, $S2$, $S1$, $S1$) równolegle połączonych S-bloków.



Rys. 3.10. LOKI97 – struktura funkcji *f*(*A*, *B*)

Funkcja generowania kluczy rundowych w szyfrze LOKI97 (rys. 3.9 (b)), o strukturze niezrównoważonej sieci Feistela, przetwarza w $R = 48$ iteracjach 256-bitowy klucz główny k, podzielony na cztery 64-bitowe podklucze. W iteracji i ($i = 1, 2, \ldots, R$), indywidualna funkcja bazowa g_i obliczana jest na podstawie trzech podkluczy, a jej wartość, zsumowana operacją XOR z czwartym podkluczem, stanowi jednocześnie nową wartość tego podklucza i wartość 64-bitowego klucza rundowego SK_i.

W funkcji generowania kluczy rundowych i w algorytmie (algorytm 3.3) wykonywane są następujące operacje:

- $\oplus$ – suma wyłączająca (XOR) słów 64-bitowych,
- g_i – nieliniowa funkcja bazowa iteracji i taka, że:

  $$g_i\,(K1_{i-1}, K3_{i-1}, K2_{i-1}) = f\,((K1_{i-1} + K3_{i-1} + (\Delta \cdot i)), K2_{i-1}),$$

  gdzie:
  - f – funkcja bazowa rundy (rys. 3.10),

- + – dodawnie modulo 2^{64},
- · – mnożenie modulo 2^{64},
- Δ – stała 64-bitowa.

Algorytm 3.3. Algorytm generowania kluczy rundowych szyfru LOKI97

WEJŚCIE: klucz główny k o długości $|k| = 128$, 192 lub 256 bitów

WYJŚCIE: klucze rundowe SK_1, SK_2, …, SK_{48} o długości 64 bity

PROCEDURA:

1. Dla $i = 1, 2, \ldots, 48$ oblicz stałą iteracji:

   $C_i = \Delta \cdot i$, gdzie $\Delta = \lfloor(\sqrt{5} - 1) \cdot 2^{63}\rfloor = 0x9E3779B97F4A7C15$.

2. Uzupełnij klucz k do długości 256 bitów i inicjalizuj wejście pierwszej iteracji:
   - jeśli $|k| = 256$ bitów i $k = [K_a||K_b||K_c||K_d]$ to:

     $[K4_0||K3_0||K2_0||K1_0] = [K_a||K_b||K_c||K_d]$.
   - jeśli $|k| = 192$ bity i $k = [K_a||K_b||K_c]$ to:

     $[K4_0||K3_0||K2_0||K1_0] = [K_a||K_b||K_c||f(K_a,K_b)]$,
   - jeśli |k| = 128 bitów i $k = [K_a||K_b]$ to:

     $[K4_0||K3_0||K2_0||K1_0] = [K_a||K_b||f(K_b,K_a)||f(K_a,K_b)]$.

3. Dla $i = 1, 2, \ldots, 48$ powtarzaj:
   - oblicz wartość funkcji bazowej:

     $g = g_i\,(K1_{i-1}, K3_{i-1}, K2_{i-1}) = f\,(K1_{i-1} + K3_{i-1} + C_i, K2_{i-1})$.
   - oblicz wartość klucza rundowego:

     $SK_i = K4_{i-1} \oplus g$.
   - zamień słowa (klucze) 64-bitowe:

     $K4_i = K3_{i-1}$, $K3_i = K2_{i-1}$, $K2_i = K1_{i-1}$, $K1_i = SK_i$.

Należy zwrócić uwagę na fakt, że w procesie generowania pierwszego klucza rundowego SK_1 zaangażowane są wszystkie bity klucza głównego. Do wyliczenia kolejnych kluczy rundowych SK_i wymagana jest znajomość poprzedniego klucza rundowego SK_{i-1}, a ponadto trzech innych podkluczy.

Tab. 3.7. LOKI97 – tablica permutacji *P*

56	48	40	32	24	16	08	00	57	49	41	33	25	17	09	01
58	50	42	34	26	18	10	02	59	51	43	35	27	19	11	03
60	52	44	36	28	20	12	04	61	53	45	37	29	21	13	05
62	54	46	38	30	22	14	06	63	55	47	39	31	23	15	07

W tabeli (tab. 3.7) przedstawiono tablicę permutacji *P*. Bit 63 argumentu permutacji jest bitem 56 jej wartości.

Tab. 3.8. LOKI97 – zestawienie cech szyfru

rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego |*k*|	liczba kluczy rundowych *nrk*	długość klucza rundowego |*rk*|	długość próbki kluczy rundowych |*rkp*|
128b	16	128b	48	64b	3072b
		192b			
		256b			

Zestawienie podstawowych cech szyfru LOKI97 zawarto w tabeli (tab. 3.8). W tabeli (tab. 3.9) przedstawiono listę operacji funkcji szyfrującej/deszyfrującej i algorytmu generowania kluczy rundowych. Operacje $\boxplus$, *f*, *KP*, ∧, ∨, *S*1, *S*2, *g*, +, · są nieliniowe, a operacje $\oplus$, ~, *E*, *P* są liniowe (formuła (3.17)) lub afiniczne.

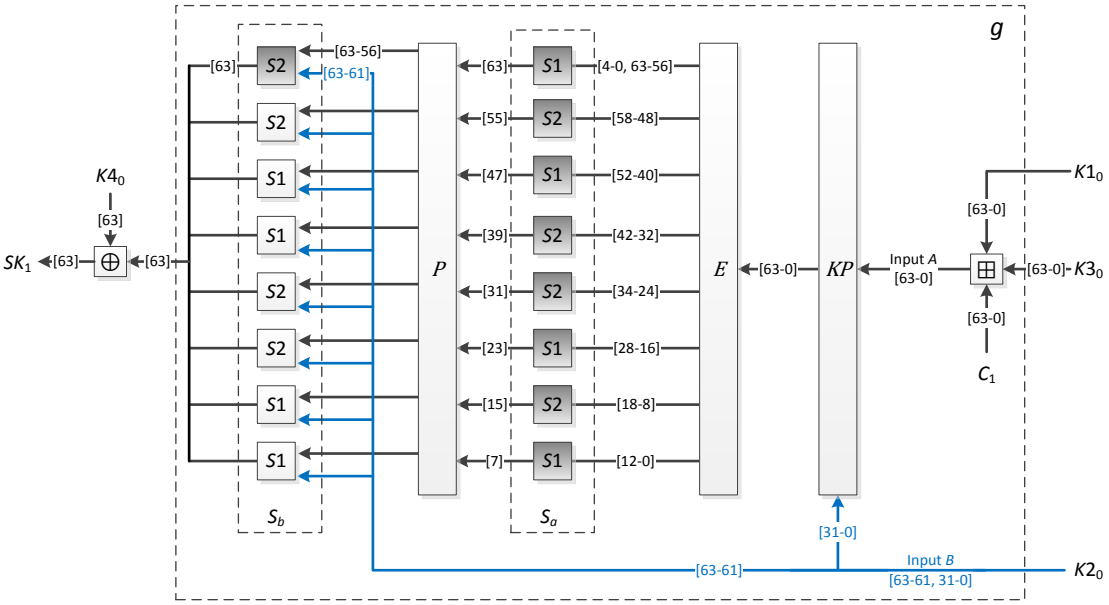
Tab. 3.9. LOKI97 – lista operacji

rodzaj	operacja	funkcja szyfrująca/deszyfrująca	algorytm generowania kluczy rundowych
dodawanie modulo 2^{64}	$\boxplus$	tak	tak
XOR	$\oplus$	tak	tak
funkcja bazowa rundy	*f*	tak	tak
permutacja z kluczem	*KP*	tak	tak
koniunkcja (AND)	∧	tak	tak
negacja (NOT)	~	tak	tak
dysjunkcja (OR)	∨	tak	tak
rozszerzenie	*E*	tak	tak
podstawienie	*S*1, *S*2	tak	tak
permutacja	*P*	tak	tak
funkcja bazowa iteracji	*g*	nie	tak
dodawanie modulo 2^{64}	+	nie	tak
mnożenie modulo 2^{64}	·	nie	tak

Wniosek 3.5.

W algorytmie generowania kluczy rundowych szyfru LOKI97 wykorzystywane są operacje liniowe i nieliniowe (w szczególności S-bloki *S*1 i *S*2 o rozmiarach 13 × 8 i 11 × 8 bitów). Przez zwiększenie liczby iteracji tego algorytmu możliwe jest zwiększenie liczby rund szyfru LOKI97 (metoda B).

■

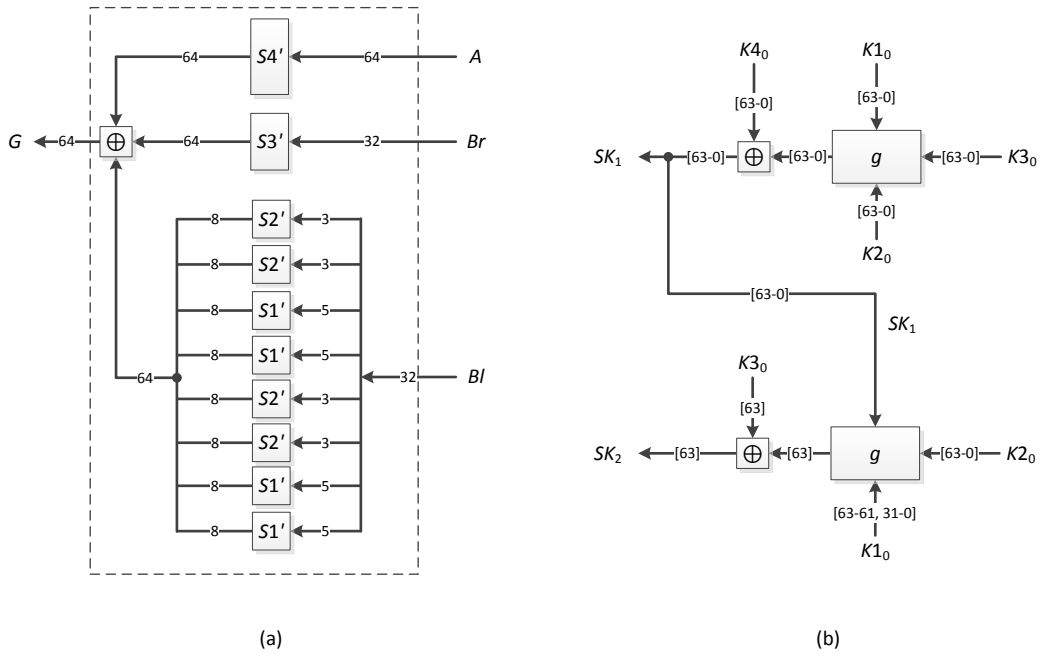


Rys. 3.11. LOKI97 – zależność bitu $SK_1[63]$ klucza rundowego od bitów klucza głównego $k = K4_0||K3_0||K2_0||K1_0$

Na rysunku (rys. 3.11) przedstawiono zależność bitu $SK_1[63]$ klucza rundowego od bitów klucza głównego $k = K4_0||K3_0||K2_0||K1_0$. Przy określeniu tej zależności uwzględniono następujące własności funkcji elementarnych:

- $\oplus$ – bit i wyjścia (wartości) funkcji XOR zależy od bitu i wszystkich wejść (argumentów),
- $S1$, $S2$ – bit i wyjścia S-bloku, $S1$ lub $S2$, zależy od wszystkich bitów wejścia,
- P – bit i wyjścia permutacji bitowej P, jako jedyny, zależy od pewnego bitu j wejścia,
- E – bit i wyjścia funkcji rozszerzenia E, jako jedyny lub jako jeden z dwóch, zależy od pewnego bitu j wejścia,
- KP – bit i wyjścia permutacji sterowanej kluczem, zależy od bitu i słów Al, Ar i Br, gdzie wejście $A = Al||Ar$, a wejście klucza jest równe Br ($B = Bl||Br$),
- $\boxplus$ – bit i wyjścia funkcji dodawania modulo 2^l zależy od bitów i, $i - 1$, …, 0 wejść, gdzie bit numer 0 jest najmniej znaczący.

Ostatecznie bit $SK_1[63]$ zależy od bitów $K4_0[63]$, $K3_0[63\text{-}0]$, $K2_0[63\text{-}61, 31\text{-}0]$ oraz $K1_0[63\text{-}0]$, a więc zależy od 164 bitów 256-bitowego klucza k, co stanowi 64.06%.



Rys. 3.12. LOKI97 – (a) schemat zastępczy funkcji *f*(*A*,*B*),
(b) zależność bitu SK_2[63] klucza rundowego od bitów klucza głównego $k = K4_0||K3_0||K2_0||K1_0$

Na rysunku (rys. 3.12 (a)) przedstawiono schemat zastępczy funkcji *f*(*A*,*B*). Każdy bit wyjścia *G* zależy od 64 bitów wejścia *A* oraz 32-bitów prawej i 3 lub 5 lewej połowy wejścia *B*. W najgorszym przypadku zależy więc od 99 spośród 128 bitów, co stanowi 77.34%.

Z rysunku (rys. 3.12 (b)) wynika, że bit SK_2[63] zależy od wszystkich bitów klucza *k*.

Wniosek 3.6.

W szyfrze LOKI97-256, klucz rundowy SK_i (i = 1, 2, …, 48) zależy od wszystkich bitów klucza głównego *k*, przy czym każdy bit klucza SK_i zależy od liczby bitów klucza *k*: 164 (i = 1, najgorszy przypadek) lub 256 (i = 2, 3, …, 48).

■

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru LOKI97, przedstawiono w postaci charakterystyki (charakterystyka 3.3).

Charakterystyka 3.3. Charakterystyka algorytmu generowania kluczy rundowych szyfru LOKI97

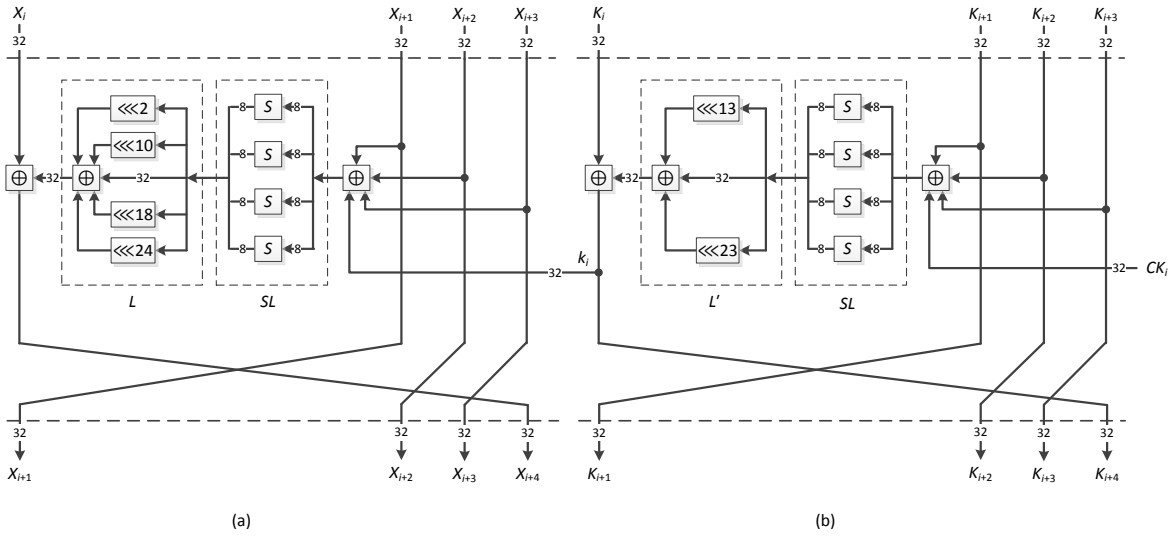
1. Długość klucza głównego: $|k|$ **= 128b, 192b, 256b**.
2. Długość klucza rundowego: $|SK_i|$ **= 64b (*i* = 1, 2, …, 48)**.
3. Liczba iteracji: R **= 48**.
4. Zależność SK_i od bitów *k* (LOKI97-256):
   - **256b / 256b (100%)**.

5. Zależność bitu SK_i od bitów k (LOKI97-256):
   - $\boldsymbol{SK_1}$ **: 164b / 256b (64.06%)**,
   - $\boldsymbol{SK_2}$**,** $\boldsymbol{SK_3}$**, …,** $\boldsymbol{SK_{48}}$ **: 256b / 256b (100%)**.
6. Zależność grupy bitów SK_i od grupy bitów k: **NIE**.
7. Operacje liniowe lub afiniczne: $\oplus$**, ~,** ***E*****,** ***P***.
8. Operacje nieliniowe: $\boxplus$**,** ***f*****,** ***KP*****, ∧, ∨,** ***S*****1 (13b × 8b),** ***S*****2 (11b × 8b),** ***g*****, +, ·**.
9. Możliwości zwiększenia liczby iteracji: **TAK**.
10. Indywidualizacja iteracji: **przez stałe** $\boldsymbol{C_i}$**.**
11. Teoretyczna ocena jakości algorytmu: **92.81%**.

W punkcie 11 charakterystyki, teoretyczna ocena jakości algorytmu obliczona jest jako, wyrażona w procentach, średnia zależność od bitów klucza głównego k, bitu pierwszych pięciu kluczy rundowych (SK_1, SK_2, …, SK_5), tj. (64.06% + 4 · 100%) / 5 = 92.81%.

3.2.4. SM4 (2006)

Szyfr blokowy SM4 (dawniej SMS4) [50] został odtajniony w 2006 roku, a od 2012 roku jest stosowany jako chiński standard zabezpieczeń lokalnych sieci bezprzewodowych WAPI (ang. *WLAN Authentication and Privacy Infrastructure*). Szyfr SM4, o konstrukcji Feistela, przetwarza bloki danych o rozmiarze 128 bitów, podzielone na cztery 32-bitowe podbloki, z wykorzystaniem klucza głównego k o długości 128 bitów. Przekształcenie odbywa się w r = 32 rundach i w rundzie i (i = 0, 1, …, 31) wykorzystywany jest jeden 32-bitowy klucz rundowy k_i.



Rys. 3.13. SM4 – struktura: (a) rundy funkcji szyfrującej/deszyfrującej, (b) iteracji funkcji generowania kluczy rundowych (*i* = 0, 1, …, 31)

Strukturę pojedynczej rundy funkcji szyfrującej/deszyfrującej oraz iteracji funkcji generowania kluczy rundowych, przedstawiono na rysunku (rys. 3.13). W funkcji szyfrującej/deszyfrującej wykonywane są następujące operacje:

- $\oplus$ – suma wyłączająca (XOR) słów 32-bitowych,
- *SL* – warstwa podstawień, realizowana z wykorzystaniem czterech równolegle połączonych S-bloków *S* o rozmiarze 8 × 8 bitów,
- $\lll b$ – cykliczne przesunięcie (rotacja) w lewo słowa 32-bitowego o *b* bitów, gdzie $b = 2, 10, 18, 24$.

W algorytmie generowania kluczy rundowych (algorytm 3.4) i w funkcji przedstawionej na rysunku (rys. 3.13 (b)), wykonywane są operacje:

- $\oplus$ – suma wyłączająca (XOR) słów 32-bitowych,
- *SL* – warstwa podstawień, realizowana z wykorzystaniem czterech równolegle połączonych S-bloków *S* o rozmiarze 8 × 8 bitów,
- $\lll b$ – cykliczne przesunięcie (rotacja) w lewo słowa 32-bitowego o *b* bitów, gdzie $b = 13, 23$.

Warto zauważyć, że runda szyfrowania/deszyfrowania oraz iteracja generowania kluczy rundowych są prawie identyczne. Jedyną różnicą w ich strukturze są liniowe funkcje *L* oraz *L′*, gdzie: $L(X) = X \oplus (X \lll 2) \oplus (X \lll 10) \oplus (X \lll 18) \oplus (X \lll 24)$, natomiast $L'(K) = K \oplus (K \lll 13) \oplus (K \lll 23)$.

Algorytm 3.4. Algorytm generowania kluczy rundowych szyfru SM4

WEJŚCIE: klucz główny $k = (MK_0, MK_1, MK_2, MK_3)$ o długości 128 bitów

WYJŚCIE: klucze rundowe $k_0, k_1, \ldots, k_{31}$ o długości 32 bity

PROCEDURA:

1. Maskuj klucz główny *k* z wykorzystaniem stałych FK_j, gdzie $j = 0, 1, 2, 3$ (tab. 3.10):
   - $(K_0, K_1, K_2, K_3) = (MK_0 \oplus FK_0, MK_1 \oplus FK_1, MK_2 \oplus FK_2, MK_3 \oplus FK_3)$.
2. Dla $i = 0, 1, \ldots, 31$ oblicz klucz rundowy k_i:
   - $k_i = K_{i+4} = K_i \oplus L'(SL(K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i))$,

   gdzie *SL* to warstwa czterech S-bloków *S* taka, że $SL(B0||B1||B2||B3) = S(B0)||S(B1)||S(B2)||S(B3)$ i *Bj* oznacza bajt, a CK_i to stała (tab. 3.10).

Tab. 3.10. SM4 – wartości stałych FK_j oraz CK_i

FK_0	0xa3b1bac6	FK_1	0x56aa3350	FK_2	0x677d9197	FK_3	0xb27022dc
CK_0	0x00070e15	CK_1	0x1c232a31	CK_2	0x383f464d	CK_3	0x545b6269
CK_4	0x70777e85	CK_5	0x8c939aa1	CK_6	0xa8afb6bd	CK_7	0xc4cbd2d9
CK_8	0xe0e7eef5	CK_9	0xfc030a11	CK_{10}	0x181f262d	CK_{11}	0x343b4249
CK_{12}	0x50575e65	CK_{13}	0x6c737a81	CK_{14}	0x888f969d	CK_{15}	0xa4abb2b9
CK_{16}	0xc0c7ced5	CK_{17}	0xdce3eaf1	CK_{18}	0xf8ff060d	CK_{19}	0x141b2229
CK_{20}	0x30373e45	CK_{21}	0x4c535a61	CK_{22}	0x686f767d	CK_{23}	0x848b9299
CK_{24}	0xa0a7aeb5	CK_{25}	0xbcc3cad1	CK_{26}	0xd8dfe6ed	CK_{27}	0xf4fb0209
CK_{28}	0x10171e25	CK_{28}	0x2c333a41	CK_{30}	0x484f565d	CK_{31}	0x646b7279

Zestawienie podstawowych cech szyfru SM4 przedstawiono w tabeli (tab. 3.11).

Tab. 3.11. SM4 – zestawienie cech szyfru

rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego |*k*|	liczba kluczy rundowych *nrk*	długość klucza rundowego |*rk*|	długość próbki kluczy rundowych |*rkp*|
128b	32	128b	32	32b	1024b

W tabeli (tab. 3.12) przedstawiono listę operacji funkcji szyfrującej/deszyfrującej i algorytmu generowania kluczy rundowych. Operacja *S* jest nieliniowa, a XOR i rotacja są liniowe (formuła (3.17)).

Tab. 3.12. SM4 – lista operacji

rodzaj	operacja	funkcja szyfrująca/deszyfrująca	algorytm generowania kluczy rundowych
XOR	$\oplus$	tak	tak
podstawienie	S	tak	tak
rotacja w lewo	$\lll b$	tak	tak

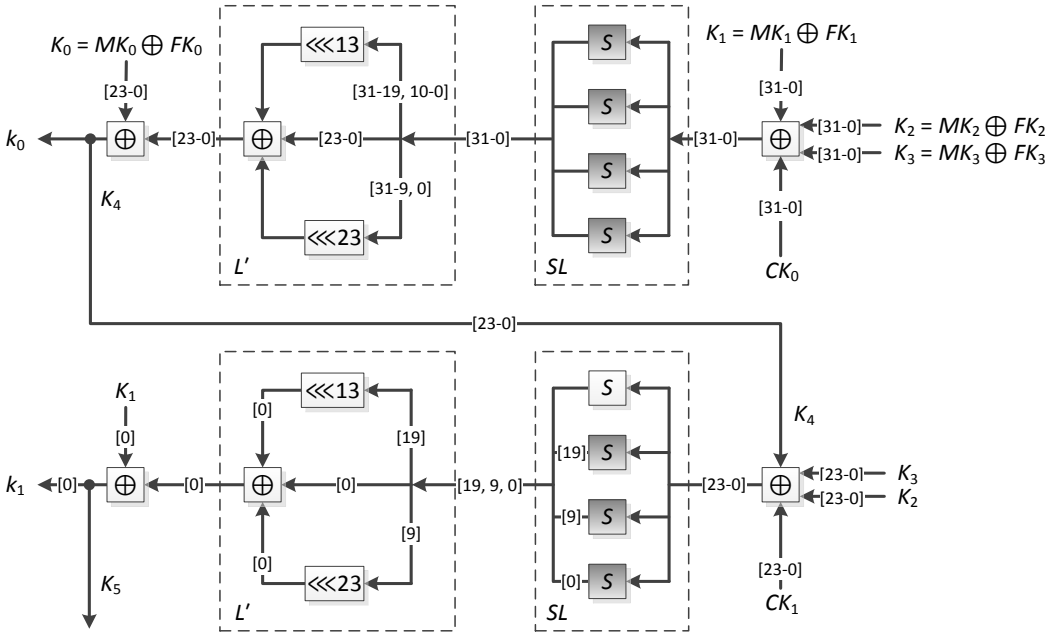
Zwiększenie liczby iteracji algorytmu generowania kluczy rundowych szyfru SM4 wymagałoby zdefiniowania kolejnych wartości stałych CK_i, w tablicy (tab. 3.10), dla $i > 31$. Nie jest możliwe zatem zwiększenie liczby rund szyfru SM4.

Wniosek 3.7.

W algorytmie generowania kluczy rundowych szyfru SM4 wykorzystywane są operacje liniowe i nieliniowe, w tym S-blok *S* o rozmiarze 8 × 8 bitów. Zwiększenie liczby iteracji algorytmu generowania kluczy rundowych szyfru SM4 nie jest możliwe, a zatem nie jest możliwe zwiększenie liczby rund szyfru SM4 (metoda B).

■

W pracach [68][77][99][109] przedstawiono podatność szyfru SM4 na kryptoanalizę różnicową oraz liniową, a najskuteczniejszy z teoretycznych ataków dotyczy zredukowanej, 24-rundowej wersji szyfru i możliwy jest do przeprowadzenia w czasie rzędu $2^{126.6}$ [78].



Rys. 3.14. SM4 – zależność bitu k_1[0] klucza rundowego od bitów klucza głównego $k = (MK_0, MK_1, MK_2, MK_3)$

Na rysunku (rys. 3.14) przedstawiono zależność bitu k_1[0] klucza rundowego, od bitów klucza głównego k. Przy określeniu tej zależności uwzględniono następujące własności funkcji elementarnych:

- $\oplus$ – bit i wyjścia (wartości) funkcji XOR zależy od bitu i wszystkich wejść (argumentów),
- $\lll b$ – bit i wyjścia rotacji w lewo o b bitów słowa l-bitowego, w przypadku LSB = 0 po prawej, zależy od bitu $(i + (l - b))$ **mod** l wejścia,
- S – bit i wyjścia S-bloku S zależy od wszystkich bitów wejścia.

Ostatecznie bit k_1[0] zależy od bitów MK_0[23-0], MK_1[31-0], MK_2[31-0] oraz MK_3[31-0], a więc zależy od 120 bitów 128-bitowego klucza k, co stanowi 93.75%.

Ten sam procentowy wynik osiągany jest dla dowolnego bitu klucza rundowego k_1. Z rysunku (rys. 3.14), analizując dolną iterację, można też odczytać, że bit k_0[0] zależy od bitów MK_0[0], MK_1[23-0], MK_2[23-0] oraz MK_3[23-0], a więc zależy od 73 bitów 128-bitowego klucza głównego k, co daje 57.03%. Wystarczy, w myśli, zastąpić zmienne (k_1, K_1, K_2, K_3, K_4) zmiennymi (k_0, K_0, K_1, K_2, K_3).

Z rysunku (rys. 3.14) wynika, że bit $k_1[0]$ nie zależy wyłącznie od bitów $K_0[31\text{-}24]$. Ponieważ K_1 zawierające 1 bit w iteracji #1 zawiera wszystkie bity w iteracji #0 to K_0 zawierające bity 23-0 w iteracji #0 zawiera wszystkie bity w, wyobrażonej, iteracji #(−1). Dlatego bit $k_2[0]$, po trzech iteracjach, zależy od wszystkich bitów klucza głównego k.

Wniosek 3.8.

W szyfrze SM4, klucz rundowy k_i (i = 0, 1, …, 31) zależy od wszystkich bitów 128-bitowego klucza głównego k, przy czym każdy bit klucza k_i zależy od liczby bitów klucza k: 73 (i = 0), 120 (i = 1) lub 128 (i = 2, 3, …, 31).

■

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru SM4, przedstawiono w postaci charakterystyki (charakterystyka 3.4).

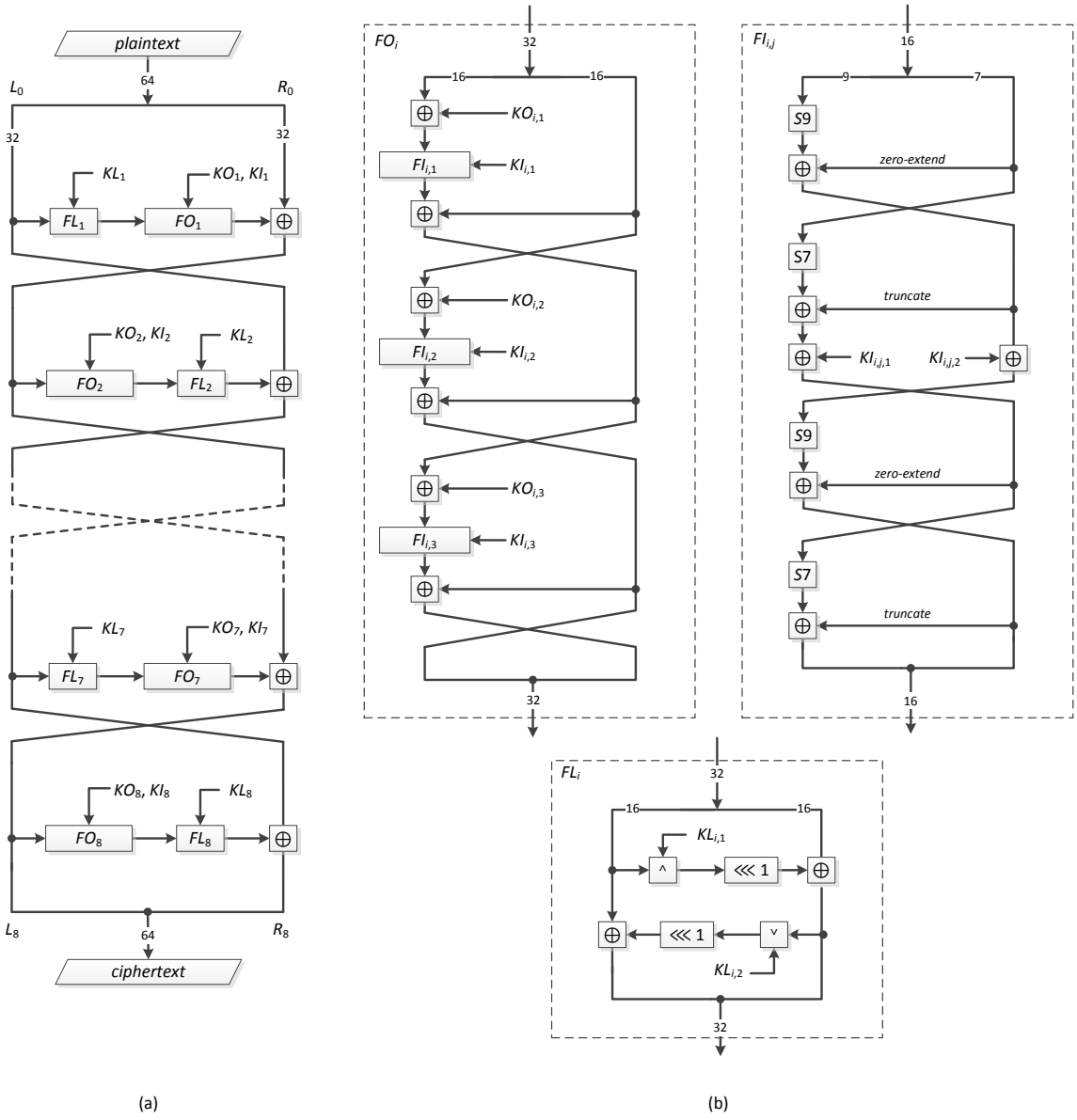
Charakterystyka 3.4. Charakterystyka algorytmu generowania kluczy rundowych szyfru SM4

1. Długość klucza głównego: $|\boldsymbol{k}|$ **= 128b**.
2. Długość klucza rundowego: $|\boldsymbol{k_i}|$ **= 32b (*i* = 1, 2, …, 31)**.
3. Liczba iteracji: $\boldsymbol{R}$ **= 32**.
4. Zależność k_i od bitów k:
   - **128b / 128b (100%)**.
5. Zależność bitu k_i od bitów k:
   - $\boldsymbol{k_0}$**: 73b / 128b (57.03%)**,
   - $\boldsymbol{k_1}$**: 120b / 128b (93.75%)**,
   - $\boldsymbol{k_2, k_3}$**, …,** $\boldsymbol{k_{31}}$**: 128b / 128b (100.00%)**.
6. Zależność grupy bitów k_i od grupy bitów k: **NIE**.
7. Operacje liniowe: $\oplus$, $\lll \boldsymbol{b}$.
8. Operacje nieliniowe: $\boldsymbol{S}$ **(8b × 8b)**.
9. Możliwości zwiększenia liczby iteracji: **NIE**.
10. Indywidualizacja iteracji: **przez stałe** $\boldsymbol{CK_i}$.
11. Teoretyczna ocena jakości algorytmu: **90.16%**.

3.2.5. KASUMI (1999/2007)

Szyfr blokowy KASUMI [89] przetwarza 64-bitowe bloki tekstu jawnego, z wykorzystaniem 128-bitowego klucza głównego k, w 64-bitowe bloki tekstu zaszyfrowanego.

Szyfr KASUMI ma postać sieci Feistela i działa w 8 rundach. W rundzie i (i = 1, 2, …, 8), wykorzystywanych jest osiem 16-bitowych kluczy rundowych $KL_{i,1}$, $KL_{i,2}$, $KO_{i,1}$, $KO_{i,2}$, $KO_{i,3}$, $KI_{i,1}$, $KI_{i,2}$, $KI_{i,3}$. Od roku 1999 szyfr KASUMI jest elementem standardu 3GPP/UMTS.

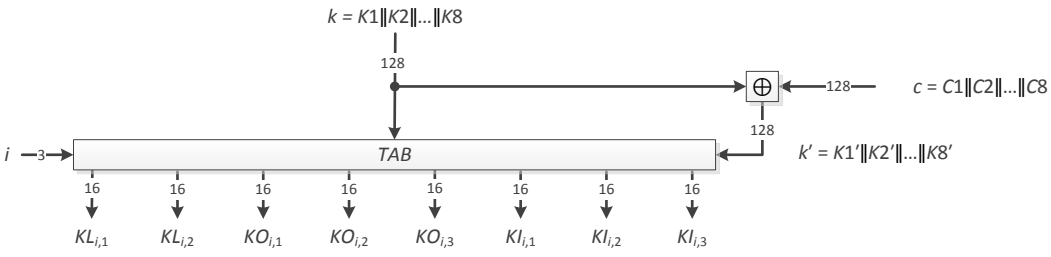


Rys. 3.15. KASUMI – struktura funkcji: (a) szyfrującej/deszyfrującej, (b) składowych funkcji bazowej *FO*, *FI* i *FL*

Strukturę funkcji szyfrującej/deszyfrującej szyfru KASUMI przedstawiono na rysunku (rys. 3.15 (a)). Podczas deszyfrowania wykorzystywana jest funkcja szyfrująca z odwrotną kolejnością kluczy rundowych, rozumianych jako 8-elementowe ciągi 16-bitowych podkluczy. Funkcja bazowa sieci Feistela jest złożeniem funkcji *FL* i funkcji *FO*, zależnej od trzech funkcji *FI*, przedstawionych na rysunku (rys. 3.15 (b)).

W funkcji szyfrującej/deszyfrującej wykonywane są następujące operacje:

- $\oplus$ – suma wyłączająca (XOR) słów 32-bitowych,
- FL_i – funkcja Feistela, zależna od danych oraz kluczy $KL_{i,1}$, $KL_{i,2}$ (i = 1, 2, …, 8), składająca się z operacji:
    - $\oplus$ – suma wyłączająca (XOR) słów 16-bitowych,
    - $\wedge$ – koniunkcja (AND) słów 16-bitowych,
    - $\vee$ – dysjunkcja (OR) słów 16-bitowych,
    - $\lll$ 1 – cykliczne przesunięcie (rotacja) w lewo słowa 16-bitowego o 1 bit,
- FO_i – funkcja Feistela, zależna od danych oraz kluczy $KO_{i,1}$, $KO_{i,2}$, $KO_{i,3}$ i $KI_{i,1}$, $KI_{i,2}$, $KI_{i,3}$ (i = 1, 2, …,8), składająca się z operacji:
    - $\oplus$ – suma wyłączająca (XOR) słów 16-bitowych,
    - $FI_{i,j}$ – funkcja Feistela (niezrównoważona), zależna od danych oraz kluczy $KI_{i,j}$ (j = 1, 2, 3), zawierająca operacje:
        - $\oplus$ – suma wyłączająca (XOR) słów 7- lub 9-bitowych,
        - $S7$, $S9$ – funkcje podstawień, S-bloki o rozmiarze, odpowiednio 7 × 7 bitów i 9 × 9 bitów,
        - *zero extend* – funkcja rozszerzenia 7-bitowego słowa do słowa 9-bitowego, przez uzupełnienie zerami,
        - *truncate* – funkcja obcięcia słowa 9-bitowego do słowa 7-bitowego.



Rys. 3.16. KASUMI – funkcja generowania kluczy rundowych rundy *i* (*i* = 1, 2, …, 8)

W algorytmie generowania kluczy rundowych (algorytm 3.5) i w funkcji przedstawionej na rysunku (rys. 3.16), wykonywane są następujące operacje:

- $\oplus$ – suma wyłączająca (XOR) słów o długości 16 lub 128 bitów,
- *TAB* – funkcja wyznaczania kluczy rundowych rundy i (i = 1, 2, …, 8), przedstawiona w tabeli (tab. 3.14), zawierające operację:

- $\lll b$ – cykliczne przesunięcie (rotacja) w lewo słowa 16-bitowego o b bitów.

Algorytm 3.5. Algorytm generowania kluczy rundowych szyfru KASUMI

WEJŚCIE: klucz główny k o długości 128 bitów

WYJŚCIE: sześćdziesiąt cztery 16-bitowe klucze rundowe ($KL_{1,1}$, $KL_{1,2}$, $KO_{1,1}$, $KO_{1,2}$, $KO_{1,3}$, $KI_{1,1}$, $KI_{1,2}$, $KI_{1,3}$), …, ($KL_{8,1}$, $KL_{8,2}$, $KO_{8,1}$, $KO_{8,2}$, $KO_{8,3}$, $KI_{8,1}$, $KI_{8,2}$, $KI_{8,3}$)

PROCEDURA:

1. Dla $k = K1||K2||\ldots||K8$, gdzie Kj dla j = 1, 2, …, 8, to 16-bitowy podklucz, oblicz klucz zmodyfikowany:
   - $k' = K1'||K2'||\ldots||K8'$, taki że $Kj' = Kj \oplus Cj$ (j = 1, 2, …, 8), a Cj to 16-bitowe stałe (tab. 3.13).
2. Dla i = 1, 2, …, 8 oblicz klucze rundowe:
   - $(KL_{i,1}, KL_{i,2}, KO_{i,1}, KO_{i,2}, KO_{i,3}, KI_{i,1}, KI_{i,2}, KI_{i,3}) = TAB(i)$.

Wynikiem działania algorytmu generowania kluczy rundowych są sześćdziesiąt cztery klucze rundowe, o długości 16 bitów, uzyskane z klucza głównego.

Tab. 3.13. KASUMI – tablica stałych *Cj* (*j* = 1, 2, …, 8)

***C*1**	0x0123	***C*2**	0x4567	***C*3**	0x89ab	***C*4**	0xcdef
***C*5**	0xfedc	***C*6**	0xba98	***C*7**	0x7654	***C*8**	0x3210

Tablica stałych Cj, dla j = 1, 2,…, 8 (tab. 3.13), zawiera osiem 16-bitowych stałych, wykorzystywanych w kroku 2 algorytmu (algorytm 3.5).

Tab. 3.14. KASUMI – tablica (funkcja) *TAB* wyznaczania kluczy rundowych rundy *i* (*i* = 1, 2, …, 8)

i	$KL_{i,1}$	$KL_{i,2}$	$KO_{i,1}$	$KO_{i,2}$	$KO_{i,3}$	$KI_{i,1}$	$KI_{i,2}$	$KI_{i,3}$
1	$K1\lll1$	$K3'$	$K2\lll5$	$K6\lll8$	$K7\lll13$	$K5'$	$K4'$	$K8'$
2	$K2\lll1$	$K4'$	$K3\lll5$	$K7\lll8$	$K8\lll13$	$K6'$	$K5'$	$K1'$
3	$K3\lll1$	$K5'$	$K4\lll5$	$K8\lll8$	$K1\lll13$	$K7'$	$K6'$	$K2'$
4	$K4\lll1$	$K6'$	$K5\lll5$	$K1\lll8$	$K2\lll13$	$K8'$	$K7'$	$K3'$
5	$K5\lll1$	$K7'$	$K6\lll5$	$K2\lll8$	$K3\lll13$	$K1'$	$K8'$	$K4'$
6	$K6\lll1$	$K8'$	$K7\lll5$	$K3\lll8$	$K4\lll13$	$K2'$	$K1'$	$K5'$
7	$K7\lll1$	$K1'$	$K8\lll5$	$K4\lll8$	$K5\lll13$	$K3'$	$K2'$	$K6'$
8	$K8\lll1$	$K2'$	$K1\lll5$	$K5\lll8$	$K6\lll13$	$K4'$	$K3'$	$K7'$

Tablica (funkcja) *TAB* wyznaczania kluczy rundowych rundy i (i = 1, 2, …, 8) (tab. 3.14), wykorzystywana w kroku 3 algorytmu (algorytm 3.5), przypisuje każdemu podkluczowi rundowemu 16-bitową wartość, obliczoną jako rotowany podklucz klucza głównego k lub podklucz klucza k', wyznaczonego w kroku 2 algorytmu (algorytm 3.5).

Tab. 3.15. KASUMI – zestawienie cech szyfru

rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego |*k*|	liczba kluczy rundowych *nrk*	długość klucza rundowego |*rk*|	długość próbki kluczy rundowych |*rkp*|
64b	8	128b	64	16b	1024b

W tabeli (tab. 3.15) przedstawiono zestawienie podstawowych cech szyfru KASUMI.

Tab. 3.16. KASUMI – lista operacji

rodzaj	operacja	funkcja szyfrująca/deszyfrująca	algorytm generowania kluczy rundowych
XOR	$\oplus$	tak	tak
podstawienie	*S*7, *S*9	tak	nie
koniunkcja (AND)	$\wedge$	tak	nie
dysjunkcja (OR)	$\vee$	tak	nie
rotacja w lewo	$\lll b$	tak	tak
rozszerzenie	*zero-extend*	tak	nie
obcięcie	*truncate*	tak	nie

W tabeli (tab. 3.16) przedstawiono listę operacji funkcji szyfrującej/deszyfrującej i algorytmu generowania kluczy rundowych. Operacje *S*7, *S*9, $\wedge$, $\vee$ są nieliniowe, a pozostałe operacje są liniowe (formuła (3.17)).

Zwiększenie liczby iteracji algorytmu generowania kluczy rundowych szyfru KASUMI wymagałoby zdefiniowania kluczy rundowych w tablicy (tab. 3.14) dla $i > 8$. Nie jest możliwe zatem zwiększenie liczby rund szyfru KASUMI.

Wniosek 3.9.

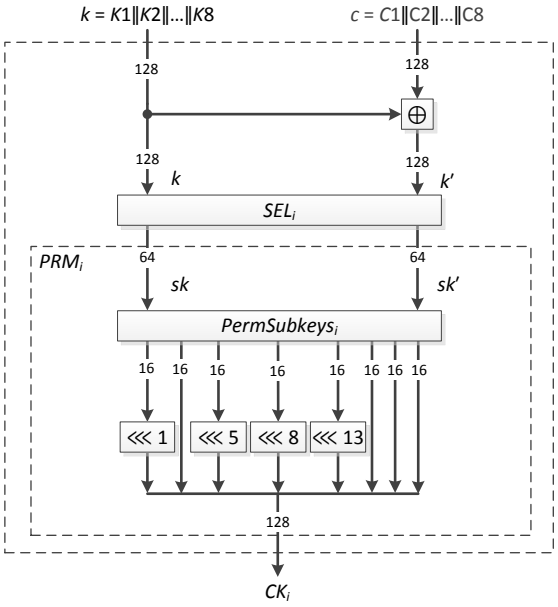
W algorytmie generowania kluczy rundowych szyfru KASUMI wykorzystywane są wyłącznie operacje liniowe. Zwiększenie liczby iteracji tego algorytmu nie jest możliwe, a zatem nie jest możliwe zwiększenie liczby rund szyfru KASUMI (metoda B).

■

Zdefiniujmy dla iteracji i (i = 1, 2, …, 8), złożony klucz (nadklucz) rundowy $CK_i = (KL_{i,1}, KL_{i,2}, KO_{i,1}, KO_{i,2}, KO_{i,3}, KI_{i,1}, KI_{i,2}, KI_{i,3})$. Klucz CK_i, zgodnie z tablicą (funkcją) *TAB* (tab. 3.14), można określić następująco:

$$CK_i = Ki \lll 1 || KI_{(i+2)} \oplus CI_{(i+2)} || KI_{(i+1)} \lll 5 || KI_{(i+5)} \lll 8 || KI_{(i+6)} \lll 13 || KI_{(i+4)} \oplus CI_{(i+4)} || KI_{(i+3)} \oplus CI_{(i+3)} || KI_{(i+7)} \oplus CI_{(i+7)}, \quad (3.19)$$

gdzie $I(j) = (j-1)$ **mod** 8 + 1 (j = 1, 2, 3, …).



Rys. 3.17. KASUMI – schemat zastępczy funkcji generowania klucza rundowego CK_i (i = 1, 2, …, 8)

W schemacie zastępczym funkcji generowania klucza rundowego CK_i, przedstawionym na rysunku (rys. 3.17), wykonywane są operacje:

- SEL_i – funkcja wyboru 4 podkluczy klucza k i 4 podkluczy klucza k' taka, że dla każdego j = 1, 2, …, 8 zostaje wybrany podklucz Kj albo Kj' – wyjście SEL_i jest zależne od wszystkich bitów klucza k,
- PRM_i – permutacja bitowa taka, że:
  - $PermSubkeys_i$ – permutuje podklucze,
  - podklucze k są rotowane w lewo o 1, 5, 8 lub 13 bitów – podklucz wyjścia PRM_i jest zależny od jednego podklucza wejścia PRM_i.

Wniosek 3.10.

W szyfrze KASUMI, klucz rundowy CK_i (i = 1, 2, …, 8) jest zależny od wszystkich bitów 128-bitowego klucza głównego k, przy czym każdy bit klucza CK_i zależy od jednego bitu klucza k.

■

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru KASUMI, przedstawiono w postaci charakterystyki (charakterystyka 3.5).

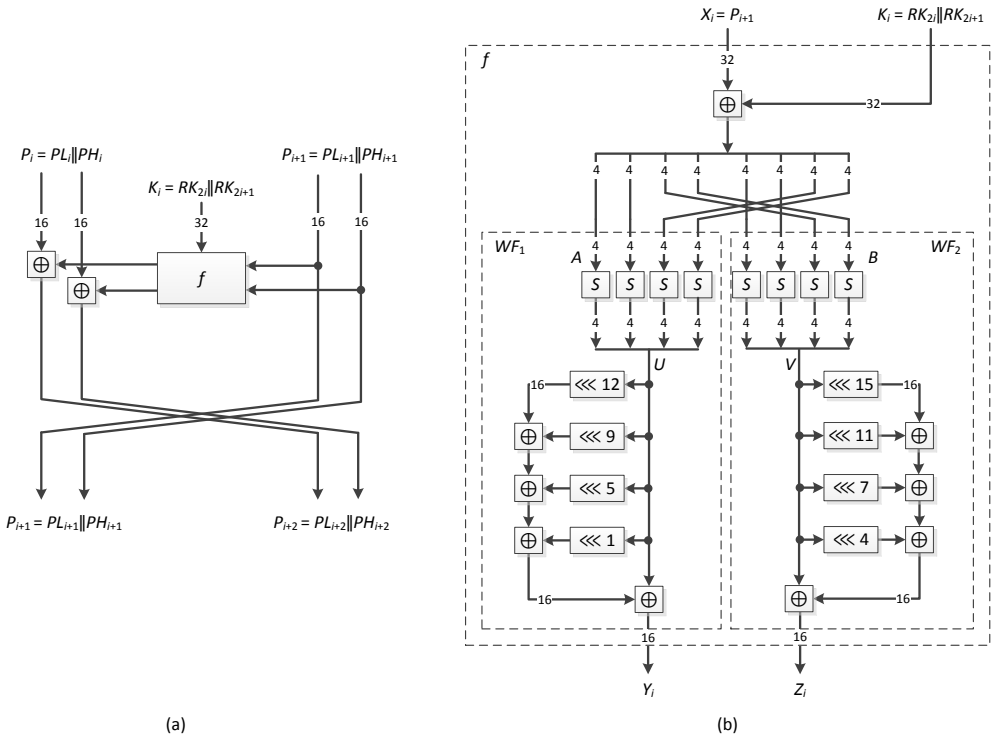
Charakterystyka 3.5. Charakterystyka algorytmu generowania kluczy rundowych szyfru KASUMI

1. Długość klucza głównego: $|k| = 128b$.

2. Długość klucza rundowego: $|CK_i| = 128b$ ($i = 1, 2, \ldots, 8$).
3. Liczba iteracji: $R = 8$.
4. Zależność CK_i od bitów k:
   - **128b / 128b (100%)**.
5. Zależność bitu CK_i od bitów k:
   - **1b / 128b (0.78%)**.
6. Zależność grupy bitów CK_i od grupy bitów k: **16b**.
7. Operacja liniowe: $\oplus$, $\lll b$.
8. Operacje nieliniowe: **BRAK**.
9. Możliwości zwiększenia liczby iteracji: **NIE**.
10. Indywidualizacja iteracji: **przez stałe *Cj***.
11. Teoretyczna ocena jakości algorytmu: **0.78%**.

3.2.6. FeW (2014/2019)

Szyfr blokowy FeW [73] to tzw. lekki (ang. *lightweight*) szyfr, przetwarzający bloki danych o rozmiarze $n = 64$ bity, w $r = 32$ rundach, z kluczem głównym k o długości 80 bitów lub 128 bitów. Zależnie od zastosowanej długości klucza głównego k, szyfr oznaczany jest jako FeW-80 lub FeW-128.

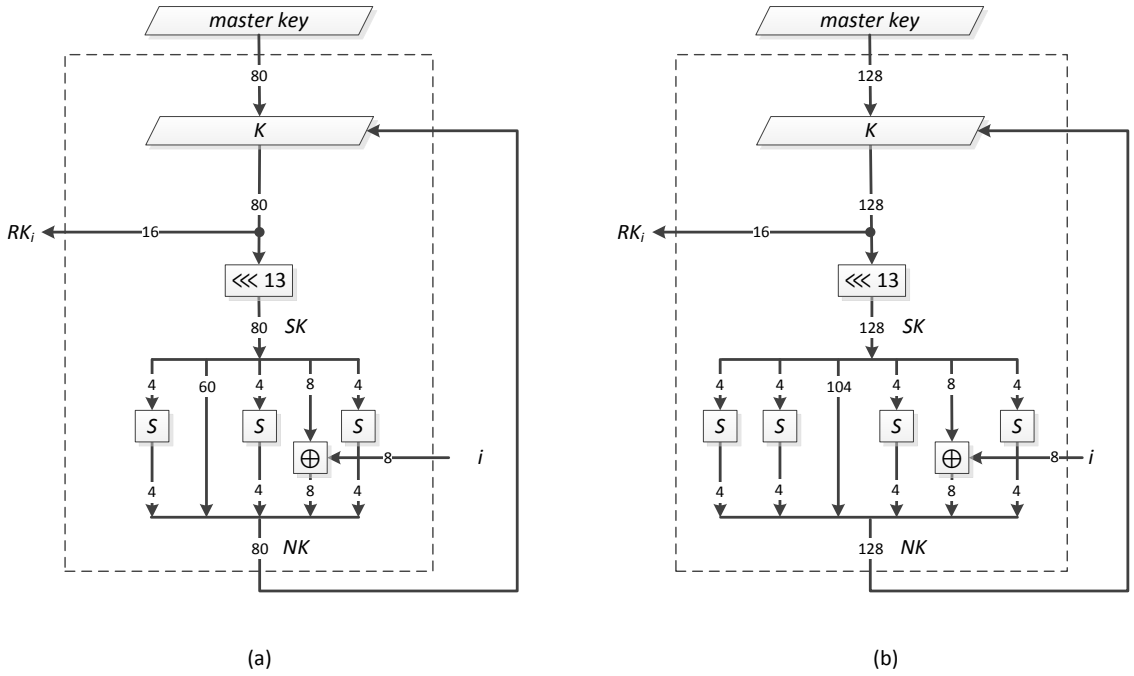


Rys. 3.18. FeW – struktura: (a) rundy *i* (*i* = 0, 1, …, 31) funkcji szyfrującej/deszyfrującej, (b) funkcji bazowej *f*

Strukturę rundy funkcji szyfrującej/deszyfrującej szyfru FeW przedstawiono na rysunku (rys. 3.18 (a)). W ostatniej rundzie zamiana słów 32-bitowych nie jest wykonywana. Podczas deszyfrowania, wykorzystywana jest funkcja szyfrująca z odwrotną kolejnością kluczy rundowych. Zastosowana funkcja bazowa *f* sieci Feistela przedstawiona została na rysunku (rys. 3.18 (b)).

W funkcji szyfrującej/deszyfrującej wykonywane są następujące operacje:

- $\oplus$ – suma wyłączająca (XOR) słów 32-bitowych,
- WF_1 – funkcja złożona, w której wykonywane są:
  - $\oplus$ – suma wyłączająca (XOR) słów 16-bitowych,
  - S – funkcja podstawień, S-blok o rozmiarze 4 × 4 bity,
  - $\lll b$ – cykliczne przesunięcie (rotacja) w lewo słowa 16-bitowego o b bitów, gdzie b = 1, 5, 9, 12.
- WF_2 – funkcja złożona, w której wykonywane są:
  - $\oplus$ – suma wyłączająca (XOR) słów 16-bitowych,
  - S – funkcja podstawień, S-blok o rozmiarze 4 × 4 bity,
  - $\lll b$ – cykliczne przesunięcie (rotacja) w lewo słowa 16-bitowego o b bitów, gdzie b = 4, 7, 11, 15.



Rys. 3.19. FeW – struktura funkcji generowania podkluczy rundowych RK_i (i = 0, 1, …, 63), w przypadku: (a) $|k|$ = 80b, (b) $|k|$ = 128b

W algorytmie generowania kluczy rundowych (algorytm 3.6) i w funkcjach przedstawionych na rysunku (rys. 3.19), wykonywane są operacje:

- $\lll$ 13 – cykliczne przesunięcie (rotacja) w lewo słowa 80- lub 128-bitowego o 13 bitów,
- S – funkcja podstawień, S-blok o rozmiarze 4 × 4 bity,
- $\oplus$ – suma wyłączająca (XOR) słów 8-bitowych.

Algorytm 3.6. Algorytm generowania kluczy rundowych szyfru FeW

WEJŚCIE: klucz główny k o długości 80 lub 128 bitów

WYJŚCIE: klucze rundowe $K_0, K_1, \ldots, K_{31}$ o długości 32 bity

PROCEDURA:

1. Wpisz k do rejestru K, tj. $K = k$.
2. Dla $i = 0$ do 63 wykonaj:
   - $RK_i = K[0\text{-}15]$,
   - $SK = K \lll 13$,
   - w przypadku:
     - $|k| = 80$: $NK = S(SK[0\text{-}3])||SK[4\text{-}63]||S(SK[64\text{-}67])||SK[68\text{-}75] \oplus i \; || \; S(SK[76\text{-}79])$,
     - $|k| = 128$: $NK = S(SK[0\text{-}3])||S(SK[4\text{-}7])||SK[8\text{-}111]||S(SK[112\text{-}115])||SK[116\text{-}123] \oplus i \; ||S(SK[124\text{-}127])$,
   - wpisz NK do rejestru K, tj. $K=NK$.
3. Dla $i = 0$ do 31 podstaw:
   - $K_i = RK_{2i}||RK_{2i+1}$.

Niezależnie od długości klucza głównego, wynikiem działania algorytmu generowania kluczy rundowych są trzydzieści dwa klucze rundowe o długości 32 bity. W przypadku wariantu FeW-80 stosowane są trzy S-bloki S, a w przypadku wariantu FeW-128 – cztery S-bloki.

Tab. 3.17. FeW – zestawienie cech szyfru

rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego *|k|*	liczba kluczy rundowych *nrk*	długość klucza rundowego *|rk|*	długość próbki kluczy rundowych *|rkp|*
64b	32	80b	32	32b	1024b
		128b			

W tabeli (tab. 3.17) przedstawiono zestawienie podstawowych cech szyfru FeW.

Tab. 3.18. FeW – lista operacji

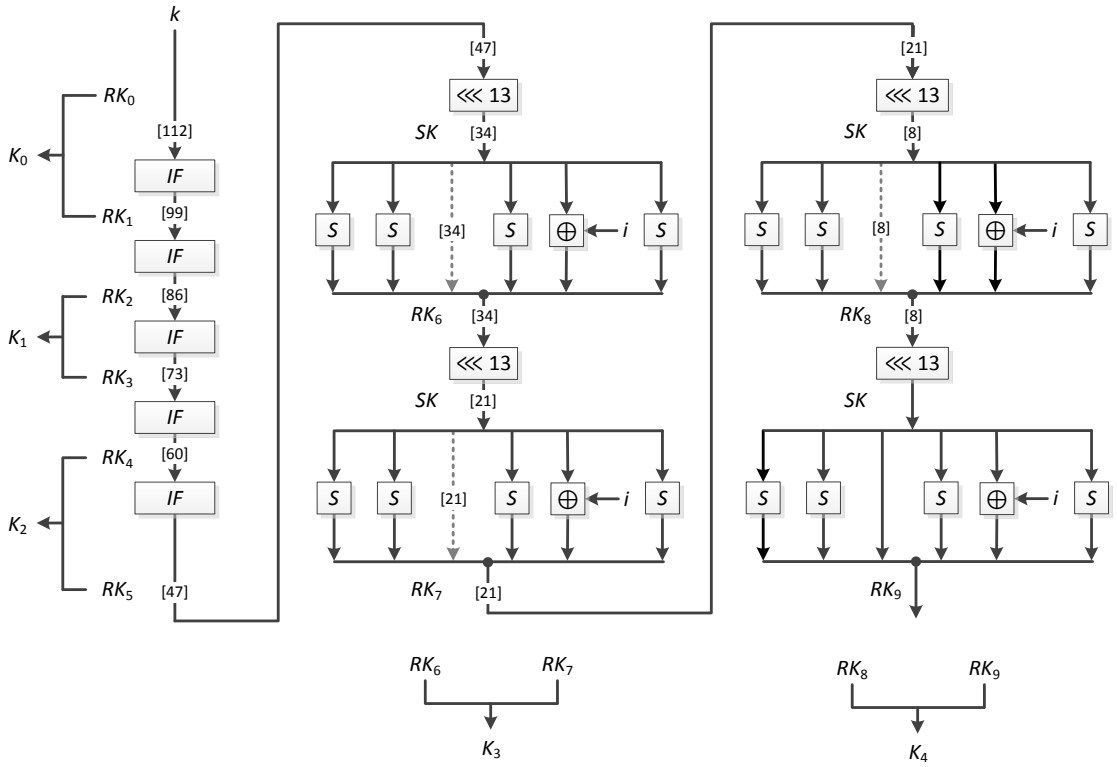
rodzaj	operacja	funkcja szyfrująca/deszyfrująca	algorytm generowania kluczy rundowych
XOR	$\oplus$	tak	tak
podstawienie	S	tak	tak
rotacja w lewo	$\lll b$	tak	tak

W tabeli (tab. 3.18) przedstawiono listę operacji funkcji szyfrującej/deszyfrującej i algorytmu generowania kluczy rundowych. Operacja S jest nieliniowa, a pozostałe operacje są liniowe (formuła (3.17)).

Wniosek 3.11.

W algorytmie generowania kluczy rundowych szyfru FeW wykorzystywane są operacje nieliniowe – podstawienie S oraz operacje liniowe – XOR i rotacja. W algorytmie tym możliwe jest zwiększenia liczby iteracji (metoda B) i możliwe jest odpowiednie zwiększenie liczby rund szyfru, przy założeniu zwiększenia liczby bitów licznika i iteracji.

■



Rys. 3.20. FeW 64/128 – zależność bitu $RK_8[8]$ podklucza rundowego od bitów klucza głównego k

Na rysunku (rys. 3.20) przedstawiono zależność bitu $RK_8[8]$ podklucza rundowego od bitów klucza głównego k. Przy określeniu tej zależności uwzględniono następujące własności funkcji elementarnych:

- $\oplus$ – bit i wyjścia (wartości) funkcji XOR zależy od bitu i wszystkich wejść (argumentów),
- $\lll b$ – bit i wyjścia rotacji w lewo o b bitów słowa l-bitowego, w przypadku LSB = 0 po lewej, zależy od bitu $(i + b)$ **mod** l wejścia,
- S – bit i wyjścia S-bloku S zależy od wszystkich bitów wejścia.

Ostatecznie bit $RK_8[8]$ zależy od 1 bitu, $k[112]$, 128-bitowego klucza k, co stanowi 0.78%. Z rysunku (rys. 3.20) można też odczytać, że bity $RK_7[8]$, $RK_6[8]$, …, $RK_0[8]$ zależą od 1 bitu klucza k, odpowiednio od bitu: 99, 86, …, 8. Ponieważ klucz rundowy $K_i = RK_{2i}||RK_{2i+1}$, dla $i = 0, 1, \ldots, 31$, to bity kluczy rundowych $K_0[8]$, $K_1[8]$, …, $K_4[8]$ zależą od 1 bitu klucza głównego k. Bit 8 stanowi najgorszy przypadek, w którym podczas propagacji zależności ani razu nie natrafiono na S-blok S. Podobnie jak zależność pojedynczego bitu, można określić zależność całego klucza rundowego $K_i = RK_{2i}||RK_{2i+1}$ od bitów klucza głównego k.

Wniosek 3.12.

W szyfrze FeW-128 klucze rundowe K_0, K_1,…, K_4 zależą od 29 bitów 128-bitowego klucza głównego k, przy czym bit klucza K_i, w najgorszym przypadku, zależy od liczby bitów klucza k: 1 ($i = 0, 1, \ldots, 4$).

■

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru FeW, przedstawiono w postaci charakterystyki (charakterystyka 3.6).

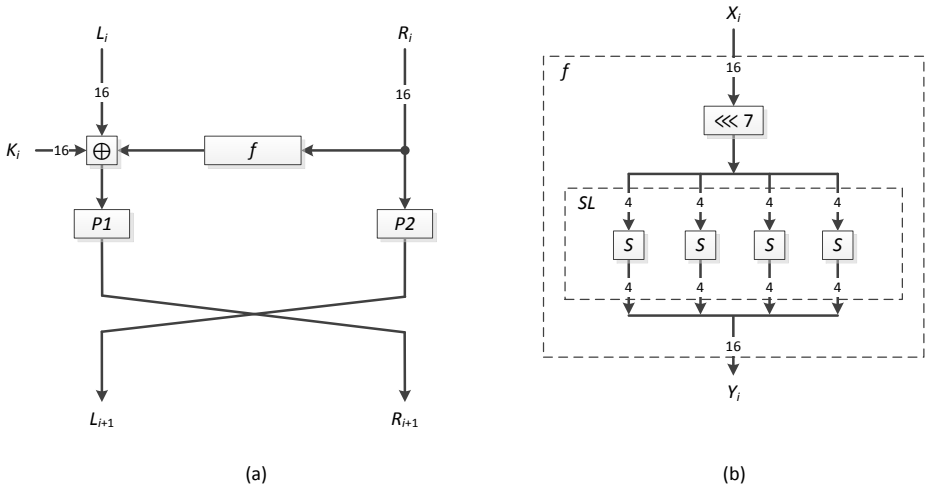
Charakterystyka 3.6. Charakterystyka algorytmu generowania kluczy rundowych szyfru FeW

1. Długość klucza głównego: $|k|$ **= 80b, 128b**.
2. Długość klucza rundowego: $|K_i|$ **= 32b (**i **= 0, 1, …, 31)**.
3. Liczba iteracji: R **= 64**.
4. Zależność K_i od bitów k (FeW-128):
   - K_0, K_1, K_2, K_3, K_4**: 29b / 128b (22.66%)**.
5. Zależność bitu K_i od bitów k (FeW-128):
   - $K_0[8], K_1[8], K_2[8], K_3[8], K_4[8]$**: 1b / 128b (0.78%)**.

6. Zależność grupy bitów K_i od grupy bitów k: **NIE**.
7. Operacja liniowe: $\oplus$, $\lll$ ***b***.
8. Operacje nieliniowe: ***S* (4b × 4b)**.
9. Możliwości zwiększenia liczby iteracji: **warunek – zwiększenie liczby bitów licznika iteracji *i***.
10. Indywidualizacja iteracji: **przez stan licznika *i***.
11. Teoretyczna ocena jakości algorytmu: **0.78%**.

3.2.7. LCB-IoT (2021)

Szyfr LCB-IoT [92] został opublikowany w 2021 roku i należy do kategorii tzw. ultra lekkich (ang. *ultra-lightweight*) szyfrów blokowych, przeznaczonych do zastosowań w Internecie rzeczy (ang. *Internet of Things*). Szyfr LCB-IoT jest 32-rundową siecią Feistela, która przetwarza 32-bitowe bloki tekstu jawnego, z wykorzystaniem 80-bitowego klucza głównego *k*, w 32-bitowe bloki tekstu zaszyfrowanego. Przekształcenie odbywa się w $r = 32$ rundach i w rundzie i ($i = 1, 2, \ldots, 32$) wykorzystywany jest jeden 16-bitowy klucz rundowy K_i.



Rys. 3.21. LCB-IoT – struktura: (a) rundy *i* (*i* = 1, 2, …, 31) funkcji szyfrującej/deszyfrującej, (b) funkcji bazowej *f*

Na rysunku (rys. 3.21 (a)) przedstawiono strukturę rundy i ($i = 1, 2, \ldots, 31$) funkcji szyfrującej/deszyfrującej szyfru LCB-IoT. W ostatniej rundzie ($i = 32$), nazywanej transformacją wyjściową, operacje *P1* i *P2*, a także zamiana słów, nie są wykonywane.

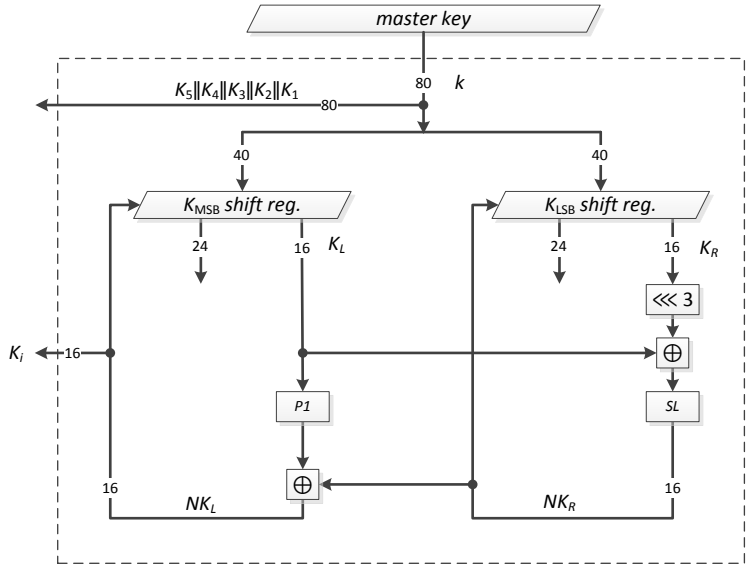
Podczas deszyfrowania wykorzystywana jest funkcja szyfrująca z odwrotną kolejnością kluczy rundowych. Funkcję bazową *f* sieci Feistela przedstawiono na rysunku (rys. 3.21 (b)).

W funkcji szyfrującej/deszyfrującej wykonywane są następujące operacje:

- $\oplus$ – suma wyłączająca (XOR) dwóch słów 16-bitowych i 16-bitowego klucza rundowego,
- *f* – funkcja bazowa, złożona z operacji:
  - $\lll$ 7 – cykliczne przesunięcie (rotacja) w lewo słowa 16-bitowego o 7 bitów,
  - *SL* – warstwa podstawień, z wykorzystaniem czterech S-bloków *S*, o rozmiarze 4 × 4 bity,
- *P1*, *P2* – inwolucyjne permutacje bitowe, tj. $P1^{-1} = P1$ i $P2^{-1} = P2$.

W algorytmie generowania kluczy rundowych (algorytm 3.7) i w funkcji przedstawionej na rysunku (rys. 3.22), wykonywane są następujące operacje:

- $\oplus$ – suma wyłączająca (XOR) słów o długości 16 bitów,
- $\lll$ 3 – cykliczne przesunięcie (rotacja) w lewo słowa 16-bitowego o 3 bity,
- *SL* – warstwa podstawień, z wykorzystaniem czterech S-bloków *S*, o rozmiarze 4 × 4 bity,
- *P1* – inwolucyjna permutacja bitowa, tj. $P1^{-1} = P1$ (tab. 3.19),
- » *b* – przesunięcie w prawo słowa 40-bitowego o *b* bitów, gdzie *b* = 16.



Rys. 3.22. LCB-IoT – struktura funkcji generowania kluczy rundowych (*i* = 6, 7, …, 32)

W tabeli (tab. 3.19) przedstawiono permutację *P1*, wykorzystywaną w rundzie szyfru i podczas generowania kluczy rundowych. Bit 1 argumentu permutacji jest bitem 13 jej wartości, a ponieważ *P1* jest inwolucją to bit 13 argumentu jest bitem 1 jej wartości.

Tab. 3.19. LCB-IoT – tablica permutacji *P1*

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
P1(j)	13	10	7	12	9	14	3	16	5	2	15	4	1	6	11	8

Algorytm 3.7. Algorytm generowania kluczy rundowych szyfru LCB-IoT

WEJŚCIE: klucz główny k o długości 80 bitów

WYJŚCIE: klucze rundowe $K_1, K_2, \ldots, K_{32}$ o długości 16 bitów

PROCEDURA:

1. Podstaw: $K_5||K_4||K_3||K_2||K_1 = k$.
2. Wpisz k do rejestrów K_{MSB} i K_{LSB}, tj.: $K_{MSB}||K_{LSB} = k$.
3. Dla $i = 6$ do 32 wykonaj:
   - $K_L = K_{MSB}[15\text{-}0]$, $K_R = K_{LSB}[15\text{-}0]$,
   - $NK_R = SL(K_L \oplus (K_R \lll 3))$,
   - $NK_L = P1(K_L) \oplus NK_R$,
   - $K_i = NK_L$,
   - wpisz NK_L i NK_R do rejestrów K_{MSB} i K_{LSB}, z przesunięciem ich zawartości w prawo o 16 bitów: $K_{MSB} = NK_L||(K_{MSB} \gg 16)$, $K_{LSB} = NK_R||(K_{LSB} \gg 16)$.

Wynikiem działania algorytmu generowania kluczy rundowych są trzydzieści dwa klucze rundowe, o długości 16 bitów, uzyskane z klucza głównego o długości 80 bitów.

Tab. 3.20. LCB-IoT – zestawienie cech szyfru

rozmiar bloku danych *n*	**liczba rund *r***	**długość klucza głównego *|k|***	**liczba kluczy rundowych *nrk***	**długość klucza rundowego *|rk|***	**długość próbki kluczy rundowych *|rkp|***
32b	32	80b	32	16b	512b

Zestawienie podstawowych cech szyfru LCB-IoT, przedstawiono w tabeli (tab. 3.20).

Tab. 3.21. LCB-IoT – lista operacji

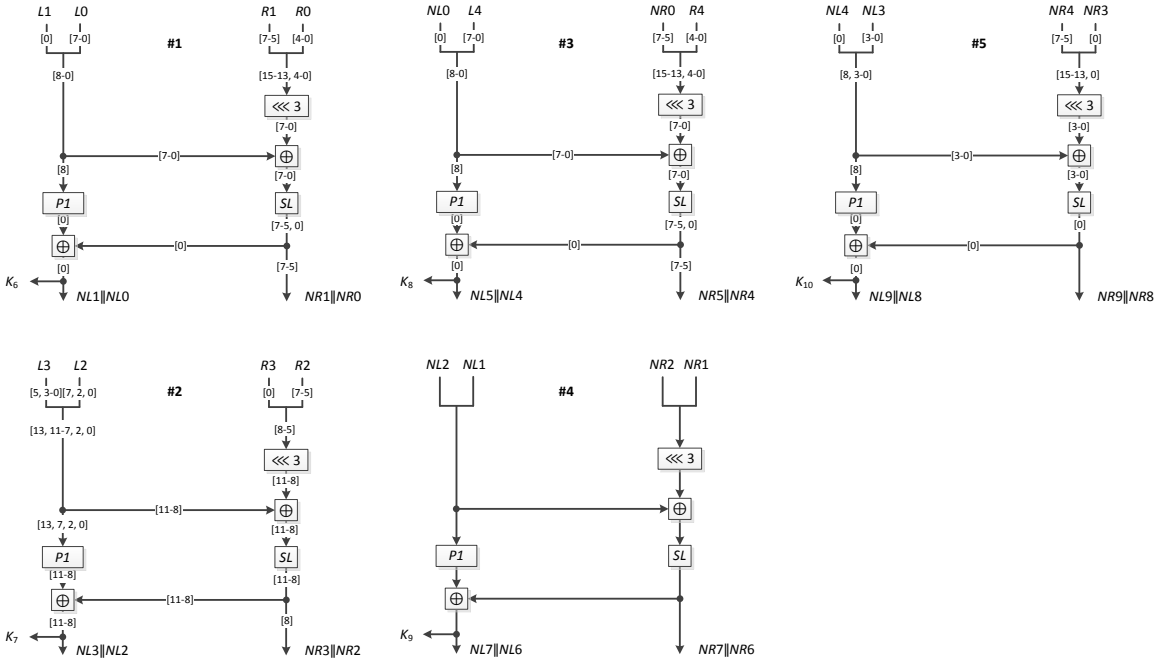
rodzaj	**operacja**	**funkcja szyfrująca/deszyfrująca**	**algorytm generowania kluczy rundowych**
XOR	$\oplus$	tak	tak
permutacja	*P1*	tak	tak
permutacja	*P2*	tak	nie
podstawienie	*SL*	tak	tak
rotacja w lewo	$\lll b$	tak	tak
przesunięcie w prawo	$\gg b$	nie	tak

W tabeli (tab. 3.21) przedstawiono listę operacji funkcji szyfrującej/deszyfrującej i algorytmu generowania kluczy rundowych. Operacja podstawień *S* jest nieliniowa, a pozostałe operacje są liniowe (formuła (3.17)). Konstrukcja algorytmu generowania kluczy rundowych umożliwia zwiększenie liczby iteracji tego algorytmu do dowolnej wartości.

Wniosek 3.13.

W algorytmie generowania kluczy rundowych szyfru LCB-IoT wykorzystywane są zarówno operacje liniowe – suma wyłączająca (XOR), permutacja *P1*, rotacja, przesunięcie jak i operacje nieliniowe – podstawienia (S-blok *S* o rozmiarze 4 × 4 bity). Przez zwiększenie liczby iteracji tego algorytmu możliwe jest zwiększenie liczby rund szyfru LCB-IoT (metoda B).

■



Rys. 3.23. LCB-IoT – zależność bitu $K_{10}[0]$ klucza rundowego od bitów klucza głównego *k*

Na rysunku (rys. 3.23) przedstawiono zależność bitu $K_{10}[0]$ klucza rundowego od bitów klucza głównego *k*. Przy określeniu tej zależności uwzględniono następujące własności funkcji elementarnych:

- $\oplus$ – bit *i* wyjścia (wartości) funkcji XOR zależy od bitu *i* wszystkich wejść (argumentów),
- $\lll b$ – bit *i* wyjścia rotacji w lewo o *b* bitów słowa *l*-bitowego, w przypadku LSB = 0 po prawej, zależy od bitu $(i + (l - b))$ **mod** *l* wejścia,
- *S* – bit *i* wyjścia S-bloku *S* zależy od wszystkich bitów wejścia.

Ostatecznie bit $K_{10}[0]$ zależy od bitów $L4[7\text{-}0]$, $L3[5, 3\text{-}0]$, $L2[7, 2, 0]$, $L1[0]$, $L0[7\text{-}0]$, $R4[4\text{-}0]$, $R3[0]$, $R2[7\text{-}5]$, $R1[7\text{-}5]$ oraz $R0[4\text{-}0]$, a ponieważ $L4||L3||\ldots||L0||R4||R3||\ldots||R0 = k$ to bit $K_{10}[0]$ zależy od 42 bitów 80-btowego klucza k, co stanowi 52.50%.

Z rysunku (rys. 3.23) można też odczytać, że:

- bit $K_9[0]$ zależy od 29 bitów,
- bit $K_8[0]$ zależy od 22 bitów,
- bit $K_7[0]$ zależy od 9 bitów,
- bit $K_6[0]$ zależy od 9 bitów.

Podobne wyniki uzyskiwane są dla dowolnego bitu kluczy rundowych K_6, K_7, …, K_{10}. Ponieważ $K_5||K_4||\ldots||K_1 = k$ to każdy bit kluczy rundowych K_5, K_4, …, K_1 zależy od 1 bitu klucza głównego k.

Wniosek 3.14.

W szyfrze LCB-IoT klucz rundowy K_i (i = 1, 2, …, 32) zależy od liczby bitów: 16 (i = 1, 2, …, 5), 32 (i = 6, 7), 48 (i = 8), 64 (i = 9) lub 80 (i = 10, 11, …, 32) 80-bitowego klucza głównego k, przy czym bit klucza K_i, zależy od liczby bitów klucza k: 1 (i = 1, 2, …, 5), 9 (i = 6, 7), 22 (i = 8), 29 (i = 9), 42 (i = 10).

■

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru LCB-IoT, przedstawiono w postaci charakterystyki (charakterystyka 3.7).

Charakterystyka 3.7. Charakterystyka algorytmu generowania kluczy rundowych szyfru LCB-IoT

1. Długość klucza głównego: $|k| = 80b$.
2. Długość klucza rundowego: $|K_i| = 16b$ ($i = 1, 2, \ldots, 32$).
3. Liczba iteracji: $R = 27$.
4. Zależność K_i od bitów k:
   - $K_1, K_2, \ldots, K_5$: 16b / 80b (20.00%),
   - K_6, K_7: 32b / 80b (40.00%),
   - K_8: 48b / 80b (60.00%),
   - K_9: 64b / 80b (80.00%),
   - $K_{10}, K_{11}, \ldots, K_{32}$: 80b / 80b (100.00%).
5. Zależność bitu K_i od bitów k:
   - $K_1, K_2, \ldots, K_5$: 1b / 80b (1.25%),

- **K_6, K_7: 9b / 80b (11.25%),**
- **K_8: 22b / 80b (27.50%),**
- **K_9: 29b / 80b (36.25%),**
- **K_{10}: 42b / 80b (52.50%).**

6. Zależność grupy bitów K_i od grupy bitów k: **NIE**.
7. Operacja liniowe: **$\oplus$, *P1*, $\lll$ *b*, $\ggg$ *b*.**
8. Operacje nieliniowe: ***S*** **(4b × 4b)**.
9. Możliwości zwiększenia liczby iteracji: **TAK**.
10. Indywidualizacja iteracji: **NIE.**
11. Teoretyczna ocena jakości algorytmu:
    - **K_1, K_2, …, K_5: 1.25%,**
    - **(K_6, K_7, …, K_{10}: 27.75%).**

3.3. Szyfry blokowe o konstrukcji SPN

3.3.1. AES (Rijndael) (1998)

Szyfr Rijndael [47][48] to rodzina szyfrów przetwarzająca bloki danych 128-, 192- lub 256-bitowe, z wykorzystaniem 128-, 192- lub 256-bitowego klucza głównego. Liczba rund *Nr* szyfru Rijndael zależna jest od rozmiaru przetwarzanego bloku oraz od długości klucza głównego, co pokazano w tabeli (tab. 3.22).

Blok danych rozmiaru *n* bitów, przedstawiany jest w postaci dwuwymiarowej tablicy bajtów, nazywanej stanem i oznaczanej jako *s*, o czterech wierszach i *Nb* kolumnach. W kolumnach zapisane są 32-bitowe podbloki bloku danych, a więc $Nb = n / 32$. Klucz główny *k* składa się z *Nk* 32-bitowych podkluczy, a zatem $Nk = |k| / 32$.

Tab. 3.22. Rijndael – liczba rund *Nr* szyfru i parametry *Nb, Nk*

długość klucza |*k*|	**rozmiar bloku *n***		
	128b, *Nb* = 4	**192b, *Nb* = 6**	**256b, *Nb* = 8**
128b, *Nk* = 4	10	12	14
192b, *Nk* = 6	12	12	14
256b, *Nk* = 8	14	14	14

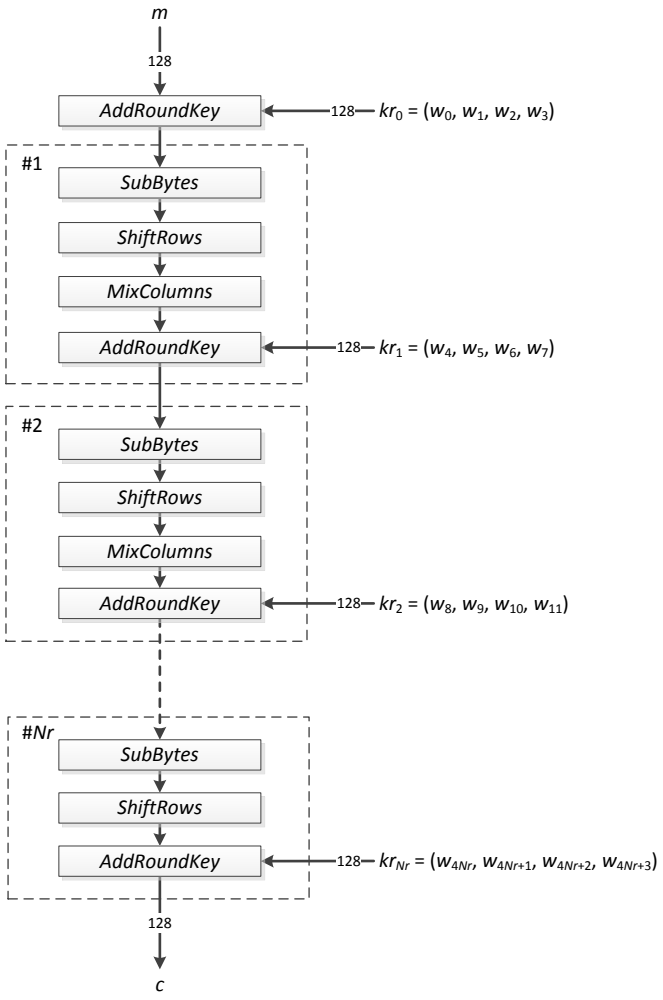
Szyfr Rijndael to zwycięzca konkursu na standard AES (ang. *Advanced Encryption Standard*) i obecny standard kryptograficzny (FIPS 197 [52]), zastępujący od roku 2001 standard DES. Jako wersję obowiązującą, w ramach standardu AES wybrano wersję szyfru Rijndael przetwarzającą bloki wyłącznie o długości 128 bitów, z kluczem 128-, 192- lub

256-bitowym. W szyfrze AES zatem, parametry szyfru Rijndael ograniczono do wartości: *Nb* = 4, *Nk* = 4, 6 lub 8, *Nr* = 10, 12 lub 14 (tab. 3.23).

Tab. 3.23. AES – wartości parametrów *Nb, Nk, Nr*

wariant szyfru	parametry		
	Nb	***Nk***	***Nr***
AES-128	4	4	10
AES-192	4	6	12
AES-256	4	8	14

Na rysunku (rys. 3.24) przedstawiono strukturę funkcji szyfrującej szyfru AES. W rundzie *rd* (*rd* = 1, 2, …, *Nr*) wykorzystywany jest jeden klucz rundowy, kr_{rd}, o długości 128 bitów, traktowany jako ciąg czterech 32-bitowych podkluczy (w_{4rd}, w_{4rd+1}, w_{4rd+2}, w_{4rd+3}), nazywanych słowami (ang. *word*). Ponadto, w transformacji początkowej, wykorzystywany jest 128-bitowy klucz $kr_0 = (w_0, w_1, w_2, w_3)$. W sumie więc, wykorzystywane są $4(Nr + 1)$ podklucze w_i ($i = 0, 1, \ldots, 4(Nr + 1) - 1$), o długości 32 bity.



Rys. 3.24. AES – struktura funkcji szyfrującej

Podczas deszyfrowania stosowana jest funkcja deszyfrująca, z odwrotnymi przekształceniami, wykonywanymi w odwrotnej kolejności. Kolejność kluczy rundowych jest zatem też odwrotna, w porównaniu z ich kolejnością w funkcji szyfrującej.

Niech dla stanu s, stan s' będzie stanem następnym i niech r ($0 \leq r < 4$) oznacza numer wiersza, a c ($0 \leq c < 4$) – numer kolumny tablicy stanu. W funkcji szyfrującej szyfru AES wykonywane są następujące operacje:

- *AddRoundKey* – funkcja dodania, operacją XOR, klucza rundowego do stanu s:

  $$s_{r,c}' = s_{r,c} \oplus w_{r,4rd+c},$$

  gdzie rd ($0 \leq rd \leq Nr$) oznacza numer rundy, a $w_i = (w_{0,i}, w_{1,i}, w_{2,i}, w_{3,i})$ jest podkluczem, złożonym z 4 bajtów.
- *SubBytes* – funkcja podstawień w stanie s, z wykorzystaniem S-bloku S, o rozmiarze 8 × 8 bitów (S-blok identyczny w każdej rundzie szyfru), wykonywana niezależnie na każdym z bajtów stanu s:

  $$s_{r,c}' = S(s_{r,c}).$$
- *ShiftRows* – funkcja przesunięcia cyklicznego w lewo wierszy stanu s, o numerach r równych 1, 2 i 3, odpowiednio o 1, 2 oraz 3 bajty:

  $$s_{r,c}' = s_{r,(c+r) \bmod 4}.$$
- *MixColumns* – funkcja modyfikacji kolumny stanu s, przy czym nowa wartość kolumny c obliczana jest na podstawie bieżącej wartości kolumny c:

  $$s_{r,c}' = xtime(s_{r,c}) \oplus xtime(s_{(r+1) \bmod 4,c}) \oplus s_{(r+1) \bmod 4,c} \oplus s_{(r+2) \bmod 4,c} \oplus s_{(r+3) \bmod 4,c},$$

  gdzie

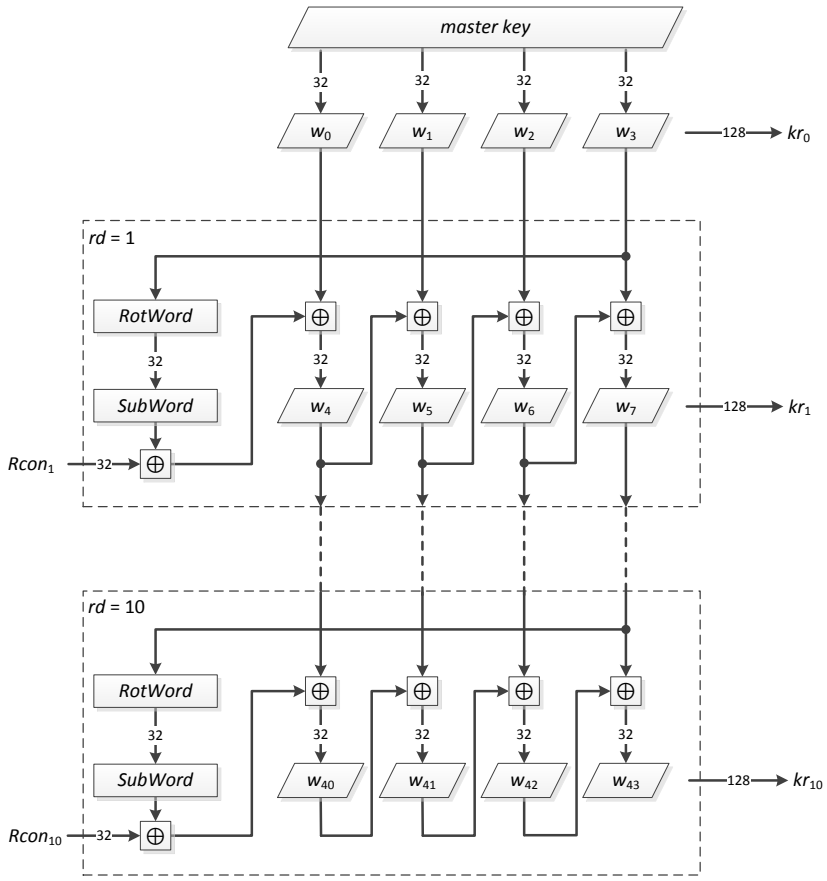
  $$xtime(byte) = \begin{cases} 2 \cdot byte & \text{, dla } byte < 2^7 \\ (2 \cdot byte) \bmod 2^8 \oplus \text{0x1B}, & \text{dla } byte \geq 2^7. \end{cases}$$

Należy nadmienić, że S-blok S o rozmiarze 8 × 8 bitów, wykorzystywany w operacji *SubBytes*, konstruowany jest z wykorzystaniem reguł matematycznych. Najpierw, dla niezerowego argumentu x obliczany jest element odwrotny x^{-1}, ze względu na mnożenie w $GF(2^8)$ oraz element zero przekształcany jest na zero ($0^{-1} = 0$), a następnie stosowana jest określona transformacja afiniczna nad $GF(2)$, w odniesieniu do x^{-1}.

Warto również zauważyć, że warstwa dyfuzji, realizowana z wykorzystaniem operacji *ShiftRows* i *MixColumns*, jest bardziej złożona od zwykłej permutacji bitowej, ale zapewnia szybszą dyfuzję i mniejszą liczbę niezbędnych rund. Jej konstrukcja oparta jest również na zaawansowanym aparacie matematycznym.

Generowanie kluczy rundowych w szyfrze AES, polega na poddaniu klucza głównego operacji rozszerzenia, nazwanej *KeyExpansion*, do długości równej $Nb(Nr + 1) \cdot 32$ bity, w rezultacie czego otrzymuje się:

- dla AES-128: 1408-bitowy klucz rozszerzony,
- dla AES-192: 1664-bitowy klucz rozszerzony,
- dla AES-256: 1920-bitowy klucz rozszerzony.



Rys. 3.25. AES-128 – funkcja generowania kluczy rundowych

W algorytmie generowania kluczy rundowych (algorytm 3.8) i w funkcji przedstawionej na rysunku (rys. 3.25), wykonywane są operacje:

- $\oplus$ – suma wyłączająca (XOR) słów 32-bitowych,
- *RotWord* – funkcja przesunięcia cyklicznego w lewo słowa $w_i = (w_{0,i}, w_{1,i}, w_{2,i}, w_{3,i})$ o 1 bajt:

$$w_{r,i}' = w_{(r+1) \bmod 4,i}, \text{ dla } 0 \leq r < 4.$$

- *SubWord* – funkcja podstawień w słowie w_i, z wykorzystaniem S-bloku S, o rozmiarze 8 × 8 bitów:

$$w_{r,i}' = S(w_{r,i}).$$

Algorytm 3.8. Algorytm generowania kluczy rundowych szyfru AES

WEJŚCIE: klucz główny $k = (k_0, k_1, \ldots, k_{Nk-1})$, o długości: 128, 196 lub 256 bitów

WYJŚCIE: 11, 13 lub 15 kluczy rundowych $kr_0, kr_1, \ldots, kr_{Nr}$, o długości 128 bitów

PROCEDURA:

1. Podstaw: $(w_0, w_1, \ldots, w_{Nk-1}) = (k_0, k_1, \ldots, k_{Nk-1})$.
2. Dla $i = Nk$ do $Nb \cdot (Nr + 1) - 1$ wykonaj operacje:
   - jeśli i jest wielokrotnością Nk, tj. i **mod** $Nk = 0$, to:
     $$w_i = w_{i-Nk} \oplus SubWord(RotWord(w_{i-1})) \oplus Rcon_{i/Nk},$$
   - jeśli $i - 4$ jest wielokrotnością Nk oraz $Nk = 8$ (wariant AES-256), tj. i **mod** $Nk = 4$ i $Nk = 8$, to:
     $$w_i = w_{i-Nk} \oplus SubWord(w_{i-1}),$$
   - w pozostałych przypadkach:
     $$w_i = w_{i-Nk} \oplus w_{i-1}.$$
3. Dla $rd = 0$ do Nr podstaw:
   - $kr_{rd} = (w_{4rd}, w_{4rd+1}, w_{4rd+2}, w_{4rd+3})$.

Pierwsze Nk słów klucza rozszerzonego powstaje bezpośrednio z klucza głównego, każde następne słowo w_i (dla $i = Nk, Nk + 1, \ldots, Nb \cdot (Nr + 1) - 1$) powstaje przez dodanie, operacją XOR, do słowa Nk pozycji wcześniej, tj. w_{i-Nk}, słowa poprzedniego, tj. w_{i-1}, bezpośrednio lub po przekształceniu. Dla słów na pozycjach będących wielokrotnością wartości Nk, najpierw słowo w_{i-1} poddawane jest transformacji oraz dodawana jest stała rundowa $Rcon_{i/Nk}$ (tab. 3.24). Transformacja ta składa się z funkcji przesunięcia cyklicznego bajtów, *RotWord* oraz z funkcji podstawień, z wykorzystaniem S-bloku *S*, *SubWord*. W przypadku szyfru AES-256, jeśli dla iteracji i wartość i–4 jest wielokrotnością Nk to w_i obliczane jest jako XOR słowa w_{i-Nk} i transformacji słowa w_{i-1} ograniczonej do funkcji *SubWord* (algorytm 3.8).

Tab. 3.24. AES – tablica stałych $Rcon_i$ dla $i = 1, 2, \ldots, 10$

$Rcon_1$	0x01000000	$Rcon_2$	0x02000000
$Rcon_3$	0x04000000	$Rcon_4$	0x08000000
$Rcon_5$	0x10000000	$Rcon_6$	0x20000000
$Rcon_7$	0x40000000	$Rcon_8$	0x80000000
$Rcon_9$	0x1B000000	$Rcon_{10}$	0x36000000

W tabeli (tab. 3.24) przedstawiono stałe $Rcon_i$ dla i = 1, 2, …, 10. Stała $Rcon_i$ = (rc_i, 0x00, 0x00, 0x00), gdzie rc_1 = 1 oraz rc_i = $xtime(rc_{i-1})$, dla i > 1. Można więc obliczyć stałą $Rcon_i$ dla dowolnego i, a w zakresie i od 1 do 255 uzyskiwane są różne jej wartości.

Tab. 3.25. AES – zestawienie cech szyfru

rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego |*k*|	liczba kluczy rundowych *nrk*	długość klucza rundowego |*rk*|	długość próbki kluczy rundowych |*rkp*|
128b	10	128b	11	128b	1408b
128b	12	192b	13	128b	1664b
128b	14	256b	15	128b	1920b

W tabeli (tab. 3.25) przedstawiono zestawienie cech szyfru AES-128, AES-192 oraz AES-256. Zastosowano przyjęte w rozprawie oznaczenie r dla liczby rund (r = Nr) i oznaczenie rk dla klucza rundowego (rk = kr).

Tab. 3.26. AES – lista operacji

rodzaj	operacja	funkcja szyfrująca	algorytm generowania kluczy rundowych
XOR	$\oplus$	tak	tak
podstawienie	S	tak	tak
rotacja	$\lll b$	tak	tak
przesunięcie	$\ll b$	tak	nie

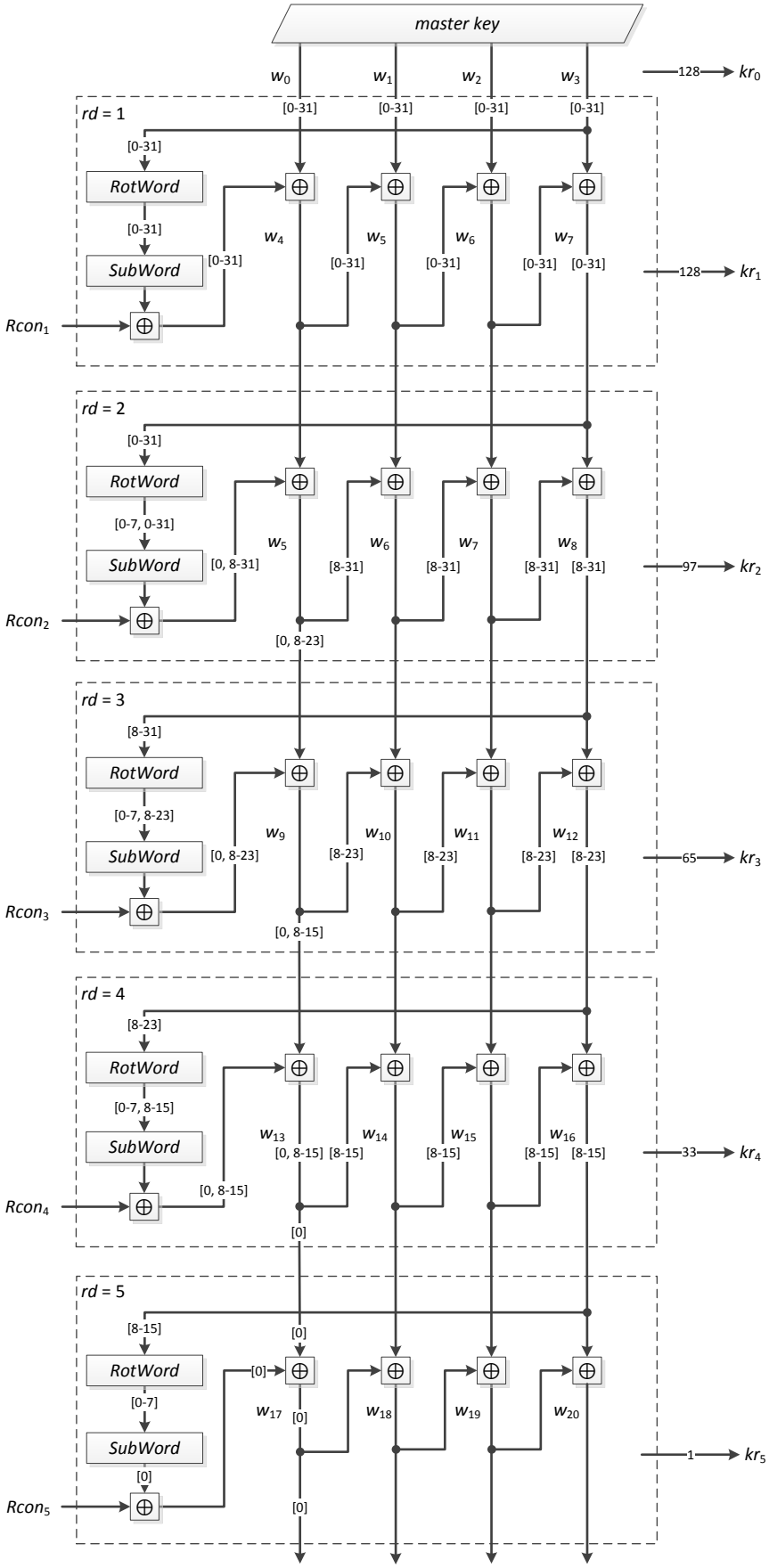
Tabela (tab. 3.26) zawiera listę operacji funkcji szyfrującej i algorytmu generowania kluczy rundowych. Podstawienia realizowane są w funkcjach *SubBytes* i *SubWord*, rotacje – w funkcjach *ShiftRows* i *RotWord*, operacja XOR – w funkcji *AddRoundKey* i bezpośrednio, a w funkcji *MixColumns* realizowane są przesunięcie oraz operacja XOR.

Warto zwrócić uwagę na różnicę w strukturze rundy #*Nr* szyfru, która nie zawiera operacji *MixColumns* oraz fakt, że pierwsze dodanie klucza rundowego kr_0 (złożonego bezpośrednio z bitów klucza głównego) następuje przed pierwszą rundą szyfru.

Wniosek 3.15.

W algorytmie generowania kluczy rundowych szyfru AES wykorzystywane są operacje liniowe i nieliniowe (w szczególności S-blok S o rozmiarze 8 × 8 bitów) oraz wyliczana stała $Rcon_i$. Przez zwiększenie liczby iteracji tego algorytmu możliwe jest zwiększenie liczby rund szyfru AES (metoda B).

■



Rys. 3.26. AES-128 – zależność bitu $kr_5[0]$ klucza rundowego od bitów klucza głównego k

Na rysunku (rys. 3.26) przedstawiono zależność bitu $kr_5[0]$ klucza rundowego od bitów klucza głównego k. Przy określeniu tej zależności uwzględniono następujące własności funkcji elementarnych:

- $\oplus$ – bit i wyjścia (wartości) funkcji XOR zależy od bitu i wszystkich wejść (argumentów),
- $\lll b$ – bit i wyjścia rotacji w lewo o b bitów słowa l-bitowego, w przypadku LSB = 0 po lewej, zależy od bitu $(i + b)$ **mod** l wejścia,
- S – bit i wyjścia S-bloku S zależy od wszystkich bitów wejścia.

Ostatecznie bit $kr_5[0]$ zależy od bitów $w_0[0\text{-}31]$, $w_1[0\text{-}31]$, $w_2[0\text{-}31]$ oraz $w_3[0\text{-}31]$, a ponieważ $(w_0, w_1, w_2, w_3) = (k_0, k_1, k_2, k_3) = k$ to bit $kr_5[0]$ zależy od wszystkich bitów 128-bitowego klucza głównego k, co stanowi 100%.

Z rysunku (rys. 3.26) można też odczytać, że:

- bit $kr_4[0]$ zależy od 128 bitów,
- bit $kr_3[0]$ zależy od 97 bitów,
- bit $kr_2[0]$ zależy od 65 bitów,
- bit $kr_1[0]$ zależy od 33 bitów,
- bit $kr_0[0]$ zależy od 1 bitu.

Te same wyniki uzyskiwane są dla dowolnego bitu kluczy rundowych $kr_0, kr_1, \ldots, kr_5$.

Wniosek 3.16.

W szyfrze AES-128 klucz rundowy kr_i (i = 0, 1, …, Nr) zależy od wszystkich bitów 128-bitowego klucza głównego k, przy czym każdy bit klucza kr_i, zależy od liczby bitów klucza k: 1 (i = 0), 33 (i = 1), 65 (i = 2), 97 (i = 3) lub 128 (i = 4,5, …, Nr).

■

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru AES, przedstawiono w postaci charakterystyki (charakterystyka 3.8).

Charakterystyka 3.8. Charakterystyka algorytmu generowania kluczy rundowych szyfru AES

1. Długość klucza głównego: $|\boldsymbol{k}|$ **= 128b, 192b, 256b**.
2. Długość klucza rundowego: $|\boldsymbol{kr_i}|$ **= 128b (**$\boldsymbol{i}$ **= 0, 1, …,** $\boldsymbol{Nr}$**)**.
3. Liczba iteracji (Nb = 4, Nr = 10, 12, 14): $\boldsymbol{R = Nb \cdot (Nr + 1) - Nk}$.
4. Zależność kr_i od bitów k (AES-128):

- **128b / 128b (100%)**.

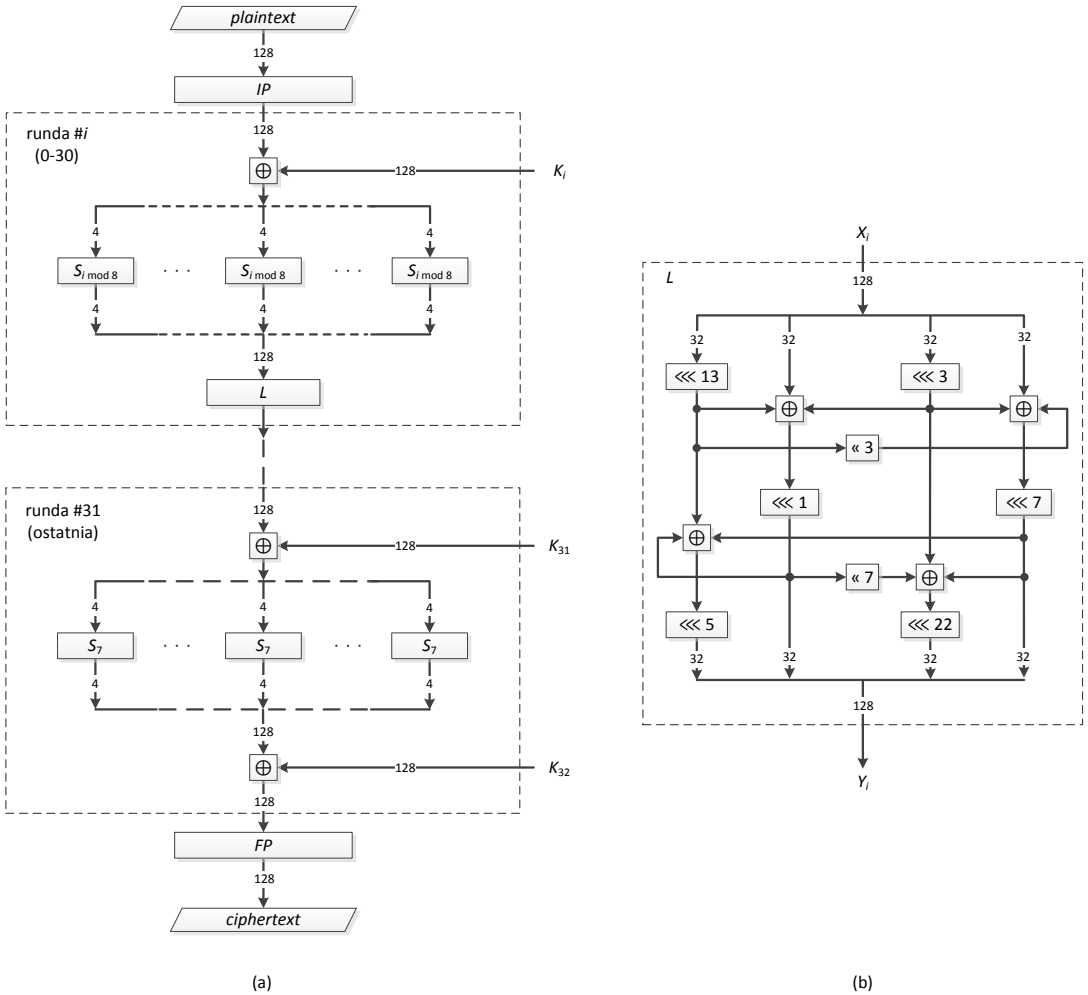
5. Zależność bitu kr_i od bitów k (AES-128):
   - **kr_0: 1b / 128b (0.78%)**,
   - **kr_1: 33b / 128b (25.78%)**,
   - **kr_2: 65b / 128b (50.78%)**,
   - **kr_3: 97b / 128b (75.78%)**,
   - **kr_4, kr_5, …, kr_{Nr}: 128b / 128b (100.00%)**.
6. Zależność grupy bitów kr_i od grupy bitów k: **NIE**.
7. Operacja liniowe: **$\oplus$, *RotWord***.
8. Operacje nieliniowe: ***SubWord* – *S* (8b × 8b)**.
9. Możliwości zwiększenia liczby iteracji: **TAK**.
10. Indywidualizacja iteracji: **przez stałe $Rcon_i$**.
11. Teoretyczna ocena jakości algorytmu: **50.63%**.

3.3.2. Serpent (1998)

Szyfr Serpent [1] został zaprojektowany w roku 1998 przez R. Andersona, E. Bihama oraz L. Knudsena i był jednym z pięciu finalistów w konkursie na standard AES. Ostatecznie Serpent zajął drugie miejsce w konkursie, pomimo że został uznany za bezpieczniejszy od szyfru Rijndael. Zadecydowało to, że jest znacznie wolniejszy, ze względu na zastosowaną liczbę 32 rund. Żaden z obecnie znanych ataków na szyfr Serpent nie jest wykonywany w akceptowalnym czasie [20][21][51][88], a realne ataki dotyczą wyłącznie wersji szyfru, zredukowanej do 10 i 11 rund.

Szyfr Serpent przetwarza 128-bitowe bloki danych w r = 32 rundach. Klucz główny k może przyjmować długość 128, 192 lub 256 bitów. Gdy klucz główny jest krótszy niż 256 bitów, to zostaje uzupełniony do długości 256 bitów, poprzez wstawienie na najbardziej znaczących pozycjach ciągu 10…0, o odpowiedniej liczbie zer. Z 256-bitowego klucza głównego, generowane są trzydzieści trzy klucze rundowe K_i, o długości 128 bitów.

Podczas deszyfrowania stosowane są funkcje odwrotne i wykonywane są one w odwrotnej kolejności, co implikuje odwrotną kolejność kluczy rundowych.

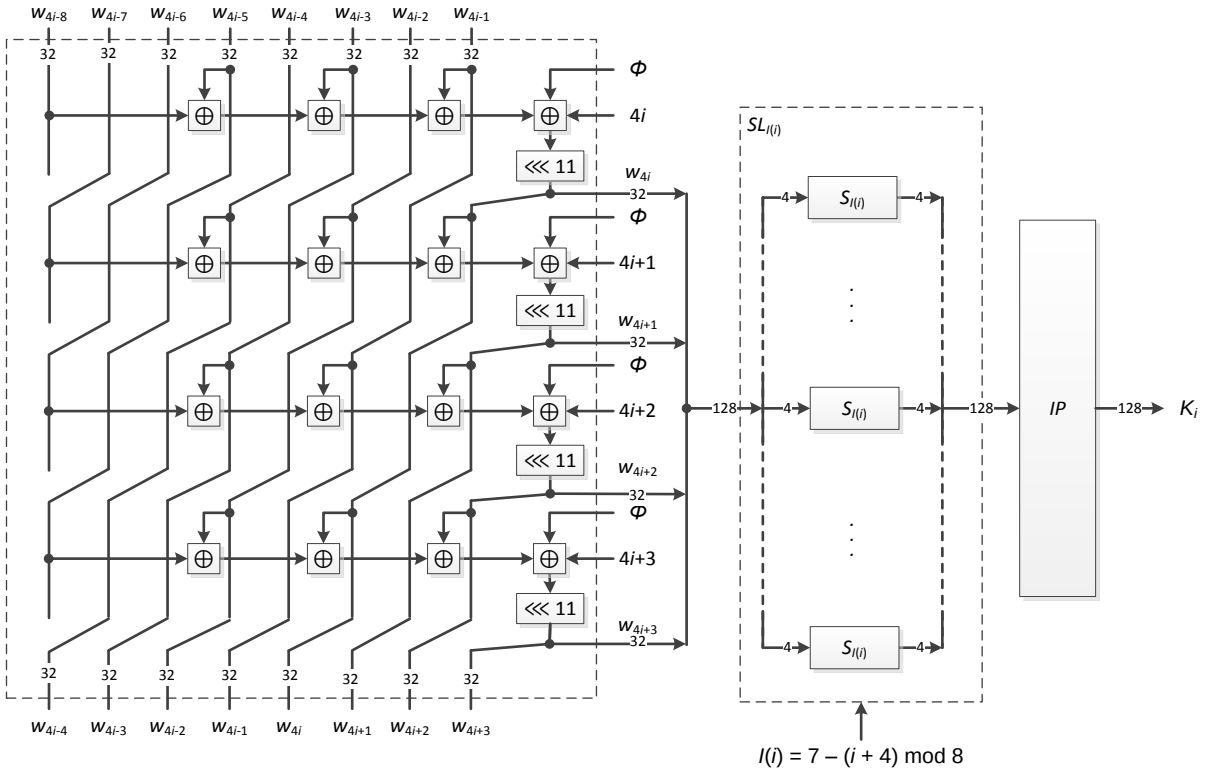


Rys. 3.27. Serpent – struktura funkcji: (a) szyfrującej, (b) liniowej *L*

W funkcji szyfrującej, przedstawionej na rysunku (rys. 3.27), wykonywane są następujące operacje:

- *IP* – permutacja początkowa (ang. *initial permutation*),
- $\oplus$ – suma wyłączająca (XOR) słów 128-bitowych,
- S_0, S_1, …, S_7 – osiem różnych S-bloków, o rozmiarze 4 × 4 bity (podstawienia), przy czym w rundzie i, gdzie $i = 0, 1, \ldots, r - 1$, wykorzystuje się trzydzieści dwa S-bloki $S_{i \bmod 8}$,
- *L* – funkcja liniowa, złożona z operacji:
  - $\lll b$ – cykliczne przesunięcie (rotacja) w lewo słowa 32-bitowego o b bitów, gdzie $b = 1, 3, 5, 7, 13, 22$,
  - $\oplus$ – suma wyłączająca (XOR) słów 32-bitowych,
  - $\ll b$ – przesunięcie w lewo słowa 32-bitowego o b bitów, gdzie $b = 3, 7$,
- *FP* – permutacja końcowa (ang. *final permutation*).

Należy zwrócić uwagę na różnicę w konstrukcji ostatniej rundy, względem pozostałych rund, w której funkcja L zastąpiona została operacją XOR z kluczem rundowym K_{32}.



Rys. 3.28. Serpent – struktura funkcji generowania klucza rundowego K_i (i = 0, 1,…, 32)

W algorytmie generowania kluczy rundowych (algorytm 3.9) i w funkcji przedstawionej na rysunku (rys. 3.28) wykonywane są następujące operacje:

- $\oplus$ – suma wyłączająca (XOR) słów 32-bitowych,
- $\lll$ 11 – cykliczne przesunięcie (rotacja) w lewo słowa 32-bitowego o 11 bitów
- $SL_{I(i)}$ – warstwa 32 równolegle połączonych S-bloków $S_{I(i)}$, o rozmiarze 4 × 4 bity,
- $I(i)$ – funkcja obliczania indeksu S-bloków, stosowanych do generowania klucza rundowego K_i taka, że $I(i) = 7 – (i + 4)$ **mod** 8,
- IP – permutacja początkowa.

Algorytm 3.9. Algorytm generowania kluczy rundowych szyfru Serpent

WEJŚCIE: klucz główny k o długości 128, 192 lub 256 bitów

WYJŚCIE: trzydzieści trzy klucze rundowe K_i o długości 128 bitów

PROCEDURA:

1. Uzupełnij klucz główny k do długości 256 bitów:
   - jeśli $|k|$ = 128b to $k = 1||0^{127}||k$,

- jeśli $|k| = 192$b to $k = 1||0^{63}||k$.

2. Podstaw: $w_{-8}||w_{-7}||...||w_{-1} = k$.
3. Dla $i = 0$ do 131 oblicz:
   - $w_i = (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus \Phi \oplus i) \lll 11$,

   gdzie stała Φ = 0x9E3779B9 (ang. „*golden ratio*”).
4. Dla $i = 0$ do 32 oblicz:
   - $K_i = IP(SL_{I(i)}(w_{4i}||w_{4i+1}||w_{4i+2}||w_{4i+3}))$,

   gdzie $I(i) = 7 - (i + 4)$ **mod** 8, a $SL_{I(i)}$ jest warstwą 32 równolegle połączonych S-bloków $S_{I(i)}$, o rozmiarze 4 × 4 bity.

Warto zauważyć, że zastosowanie w algorytmie (algorytm 3.9) stałej Φ, o wartości 0x9E3779B9 związanej ze „złotym podziałem”, chroni przed sytuacją wyzerowania wartości podklucza w_i.

Tab. 3.27. Serpent – zestawienie cech szyfru

rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego *|k|*	liczba kluczy rundowych *nrk*	długość klucza rundowego *|rk|*	długość próbki kluczy rundowych *|rkp|*
128b	32	128b	33	128b	4224b
		192b			
		256b			

W tabeli (tab. 3.27) przedstawiono zestawienie podstawowych cech szyfru Serpent.

Tab. 3.28. Serpent – lista operacji

rodzaj	operacja	funkcja szyfrująca	algorytm generowania kluczy rundowych
permutacja początkowa	*IP*	tak	tak
XOR	$\oplus$	tak	tak
podstawienie	$S_0, S_1, \ldots, S_7$	tak	tak
rotacja	$\lll b$	tak	tak
przesuniecie	$\ll b$	tak	nie
permutacja końcowa	*FP*	tak	nie

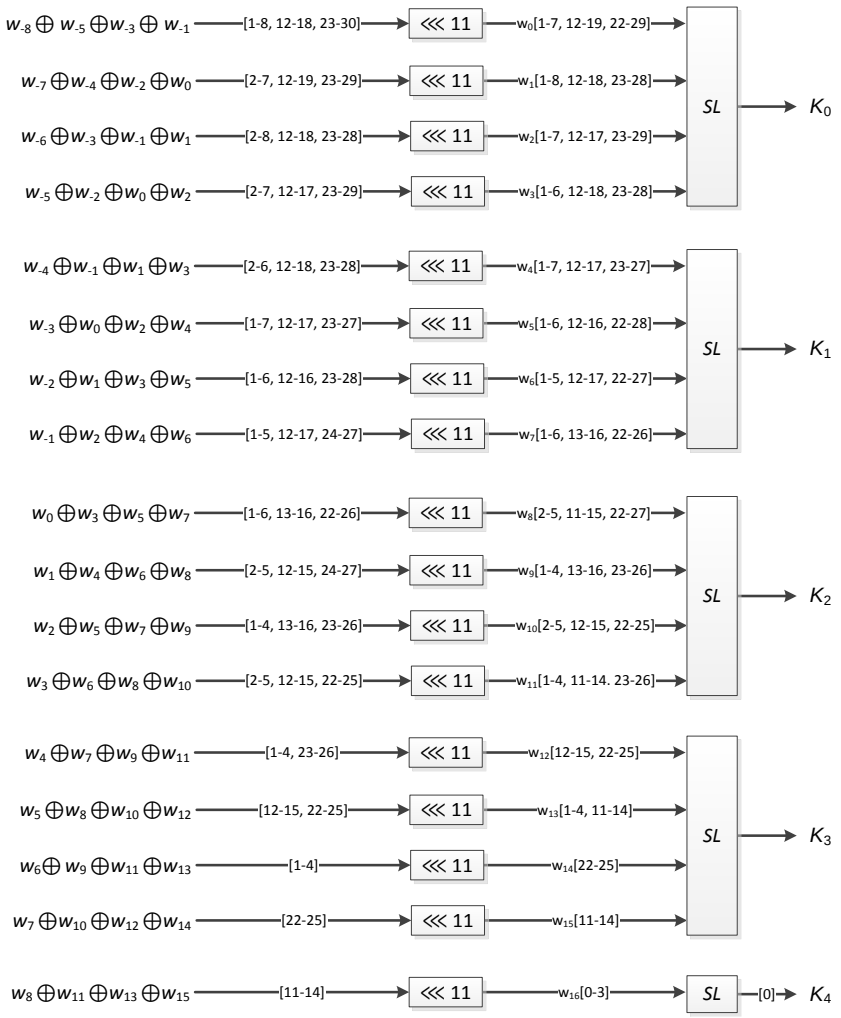
W tabeli (tab. 3.28) przedstawiono listę operacji funkcji szyfrującej i algorytmu generowania kluczy rundowych.

Wniosek 3.17.

W algorytmie generowania kluczy rundowych szyfru Serpent wykorzystywane są operacje nieliniowe – S-bloki $S_0, S_1, \ldots, S_7$ oraz operacje liniowe – XOR, rotacja, permutacja

bitowa. Możliwe jest zwiększenia liczby iteracji tego algorytmu (metoda B) i możliwe jest odpowiednie zwiększenie liczby rund szyfru Serpent.

■



Rys. 3.29. Serpent – zależność bitu $K_4[0]$ klucza rundowego od bitów klucza głównego $k = w_{-8}||w_{-7}||\ldots||w_{-1}$

Na rysunku (rys. 3.29) przedstawiono zależność bitu $K_4[0]$ klucza rundowego od bitów klucza głównego $k = w_{-8}||w_{-7}||...||w_{-1}$. Permutację bitową *IP*, stałą Φ oraz licznik iteracji *i*, jako nieistotne dla rozważań pominięto. Przy określeniu tej zależności uwzględniono następujące własności funkcji elementarnych:

- $\oplus$ – bit *i* wyjścia (wartości) funkcji XOR zależy od bitu *i* wszystkich wejść (argumentów),
- $\lll b$ – bit *i* wyjścia rotacji w lewo o *b* bitów słowa *l*-bitowego, w przypadku LSB = 0 po lewej, zależy od bitu $(i + b)$ **mod** *l* wejścia,
- $S_0, S_1, \ldots, S_7$ – bit *i* wyjścia S-bloku *S* zależy od wszystkich bitów wejścia.

Ostatecznie bit $K_4[0]$ zależy od bitów $w_{-8}[a]$, $w_{-7}[b]$, $w_{-6}[c]$, $w_{-5}[a]$, $w_{-4}[b]$, $w_{-3}[a]$, $w_{-2}[b]$ oraz $w_{-1}[a]$, gdzie $[a]$ = [1-8, 12-18, 23-30], $[b]$ = [2-7, 12-19, 23-29] i $[c]$ = [2-8, 12-18, 23-28]. Ponieważ $k = w_{-8}||w_{-7}||...||w_{-1}$ to bit $K_4[0]$ zależy od 175 bitów 256-bitowego klucza głównego *k*, co stanowi 68.36%. Z rysunku (rys. 3.29) można też odczytać, że:

- bit $K_3[0]$ zależy od 143 bitów,
- bit $K_2[0]$ zależy od 108 bitów,
- bit $K_1[0]$ zależy od 60 bitów,
- bit $K_0[0]$ zależy od 16 bitów.

Bit numer 0 kluczy rundowych K_0, K_1, …, K_4 stanowi najgorszy przypadek, co oznacza, że pozostałe bity (1, 2, …, 127) zależą od tej samej lub większej liczby bitów klucza głównego *k*.

Wniosek 3.18.

W szyfrze Serpent-256 klucz rundowy K_i (i = 0, 1, …, 32) zależy od wszystkich bitów 256-bitowego klucza głównego *k*, przy czym bit klucza K_i, w najgorszym przypadku, zależy od liczby bitów klucza *k*: 16 (i = 0), 60 (i = 1), 108 (i = 2), 143 (i = 3), 175 (i = 4).

■

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru Serpent, przedstawiono w postaci charakterystyki (charakterystyka 3.9).

Charakterystyka 3.9. Charakterystyka algorytmu generowania kluczy rundowych szyfru Serpent

1. Długość klucza głównego: $|\boldsymbol{k}|$ **= 128b, 192b, 256b**.
2. Długość klucza rundowego: $|\boldsymbol{K_i}|$ **= 128b (*i* = 0, 1, …, 32)**.
3. Liczba iteracji: $\boldsymbol{R}$ **= 132**.
4. Zależność K_i od bitów *k* (Serpent-256):
   - **256b / 256b (100%)**.
5. Zależność bitu K_i od bitów *k* (Serpent-256):
   - $\boldsymbol{K_0}$**[0]: 16b / 256b (6.25%)**,
   - $\boldsymbol{K_1}$**[0]: 60b / 256b (23.44%)**,
   - $\boldsymbol{K_2}$**[0]: 108b / 256b (42.19%)**,
   - $\boldsymbol{K_3}$**[0]: 143b / 256b (55.86%)**,
   - $\boldsymbol{K_4}$**[0]: 175b / 256b (68.36%)**.
6. Zależność grupy bitów K_i od grupy bitów *k*: **NIE**.

7. Operacja liniowe: ***IP, ⊕, ⋘b***.
8. Operacje nieliniowe: $\boldsymbol{S_0, S_1, \ldots, S_7}$ **(4b × 4b)**.
9. Możliwości zwiększenia liczby iteracji: **TAK**.
10. Indywidualizacja iteracji: **przez licznik *i*, $\boldsymbol{\Phi \oplus i}$.**
11. Teoretyczna ocena jakości algorytmu: **39.22%**.

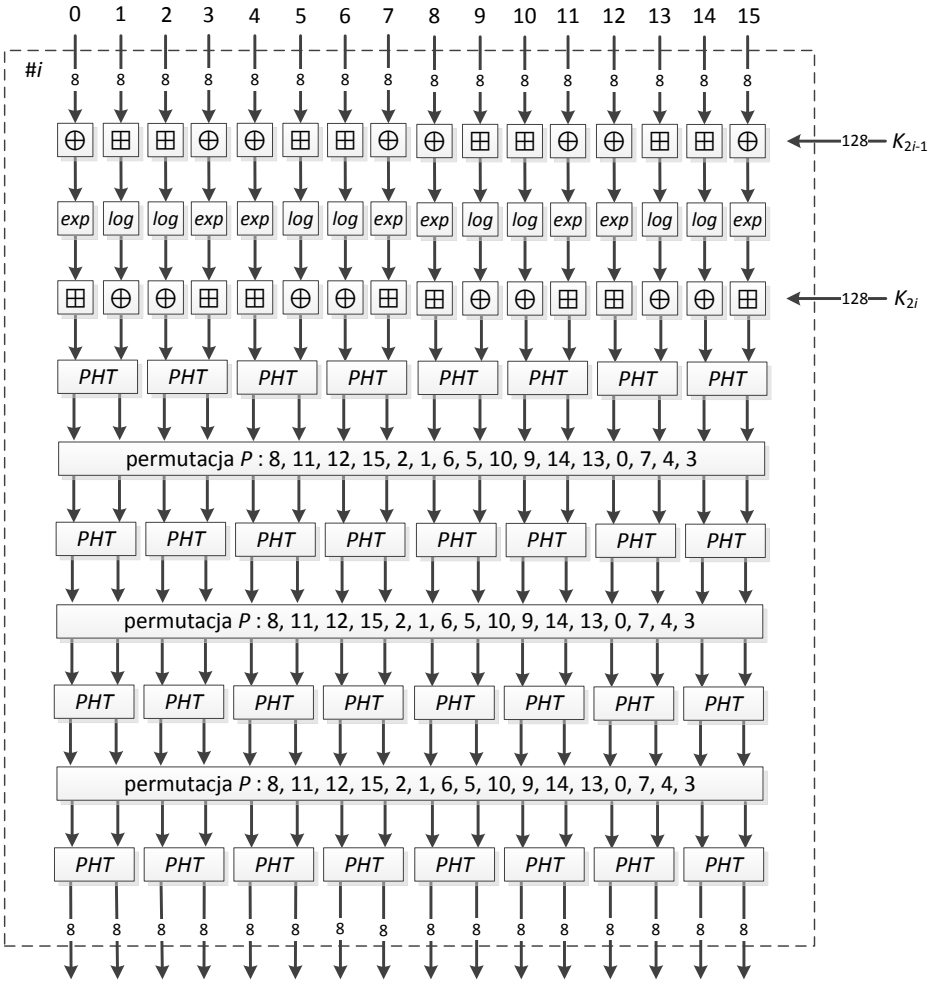
3.3.3. SAFER+ (1998)

Szyfr SAFER+ (ang. *Secure And Fast Encryption Routine+*) [83], zakwalifikowany do konkursu na standard AES, typu SPN, przetwarza 128-bitowe bloki danych z wykorzystaniem klucza głównego k o długości 128, 192 lub 256 bitów. W każdej rundzie funkcji szyfrującej wykorzystywane są dwa klucze rundowe o długości 128 bitów, a ponadto jeden klucz, o tej samej długości, w transformacji wyjściowej. W sumie więc, podczas szyfrowania, mamy $2r + 1$ kluczy rundowych o długości 128 bitów. Liczba rund szyfru zależy od długości klucza głównego w następujący sposób:

- 8 rund, dla klucza o długości 128 bitów ($r = 8$),
- 12 rund, dla klucza o długości 192 bity ($r = 12$),
- 16 rund, dla klucza o długości 256 bitów ($r = 16$).

Strukturę pojedynczej rundy funkcji szyfrującej, szyfru SAFER+, przedstawiono na rysunku (rys. 3.30). Podczas deszyfrowania wykorzystywane są funkcje odwrotne i wykonywane są one w odwrotnej kolejności – klucze rundowe podawane są też w odwrotnej kolejności. W funkcji szyfrującej wykonywane są następujące operacje:

- $\oplus$ – suma wyłączająca (XOR) słów 8-bitowych (bajtów),
- $\boxplus$ – dodawanie modulo 2^8,
- *exp* – funkcja wykładnicza $exp(x) = 45^x$ **mod** 257, przy dodatkowym założeniu, że $45^{128} = 256$ jest reprezentowane przez 0,
- *log* – funkcja logarytmiczna $log_{45}(x)$, przy dodatkowym założeniu, że $log_{45}(0) = 128$,
- *PHT* – odwracalna transformacja (ang. *pseudo-Hadamard transform*), taka że: $PHT(a, b) = ((2a + b)$ **mod** $256, (a + b)$ **mod** $256)$,
- *P* – permutacja bajtów – (8, 11, 12, 15, 2, 1, 6, 5, 10, 9, 14, 13, 0, 7, 4, 3), taka że bajt 0 argumentu permutacji *P* jest bajtem 8 jej wartości,
- *OT* – transformacja wyjściowa (ang. *output transformation*) – po rundzie *r*.

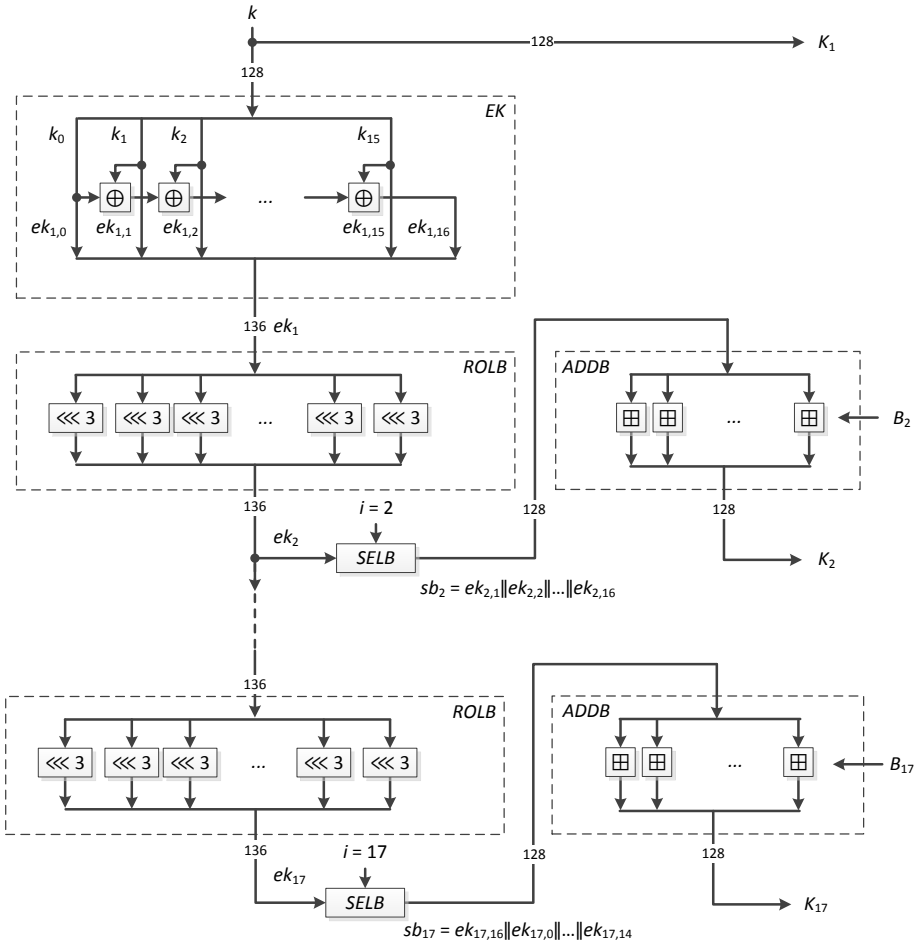


Rys. 3.30. SAFER+ – struktura rundy #*i* funkcji szyfrującej (*i* = 1, 2, …, *r*)

Należy zauważyć, że operacje *exp* i *log* stanowią definicję konstrukcji odpowiadających im S-bloków, S_{exp} i S_{log}, o rozmiarze 8 × 8 bitów.

W algorytmie generowania kluczy rundowych, w wersji szyfru z kluczem 128-bitowym (algorytm 3.10) i funkcji przedstawionej na rysunku (rys. 3.31), wykonywane są następujące operacje:

- *EK* – funkcja rozszerzenia klucza głównego *k* o bajt parzystości, będący sumą wyłączającą (XOR) wszystkich bajtów klucza głównego *k*,
- *ROLB*(*ek*, 3) – cykliczne przesunięcie (rotacja) w lewo, każdego z bajtów rozszerzonego klucza *ek*, o 3 bity,
- ⊞ – dodawanie modulo 256, wybranych bajtów rozszerzonego klucza *ek* ze stałą B_i, gdzie $i = 1, 2, \ldots, 2r + 1$.



Rys. 3.31. SAFER+ – struktura funkcji generowania kluczy rundowych z kluczem 128-bitowym

Algorytm 3.10. Algorytm generowania kluczy rundowych szyfru SAFER+ z kluczem 128-bitowym (SAFER+ 128/128)

WEJŚCIE: klucz główny $k = k_0||k_1||\ldots||k_{15}$ o długości 128 bitów

WYJŚCIE: $2r + 1 = 17$ kluczy rundowych $K_i = K_{i,0}||K_{i,1}||\ldots||K_{i,15}$ o długości 128 bitów

PROCEDURA:

1. Zdefiniuj funkcję $I(i, j) = ((i - 1) + j)$ **mod** 17.
2. Podstaw $K_1 = k$.
3. Oblicz klucz rozszerzony $ek = EK(k) = k||k_{16}$, gdzie $k_{16} = k_0 \oplus k_1 \oplus \ldots \oplus k_{15}$.
4. Dla iteracji $i = 2$ do $2r + 1 = 17$ wykonaj:
   - $ek = ROLB(ek, 3) = (ek_0 \lll 3)||(ek_1 \lll 3)||\ldots||(ek_{16} \lll 3)$,
   - $sb = SELB(ek, i) = ek_{I(i,0)}||ek_{I(i,1)}||\ldots||ek_{I(i,15)}$,
   - $K_i = ADDB(sb, B_i) = (sb_0 \boxplus B_{i,0})||\ (sb_1 \boxplus B_{i,1})||\ldots||(sb_{15} \boxplus B_{i,15})$, gdzie stała B_i obliczana jest następująco:

$$B_{i,j} = (45^{(45^{(17i+j)} \bmod 257)}) \bmod 257, \text{ dla } j = 0,1,\ldots,15.$$

W pracy [67] zwrócono uwagę na słabość algorytmu generowania kluczy rundowych, w postaci słabej dyfuzji bitów klucza głównego, szczególnie w wersji z dłuższym kluczem głównym niż 128 bitów oraz podatność na ataki z powiązanymi kluczami (ang. *releted-key attack*).

Tab. 3.29. SAFER+ – zestawienie cech szyfru

rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego |*k*|	liczba kluczy rundowych *nrk*	długość klucza rundowego |*rk*|	długość próbki kluczy rundowych |*rkp*|
128b	8	128b	$2r + 1$	128b	2176b
	12	192b			3200b
	16	256b			4224b

W tabeli (tab. 3.29) przedstawiono zestawienie podstawowych cech szyfru SAFER+.

Tab. 3.30. SAFER+ – lista operacji

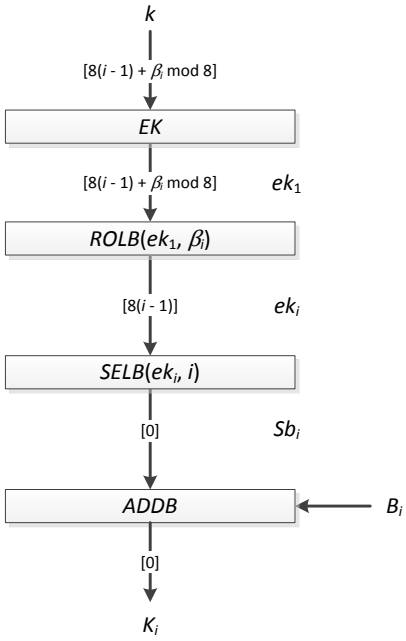
rodzaj	operacja	funkcja szyfrująca	algorytm generowania kluczy rundowych
XOR	$\oplus$	tak	tak
dodawanie modulo 2^8	$\boxplus$	tak	tak
podstawienie	*exp*, *log*	tak	nie
transformacja	*PHT*	tak	nie
permutacja bajtów	*P*	tak	nie
rozszerzenie	*EK*	nie	tak
rotacja w lewo	$\lll b$	nie	tak

W tabeli (tab. 3.30) przedstawiono listę operacji funkcji szyfrującej i algorytmu generowania kluczy rundowych. Operacje $\boxplus$, *exp*, *log* oraz *PHT* są nieliniowe, a pozostałe operacje są liniowe (formuła (3.17)). Stała B_i, wykorzystywana w algorytmie generowania kluczy rundowych, jest wyliczana, zatem możliwe jest zwiększenie liczby iteracji tego algorytmu.

Wniosek 3.19.

W algorytmie generowania kluczy rundowych szyfru SAFER+ wykorzystywane są zarówno operacje liniowe – suma wyłączająca (XOR), *EK*, rotacja jak i operacja nieliniowa – $\boxplus$. Przez zwiększenie liczby iteracji tego algorytmu możliwe jest zwiększenie liczby rund szyfru SAFER+ (metoda B).

■



Rys. 3.32. SAFER+ 128/128 – zależność bitu $K_i[0]$ klucza rundowego od bitów klucza głównego k

Zdefiniujmy dla iteracji i ($i = 2,3, \ldots, 2r + 1$), sumaryczny parametr rotacji $\beta_i = 3(i - 1)$. Klucz rundowy K_i może być określony formułą:

$$K_i = ADDB(SELB(ROLB(EK(k), \beta_i), i), B_i). \tag{3.20}$$

Na rysunku (rys. 3.32) przedstawiono zależność bitu $K_i[0]$ klucza rundowego od bitów klucza głównego k. Przy określeniu tej zależności uwzględniono następujące własności funkcji elementarnych:

- $\oplus$ – bit i wyjścia (wartości) funkcji XOR zależy od bitu i wszystkich wejść (argumentów),
- $\lll b$ – bit i wyjścia rotacji w lewo o b bitów słowa l-bitowego, w przypadku LSB = 0 po lewej, zależy od bitu $(i + b)$ **mod** l wejścia,
- $\boxplus$ – bit i wyjścia funkcji dodawania modulo 2^l zależy od bitów i, $i - 1, \ldots, 0$ wejść, gdzie bit numer 0 jest najmniej znaczący.

Ostatecznie bit $K_i[0]$, dla $i = 2, 3, \ldots, 16$, zależy od bitu klucza głównego k o numerze $8(i - 1) + \beta_i$ **mod** 8, tj. od bitu β_i **mod** 8 w bajcie o numerze $(i - 1)$. Dla $i = 17$ bit $K_i[0]$ zależy od bitu $k_{16}[\beta_i$ **mod** 8], a ponieważ bajt parzystości $k_{16} = k_0 \oplus k_1 \oplus \ldots \oplus k_{15}$, to zależy od bitu β_i **mod** 8 w 16 bajtach klucza k. Dla $i = 1$, z uwagi na $K_1 = k$, bit $K_1[0]$ zależy od bitu $k[0]$. Bit $K_i[0]$ z uwagi na funkcję *ADD* ($\boxplus$), stanowi najgorszy przypadek, tj. zależy od możliwie najmniejszej liczby bitów klucza głównego k.

Wniosek 3.20.

W szyfrze SAFER+ 128/128 klucz rundowy K_i (i = 1, 2, …, 17) zależy od wszystkich bitów 128-bitowego klucza głównego k, przy czym każdy bit klucza K_i zależy, w najgorszym przypadku, od liczby bitów klucza k: 1 (i = 1, 2, …, 16) lub 16 (i = 17).

■

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru SAFER+, przedstawiono w postaci charakterystyki (charakterystyka 3.10).

Charakterystyka 3.10. Charakterystyka algorytmu generowania kluczy rundowych szyfru SAFER+

1. Długość klucza głównego: $|\boldsymbol{k}|$ **= 128b, 192b, 256b**.
2. Długość klucza rundowego: $|\boldsymbol{K_i}|$ **= 128b (**$\boldsymbol{i}$ **= 1, 2, …, 2**$\boldsymbol{r}$ **+ 1)**.
3. Liczba iteracji (r = 8, 12, 16): $\boldsymbol{R = 2r}$.
4. Zależność K_i od bitów k (SAFER+ 128/128):
   - **128b / 128b (100.00%),**
5. Zależność bitu K_i od bitów k (SAFER+ 128/128):
   - $\boldsymbol{K_1[0], K_2[0], \ldots, K_{16}[0]}$**: 1b / 128b (0.78%),**
   - $\boldsymbol{K_{17}[0]}$**: 16b / 128b (1.25%).**
6. Zależność grupy bitów K_i od grupy bitów k: **8b**.
7. Operacja liniowe: $\oplus$, $\lll \boldsymbol{b}$.
8. Operacje nieliniowe: $\boxplus$.
9. Możliwości zwiększenia liczby iteracji: **TAK**.
10. Indywidualizacja iteracji: **przez stałe** $\boldsymbol{B_i}$.
11. Teoretyczna ocena jakości algorytmu: **0.78%**.

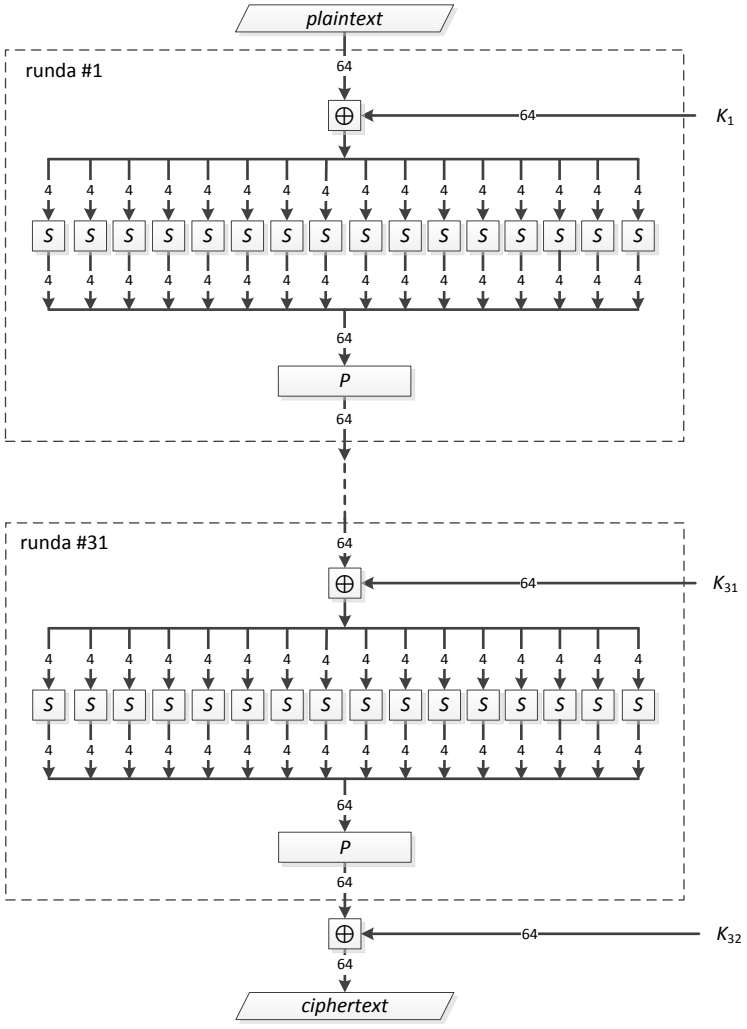
3.3.4. PRESENT (2007)

Szyfr PRESENT [34], opublikowany w roku 2007, należy do grupy tzw. lekkich (ang. *lightweight*) szyfrów blokowych i od roku 2012 jest standardem ISO/IEC 29192-2[5] [62][63]. Szyfr PRESENT przetwarza bloki danych o rozmiarze n = 64 bity, w r = 31 rundach, a klucz główny k może przyjmować długość 80 bitów lub 128 bitów. Zależnie od zastosowanej długości klucza k, szyfr oznaczany jest jako PRESENT-80 lub PRESENT-128. W każdej

[5] Aktualna wersja standardu ISO/IEC 29192-2, z roku 2019, uwzględnia szyfry: PRESENT, CLEFIA [100] oraz LEA [58].

rundzie wykorzystywany jest jeden 64-bitowy klucz rundowy K_i (i = 1, 2, …, 31), a ponadto klucz K_{32}, o tej samej długości, po ostatniej rundzie.

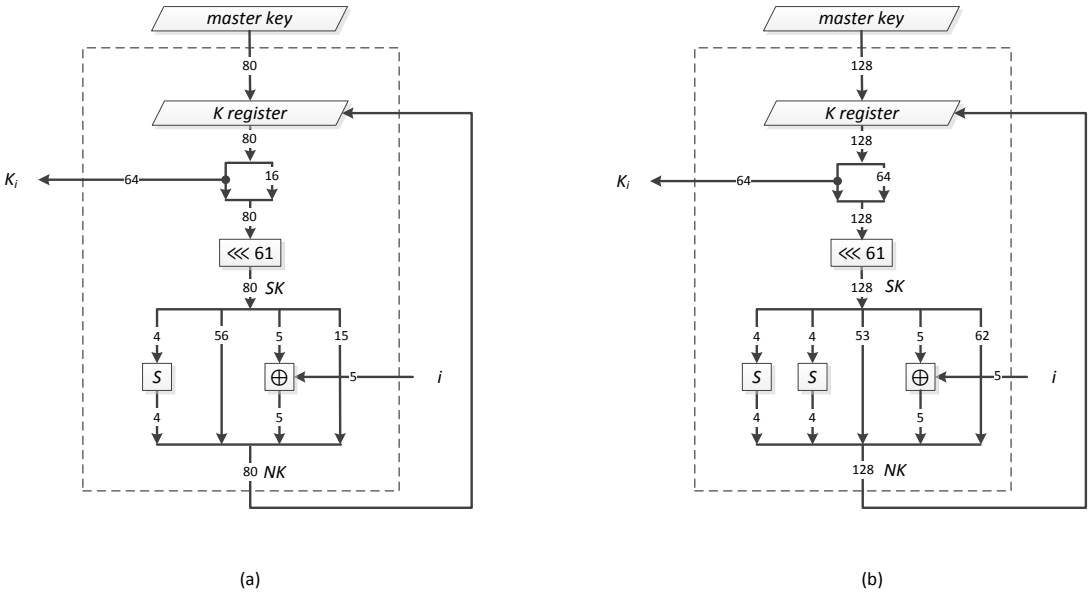
Podczas deszyfrowania stosowana jest funkcja deszyfrująca, zawierająca przekształcenia odwrotne, w której operacje wykonywane są w odwrotnej kolejności, z zastosowaniem również odwrotnej kolejność kluczy rundowych.



Rys. 3.33. PRESENT – struktura funkcji szyfrującej

W funkcji szyfrującej, przedstawionej na rysunku (rys. 3.33), wykonywane są następujące operacje:

- $\oplus$ – suma wyłączająca (XOR) słów 64-bitowych,
- S – funkcja podstawień, S-blok o rozmiarze 4 × 4 bity,
- P – permutacja bitowa.



Rys. 3.34. PRESENT – struktura funkcji generowania kluczy rundowych K_i (i = 1, 2, …, 32), w przypadku: (a) $|k|$ = 80b, (b) $|k|$ = 128b

W algorytmie generowania kluczy rundowych (algorytm 3.11) i w funkcjach przedstawionych na rysunku (rys. 3.34), wykonywane są operacje:

- ⋘ 61 – cykliczne przesunięcie (rotacja) w lewo słowa 80- lub 128-bitowego o 61 bitów,
- S – funkcja podstawień, S-blok o rozmiarze 4 × 4 bity,
- $\oplus$ – suma wyłączająca (XOR) słów 5-bitowych.

Algorytm 3.11. Algorytm generowania kluczy rundowych szyfru PRESENT

WEJŚCIE: klucz główny k o długości 80 lub 128 bitów

WYJŚCIE: klucze rundowe K_1, K_2, …, K_{32} o długości 64 bity

PROCEDURA:

1. Wpisz k do rejestru K, tj. $K = k$.
2. Dla i = 1 do 31 wykonaj:
   - w przypadku
     - $|k| = 80$: $K_i = K[79\text{-}16]$,
     - $|k| = 128$: $K_i = K[127\text{-}64]$,
   - $SK = K \lll 61$,
   - w przypadku:
     - $|k| = 80$: $NK = S(SK[79\text{-}76])||SK[75\text{-}20]||SK[19\text{-}15] \oplus i \,||SK[14\text{-}0]$,

- $|k| = 128$: $NK = S(SK[127\text{-}124])||S(SK[123\text{-}120])||SK[119\text{-}67]||SK[66\text{-}62] \oplus i ||SK[61\text{-}0]$,

▪ wpisz NK do rejestru K, tj. $K = NK$.

3. W przypadku:
   ▪ $|k| = 80$: $K_{32} = K[79\text{-}16]$,
   ▪ $|k| = 128$: $K_{32} = K[127\text{-}64]$.

Warto zauważyć, że pierwszy klucz rundowy pobierany jest, poprzez rejestr K, z klucza głównego, a klucz rundowych K_{32} dodawany jest po ostatniej rundzie funkcji szyfrującej, co nazywane jest wybielaniem–po (ang. *post-whitening*).

Tab. 3.31. PRESENT – zestawienie cech szyfru

rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego *|k|*	liczba kluczy rundowych *nrk*	długość klucza rundowego *|rk|*	długość próbki kluczy rundowych *|rkp|*
64b	31	80b	32	64b	2048b
		128b			

W tabeli (tab. 3.31) przedstawiono zestawienie podstawowych cech szyfrów PRESENT-80 oraz PRESENT-128.

Tab. 3.32. PRESENT – lista operacji

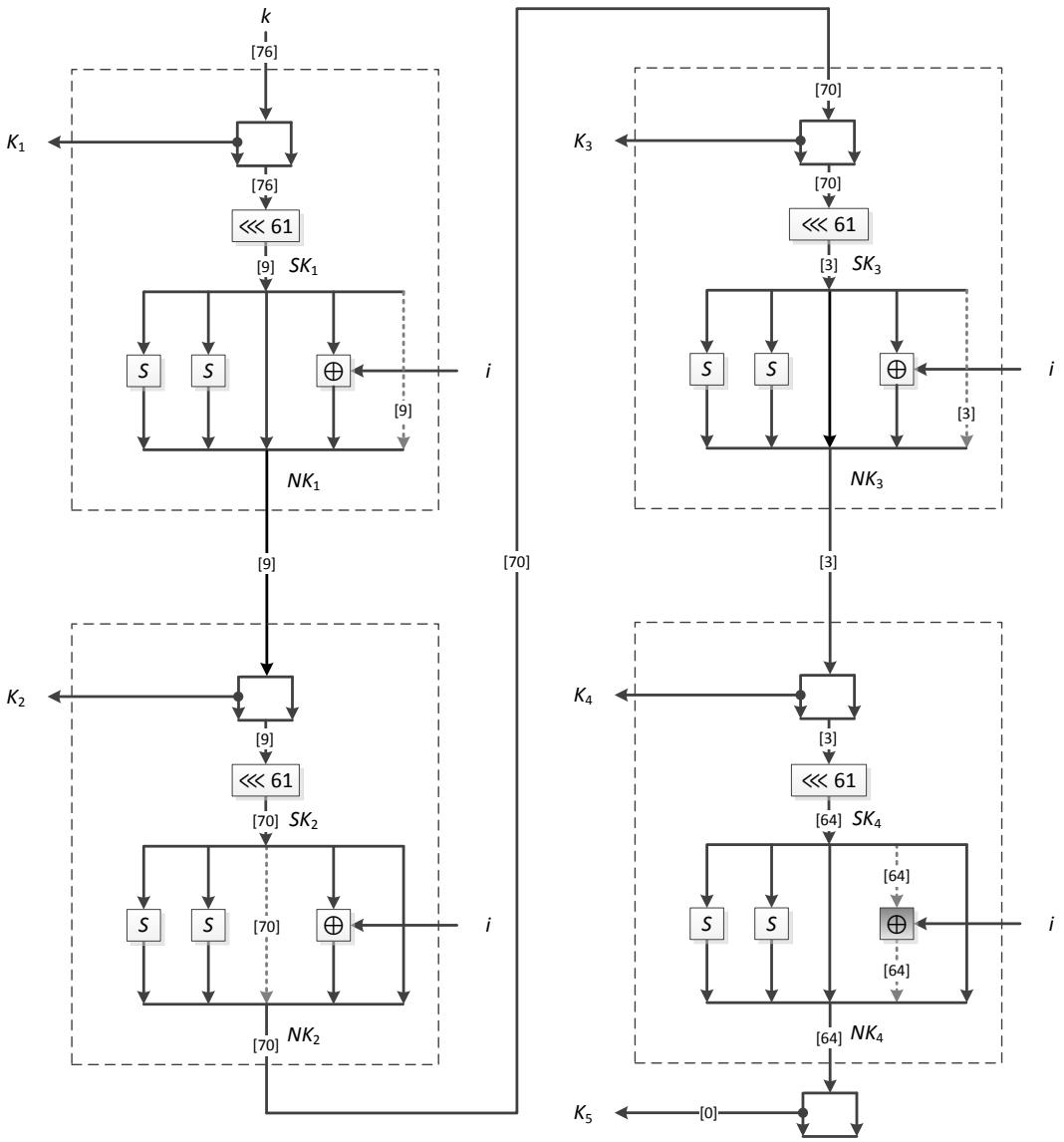
rodzaj	operacja	funkcja szyfrująca	algorytm generowania kluczy rundowych
XOR	$\oplus$	tak	tak
podstawienie	S	tak	tak
permutacja	P	tak	nie
rotacja	$\lll b$	nie	tak

W tabeli (tab. 3.32) przedstawiono listę operacji funkcji szyfrującej i algorytmu generowania kluczy rundowych.

Wniosek 3.21.

W algorytmie generowania kluczy rundowych szyfru PRESENT wykorzystywane są operacje nieliniowe – podstawienie S oraz operacje liniowe – XOR i rotacja. W algorytmie tym możliwe jest zwiększenia liczby iteracji (metoda B) i możliwe jest odpowiednie zwiększenie liczby rund szyfru, przy założeniu zwiększenia liczby bitów licznika i iteracji.

■



Rys. 3.35. PRESENT-128 – zależność bitu $K_5[0]$ klucza rundowego od bitów klucza głównego *k*

Na rysunku (rys. 3.35) przedstawiono zależność bitu $K_5[0]$ klucza rundowego od bitów klucza głównego *k*. Przy określeniu tej zależności uwzględniono następujące własności funkcji elementarnych:

- $\oplus$ – bit *i* wyjścia (wartości) funkcji XOR zależy od bitu *i* wszystkich wejść (argumentów),
- $\lll b$ – bit *i* wyjścia rotacji w lewo o *b* bitów słowa *l*-bitowego, w przypadku LSB = 0 po prawej, zależy od bitu $(i + (l - b))$ **mod** *l* wejścia,
- *S* – bit *i* wyjścia S-bloku *S* zależy od wszystkich bitów wejścia.

Ostatecznie bit $K_5[0]$ zależy tylko od bitu $k[76]$ klucza głównego, o długości 128 bitów, co stanowi 0.78%. Z rysunku (rys. 3.35) można też odczytać, że:

- bit $K_4[0]$ zależy od 1 bitu, $k[9]$,
- bit $K_3[0]$ zależy od 1 bitu, $k[70]$,
- bit $K_2[0]$ zależy od 1 bitu, $k[3]$,
- bit $K_1[0]$ zależy od 1 bitu, $k[64]$.

Jest to najgorszy przypadek, w którym podczas propagacji zależności ani razu nie natrafiono na S-blok *S*.

Wniosek 3.22.

W szyfrze PRESENT-128 klucz rundowy K_i (i = 1, 2, …, 32) zależy od 64 bitów 128-bitowego klucza głównego k, przy czym bit klucza K_i, w najgorszym przypadku, zależy od liczby bitów klucza k: 1 (i = 1,2, …, 5).

■

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru PRESENT, przedstawiono w postaci charakterystyki (charakterystyka 3.11).

Charakterystyka 3.11. Charakterystyka algorytmu generowania kluczy rundowych szyfru PRESENT

1. Długość klucza głównego: $|\boldsymbol{k}|$ **= 80b, 128b**.
2. Długość klucza rundowego: $|\boldsymbol{K_i}|$ **= 64b (*i* = 1, 2, …, 32)**.
3. Liczba iteracji: $\boldsymbol{R}$ **= 31**.
4. Zależność K_i od bitów k (PRESENT-128):
   - **64b / 128b (50.00%)**.
5. Zależność bitu K_i od bitów k (PRESENT-128):
   - $\boldsymbol{K_1[0], K_2[0], \ldots, K_5[0]}$**: 1b / 128b (0.78%)**.
6. Zależność grupy bitów K_i od grupy bitów k: **NIE**.
7. Operacja liniowe: $\oplus$, $\lll \boldsymbol{b}$.
8. Operacje nieliniowe: $\boldsymbol{S}$ **(4b × 4b)**.
9. Możliwości zwiększenia liczby iteracji: **warunek – zwiększenie liczby bitów licznika iteracji *i*.**
10. Indywidualizacja iteracji: **przez licznik *i*.**
11. Teoretyczna ocena jakości algorytmu: **0.78%.**

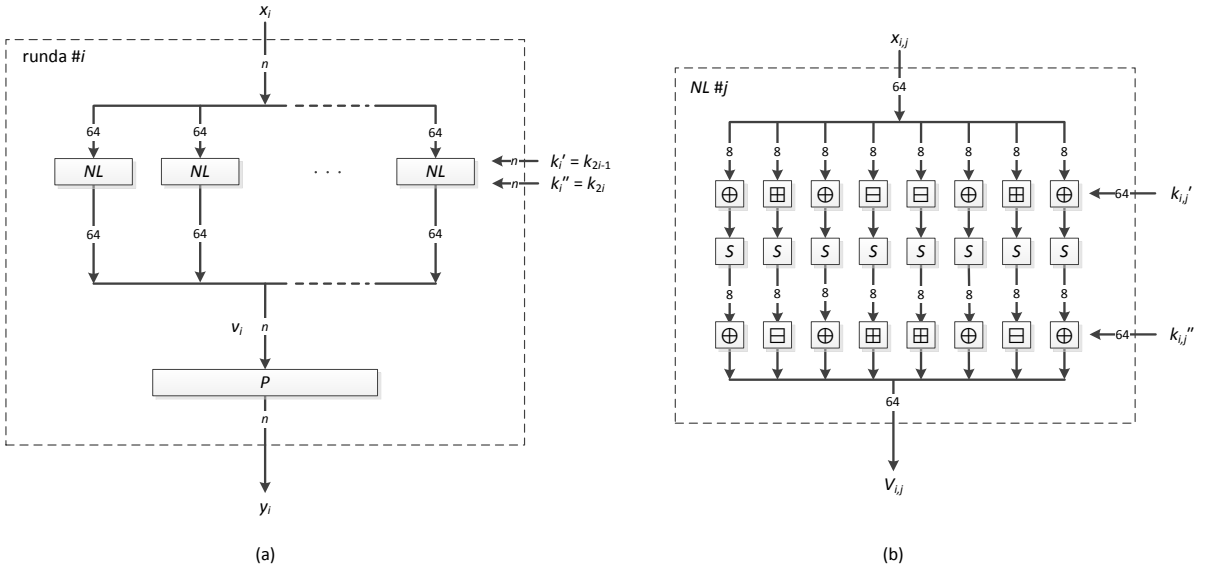
3.3.5. PP-1 (2010)

Szyfr blokowy PP-1 [38][42] jest szyfrem skalowalnym, o strukturze SPN, przetwarzającym bloki danych o rozmiarze $n = t \cdot 64$ bity ($t = 1, 2, 3, \ldots$), z wykorzystaniem klucza głównego k, o długości n bitów lub $2n$ bitów. Przekształcenie odbywa się w r rundach, gdzie r zależy od rozmiaru n przetwarzanego bloku (tab. 3.33). W rundzie i ($i = 1, 2, \ldots, r$) wykorzystywane są dwa n-bitowe klucze rundowe $k_i' = k_{2i-1}$ oraz $k_i'' = k_{2i}$.

Tab. 3.33. PP-1 – liczba rund *r* dla bloku o rozmiarze *n* bitów

rozmiar bloku *n*	64	128	192	256	…
liczba rund *r*	11	22	32	43	…

Szyfr PP-1 jest inwolucyjny, tj. ta sama struktura SPN szyfru jest wykorzystywana podczas szyfrowania i deszyfrowania. Przy deszyfrowaniu jednak, klucze rundowe podawane są w odwrotnej kolejności.



Rys. 3.36. PP-1 – struktura: (a) rundy #*i* (*i* = 1, 2, …, *r*) szyfru, (b) elementu nieliniowego *NL #j* (*j* = 1, 2, …, *t*) (szyfrowanie/deszyfrowanie)

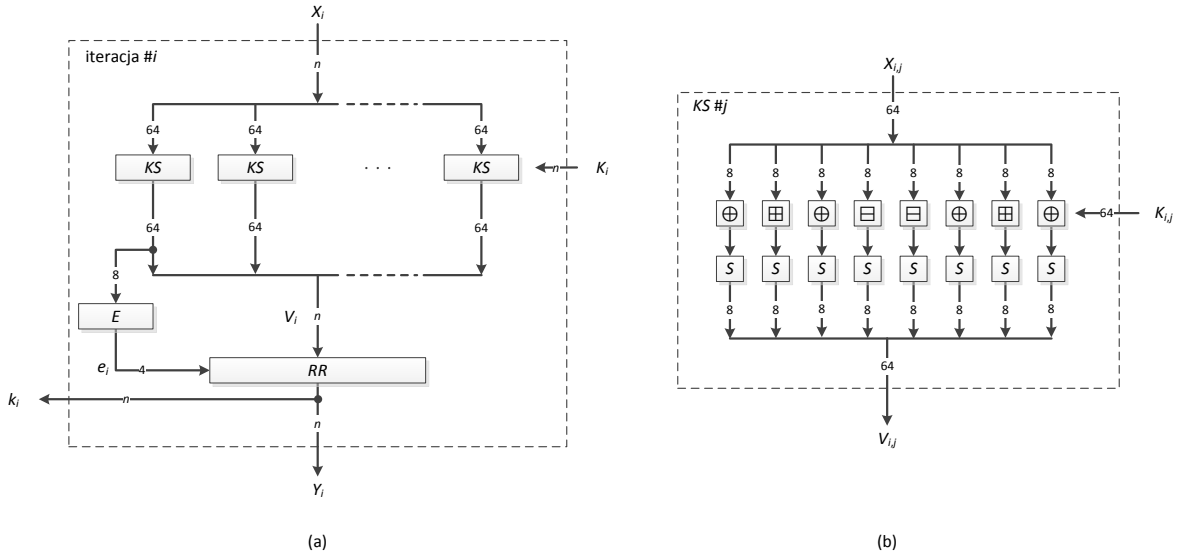
Rysunek (rys. 3.36) przedstawia pojedynczą rundę szyfru PP-1, zawierającą operacje:

- NL – inwolucyjna ($NL^{-1} = NL$) funkcja nieliniowa, składająca się z operacji wykonywanych na bajtach, 64-bitowego podbloku n-bitowego bloku x_i oraz bajtach, 64-bitowych podkluczy n-bitowych kluczy rundowych k_i' oraz k_i'', tj.:
  - $\oplus$ – suma wyłączająca (XOR) słów 8-bitowych (bajtów),
  - $\boxplus$ – dodawanie modulo 2^8,

- $\boxminus$ – odejmowanie modulo 2^8,
- S – inwolucyjna ($S^{-1} = S$) funkcja podstawień, S-blok o rozmiarze 8 × 8 bitów,

▪ P – inwolucyjna ($P^{-1} = P$), skalowalna permutacja bitowa, P-blok, definiowana z wykorzystaniem permutacji pomocniczej *Prm* (ang. *auxiliary permutation*).

Algorytm generowania kluczy rundowych szyfru PP-1 (algorytm 3.12) przetwarza n-bitową stałą X_0 w $R = 2r + 1$ iteracjach, z wykorzystaniem n-bitowych kluczy pomocniczych K_i ($i = 0, 1, \ldots, R-1$), zależnych od klucza głównego k szyfru. Wejściem tego algorytmu jest klucz k, a wyjściem są klucze rundowe $k_1, k_2, \ldots, k_{2r}$.



Rys. 3.37. PP-1 – struktura: (a) iteracji #*i* (*i* = 0, 1, …, 2*r*) funkcji generowania kluczy rundowych, (b) elementu *KS #j* (*j* = 1, 2, …, *t*)

W algorytmie generowania kluczy rundowych (algorytm 3.12) i funkcji iteracji, przedstawionej na rysunku (rys. 3.37), wykonywane są operacje:

▪ *KS* – funkcja składająca się z operacji wykonywanych na bajtach, 64-bitowego podsłowa n-bitowego słowa X_i oraz bajtach, 64-bitowego podklucza n-bitowego klucza K_i, tj.:

- $\oplus$ – suma wyłączająca (XOR) słów 8-bitowych (bajtów),
- $\boxplus$ – dodawanie modulo 2^8,
- $\boxminus$ – odejmowanie modulo 2^8,
- S – funkcja podstawień, S-blok o rozmiarze 8 × 8 bitów (identyczna z S w rundzie szyfru).

- E – funkcja obliczająca 4-bitową wartość $e_i = E(b_1b_2b_3b_4||b_9b_{10}b_{11}b_{12}) = (b_1 \oplus b_9)(b_2 \oplus b_{10})(b_3 \oplus b_{11})(b_4 \oplus b_{12})$, na podstawie konkatenacji czterech, najbardziej znaczących bitów wyjścia, dwóch skrajnych lewych S-bloków, ze skrajnego lewego elementu *KS* #1,
- RR – cykliczne przesunięcie (rotacja) w prawo n-bitowego słowa V_i o e_i bitów,
- Prm – permutacja pomocnicza wykorzystywana w obliczeniu stałej B, używanej m.in. jako wartość X_0,
- RL – cykliczne przesunięcie (rotacja) w lewo n-bitowego słowa o 1 bit,
- $\wedge$ – koniunkcja (AND) słów n-bitowych.

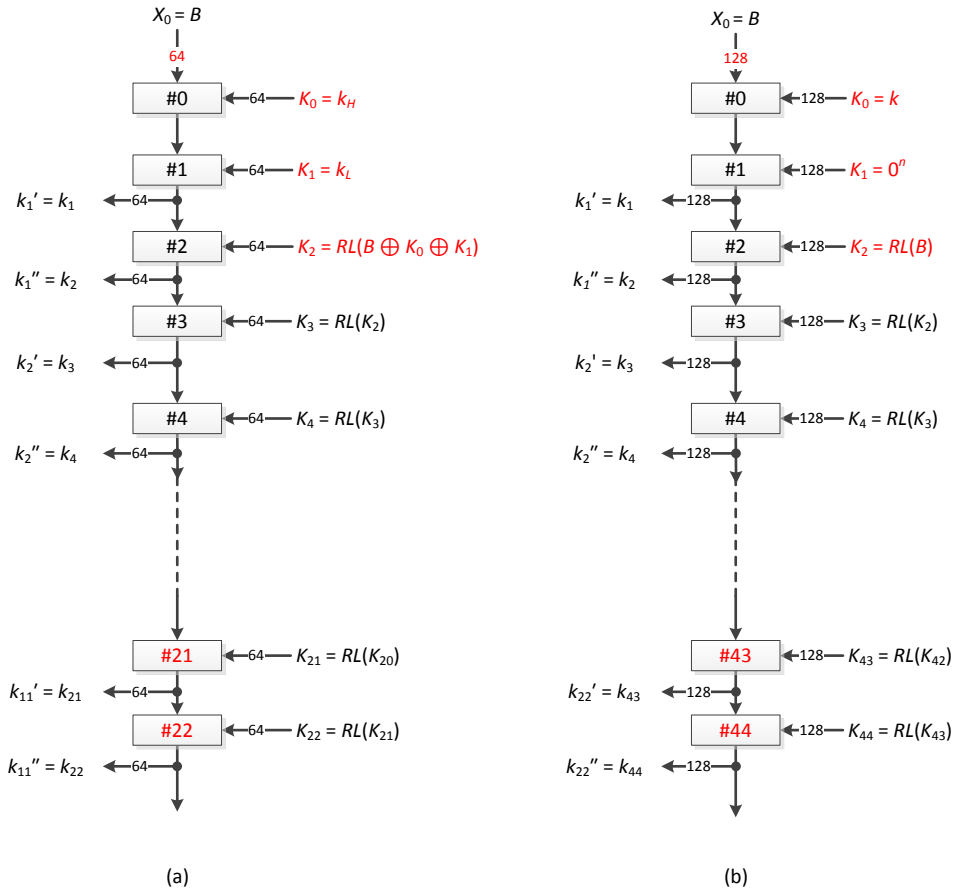
Algorytm 3.12. Algorytm generowania kluczy rundowych szyfru PP-1

WEJŚCIE: klucz główny k o długości n lub $2n$ bitów

WYJŚCIE: klucze rundowe $k_1, k_2, \ldots, k_{2r}$ o długości n bitów

PROCEDURA:

1. Na podstawie 64-bitowej stałej B_1 = 0x912B4769B2496E7C, oblicz n-bitową stałą $B = B_1||B_2||\ldots||B_t$ następująco: $B_j = Prm(B_{j-1})$ dla $j = 2, 3, \ldots, t$ i jako wejściowe słowo X_0 podstaw B, tj. $X_0 = B$, gdzie Prm to permutacja pomocnicza.
2. Na podstawie stałej B i klucza głównego k o długości n lub $2n$ bitów oblicz:
   - stałą A: $A = 0^n$ jeśli $|k| = n$ albo $A = 1^n$ jeśli $|k| = 2n$,
   - klucze pomocnicze K_i:
     - $K_0 = k$ i $K_1 = 0^n$ jeśli $|k| = n$ albo $K_0 = k_H$ i $K_1 = k_L$ jeśli $|k| = 2n$, gdzie $k = k_H||k_L$,
     - $K_2 = RL(B \oplus (A \wedge (K_0 \oplus K_1)))$,
     - $K_i = RL(K_{i-1})$ dla $i = 3, 4, \ldots, 2r$.
3. Dla $i = 0$ do $2r$ powtarzaj:
   - na podstawie $X_i = X_{i,1}||X_{i,2}||\ldots||X_{i,t}$ oraz $K_i = K_{i,1}||K_{i,2}||\ldots||K_{i,t}$ oblicz $V_i = V_{i,1}||V_{i,2}||\ldots||V_{i,t}$ w następujący sposób: $V_{i,j} = KS(X_{i,j}, K_{i,j})$ dla $j = 1, 2, \ldots, t$,
   - dla $V_{i,1} = b_1b_2\ldots b_{64}$, oblicz $e_i = E'(V_{i,1}) = (b_1 \oplus b_9)(b_2 \oplus b_{10})(b_3 \oplus b_{11})(b_4 \oplus b_{12})$, gdzie E' jest rozszerzeniem funkcji E na słowa 64-bitowe. Oblicz $Y_i = RR(V_i, e_i)$ i jeśli $i > 0$ jako k_i podstaw Y_i, tj. $k_i = Y_i$,
   - jako X_{i+1} podstaw Y_i, tj. $X_{i+1} = Y_i$.



Rys. 3.38. PP-1 – struktura funkcji generowania kluczy rundowych dla szyfru: (a) 64/128, (b) 128/128

Na rysunku (rys. 3.38) przedstawiono strukturę algorytmu generowania kluczy rundowych szyfru PP-1 dla dwóch przypadków:

- w przypadku (a), 64/128, rozmiar bloku n = 64 bity i długość klucza głównego $|k|$ = 128 bitów – dla takiego szyfru mamy t = 1 i r = 11,
- w przypadku (b), 128/128, mamy n = 128 bitów, $|k|$ = 128 bitów, a zatem t = 2 i r = 22.

Tab. 3.34. PP-1 – zestawienie cech szyfru w postaci liczbowej

rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego *|k|*	liczba kluczy rundowych *nrk*	długość klucza rundowego *|rk|*	długość próbki kluczy rundowych *|rkp|*
64b	11	64b lub 128b	22	64b	1408b
128b	22	128b lub 256b	44	128b	5632b
192b	32	192b lub 384b	64	192b	12288b
256b	43	256b lub 512b	86	256b	22016b
…	…	…	…	…	…

W tabeli (tab. 3.34) przedstawiono zestawienie podstawowych cech skalowalnego szyfru PP-1 w postaci liczbowej, natomiast w tabeli (tab. 3.35) przedstawiono te cechy w postaci ogólnej.

Tab. 3.35. PP-1 – zestawienie cech szyfru w postaci ogólnej

rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego |*k*|	liczba kluczy rundowych *nrk*	długość klucza rundowego |*rk*|	długość próbki kluczy rundowych |*rkp*|
$t \cdot 64$	$\lceil n / 6 \rceil$	n $2n$	$2r$	n	$2r \cdot n$

W tabeli (tab. 3.36) przedstawiono listę operacji funkcji szyfrującej/deszyfrującej i algorytmu generowania kluczy rundowych. Do wspólnych operacji funkcji i algorytmu należą: $\oplus$, $\boxplus$, $\boxminus$, *S*, *Prm*. Operacje $\boxplus$, $\boxminus$, *S*, *RR* i $\wedge$ są nieliniowe, a operacje $\oplus$, *P*, *Prm*, *E* i *RL* są liniowe (formuła (3.17)).

Tab. 3.36. PP-1 – lista operacji

rodzaj	operacja	funkcja szyfrująca/deszyfrująca	algorytm generowania kluczy rundowych
XOR	$\oplus$	tak	tak
dodawanie modulo 256	$\boxplus$	tak	tak
odejmowanie modulo 256	$\boxminus$	tak	tak
podstawienie	*S*	tak	tak
permutacja	*P*	tak	nie
permutacja pomocnicza	*Prm*	tak	tak
funkcja	*E*	nie	tak
rotacja sterowana danymi	*RR*	nie	tak
rotacja w lewo	*RL*	nie	tak
koniunkcja (AND)	$\wedge$	nie	tak

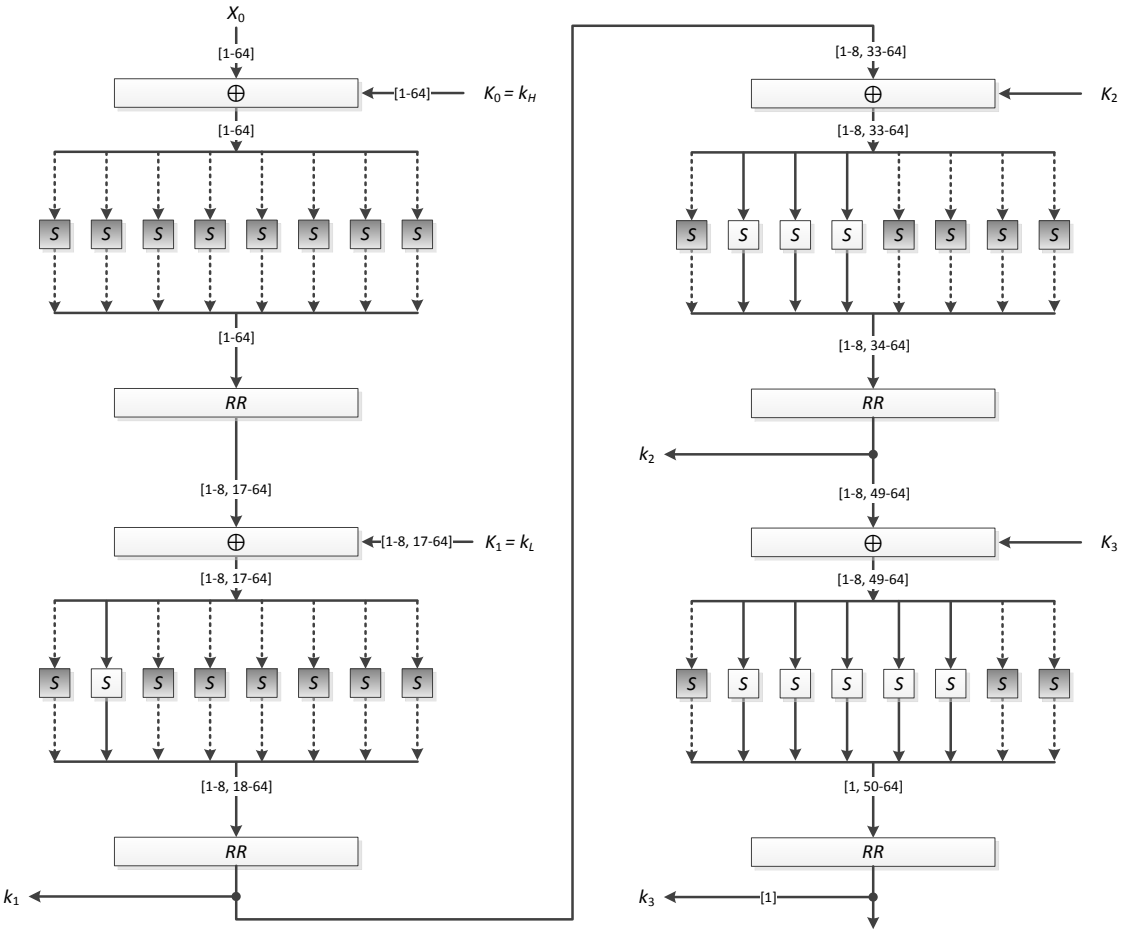
Wniosek 3.23.

W algorytmie generowania kluczy rundowych szyfru PP-1 wykorzystywane są operacje liniowe i nieliniowe (w szczególności S-blok *S* o rozmiarze 8 × 8 bitów). Przez zwiększenie liczby iteracji tego algorytmu możliwe jest zwiększenie liczby rund szyfru PP-1 (metoda B).

■

W pracach [76][86][87] wykazano, że z uwagi na inwolucyjny charakter szyfru, możliwe jest przeprowadzenia kryptoanalizy różnicowej, dla pełnego szyfru PP-1, przy rozmiarze bloku n = 64, 128, 192, 256 bitów, o złożoności odpowiednio: 2^{50}, 2^{110}, 2^{164}, 2^{224}, sugerując równocześnie, że każdy z wariantów szyfru PP-1 powinien posiadać więcej rund,

aniżeli przewidzieli to Autorzy szyfru. Natomiast w pracy [46] przedstawiono wyniki kryptoanalizy liniowej szyfru PP-1 oraz szyfru PP-2, omówionego w kolejnym punkcie.



Rys. 3.39. PP-1 64/128 – zależność bitu k_3[1] klucza rundowego od bitów klucza głównego $k = k_H||k_L$

Na rysunku (rys. 3.39) przedstawiono zależność bitu $k_3[1]$ klucza rundowego od bitów klucza głównego $k = k_H||k_L$. Funkcję E pominięto pamiętając, że rotacja RR sterowana jest danymi poprzez 4-bitową wartość e_i. Operacje $\boxplus$ i $\boxminus$ zastąpiono operacją $\oplus$, bez wpływu na wynik analizy, w celu uproszczenia rysunku. Przy określeniu tej zależności uwzględniono następujące własności funkcji elementarnych:

- $\oplus$ – bit i wyjścia (wartości) funkcji XOR zależy od bitu i wszystkich wejść (argumentów),
- S – bit i wyjścia S-bloku S zależy od wszystkich bitów wejścia,
- $\ggg_d\ b$ – bit i wyjścia rotacji w prawo, sterowanej d-bitowymi danymi, słowa l-bitowego, w przypadku MSB = 1 po lewej, zależy od bitów: i, $((i-1)+1)$ **mod** $l+1$, $((i-1)+2)$ **mod** $l+1$, …, $((i-1)+2^d-1)$ **mod** $l+1$ wejścia.

Ostatecznie bit $k_3[1]$ zależy od bitów $k_H[1\text{-}64]$, $k_L[1\text{-}8, 17\text{-}64]$, a więc zależy od 120 bitów 128-bitowego klucza głównego k, co stanowi 93.75%. Z rysunku (rys. 3.39) można też odczytać, że:

- bit $k_4[1]$ zależy od 128 bitów,
- bit $k_2[1]$ zależy od 96 bitów,
- bit $k_1[1]$ zależy od 64 bitów.

Podobne wyniki uzyskiwane są dla dowolnego bitu kluczy rundowych k_1, k_2, k_3, k_4. Należy podkreślić, że w każdej rundzie szyfru wykorzystywane są dwa klucze rundowy.

Wniosek 3.24.

W szyfrze PP-1 64/128 klucz rundowy k_i (i = 1, 2, …, 2r) zależy od wszystkich bitów 128-bitowego klucza głównego k, przy czym każdy bit klucza k_i zależy od liczby bitów klucza k: 64 (i = 1), 96 (i = 2), 120 (i = 3) lub 128 (i = 4,5, …, 2r).

■

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru PP-1, przedstawiono w postaci charakterystyki (charakterystyka 3.12).

Charakterystyka 3.12. Charakterystyka algorytmu generowania kluczy rundowych szyfru PP-1

1. Długość klucza głównego: $|\boldsymbol{k}| = \boldsymbol{n}, \boldsymbol{2n}$.
2. Długość klucza rundowego: $|\boldsymbol{k_i}| = \boldsymbol{n}$ $\boldsymbol{(i = 1, 2, \ldots, 2r)}$.
3. Liczba iteracji: $\boldsymbol{R = 2r + 1}$.
4. Zależność k_i od bitów k (PP-1 64/128):
   - **128b / 128b (100.00%)**.
5. Zależność bitu k_i od bitów k (PP-1 64/128):
   - $\boldsymbol{k_1[j]}$**: 64b / 128b (50.00%)**, $\boldsymbol{j = 1, 2, \ldots, n}$,
   - $\boldsymbol{k_2[j]}$**: 96b / 128b (75.00%)**,
   - $\boldsymbol{k_3[j]}$**: 120b / 128b (93.75%)**,
   - $\boldsymbol{k_4[j], k_5[j], \ldots, k_{2r}[j]}$**: 128b / 128b (100.00%)**.
6. Zależność grupy bitów k_i od grupy bitów k: **NIE**.
7. Operacja liniowe: $\oplus$**,** ***Prm, E, RL***.
8. Operacje nieliniowe: $\boxplus$**,** $\boxminus$**,** ***S*** **(8b × 8b),** ***RR*****,** $\wedge$.
9. Możliwości zwiększenia liczby iteracji: **TAK**.
10. Indywidualizacja iteracji: **przez klucze pomocnicze** $\boldsymbol{K_i}$**.**
11. Teoretyczna ocena jakości algorytmu: **83.75%**.

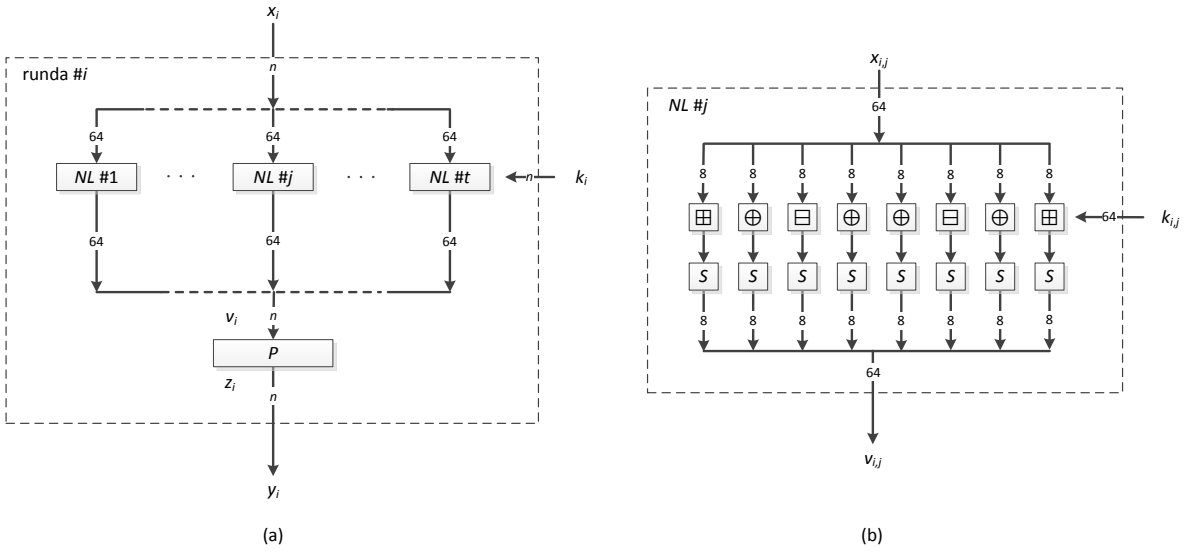
3.3.6. PP-2 (2013)

Szyfr blokowy PP-2 [39] jest szyfrem skalowalnym, o strukturze SPN, przetwarzającym bloki danych o rozmiarze $n = t \cdot 64$ bity ($t = 1, 2, 3, \ldots$), z wykorzystaniem klucza głównego k, o długości $|k| = d \cdot 64$ bity ($d = t, t + 1, t + 2, \ldots$).

Tab. 3.37. PP-2 – liczba rund *r* dla bloku o rozmiarze *n* bitów i klucza o długości |*k*| bitów

długość |*k*|	rozmiar bloku *n*				
	64b	128b	192b	256b	…
64b	11	-	-	-	-
128b	13	22	-	-	-
192b	15	24	32	-	-
256b	17	26	34	43	-
320b	19	28	36	45	…
384b	21	30	38	47	…
…	…	…	…	…	

Przekształcenie odbywa się w r rundach, gdzie r zależne jest od rozmiaru n przetwarzanego bloku oraz długości $|k|$ klucza głównego (tab. 3.37). W rundzie i ($i = 1, 2, \ldots, r$) wykorzystywany jest jeden n-bitowy klucz rundowy k_i. Szyfr PP-2, w odróżnieniu od szyfru PP-1, nie jest inwolucyjny. Przy deszyfrowaniu stosowane są funkcje odwrotne i wykonywane są one w odwrotnej kolejności. W szczególności, klucze rundowe podawane są w odwrotnej kolejności.



Rys. 3.40. PP-2 – struktura: (a) rundy #*i* (*i* = 1, 2, …, *r*) szyfru, (b) elementu nieliniowego NL #*j* (*j* = 1, 2, …, *t*) (szyfrowanie)

Strukturę pojedynczej rundy, złożonej z $t = n / 64$ równoległych ścieżek przetwarzania, przedstawiono na rysunku (rys. 3.40). Operacje składowe rundy szyfrowania to:

- *NL* – funkcja nieliniowa zawierająca operacje:
  - $\oplus$ – suma wyłączają (XOR) słów 8-bitowych,
  - $\boxplus$ – dodawanie modulo 2^8,
  - $\boxminus$ – odejmowanie modulo 2^8,
  - *S* – funkcja podstawień, S-blok o rozmiarze 8 × 8 bitów,
- *P* – skalowalna permutacja bitowa, P-blok, definiowana z wykorzystaniem wielokrotnych rotacji: $P(x) = RR\left(x \wedge 0\text{x}8^{n/4}, 12\right) \vee RR\left(x \wedge 0\text{x}4^{n/4}, 28\right) \vee RR\left(x \wedge 0\text{x}2^{n/4}, 44\right) \vee RR\left(x \wedge 0\text{x}1^{n/4}, 60\right)$.

Tab. 3.38. PP-2 – permutacja bitowa *P* oraz permutacja do niej odwrotna P^{-1} (*n* = 64)

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

W tabeli (tab. 3.38) przedstawiono permutację *P*, dla *n* = 64, w postaci tablicy. Najbardziej znaczący bit 1 wejścia, jest bitem 13 wyjścia permutacji *P*. W permutacji odwrotnej P^{-1}, wykorzystywanej w dalszej części niniejszego punktu, bit 13 wejścia jest bitem 1 wyjścia.

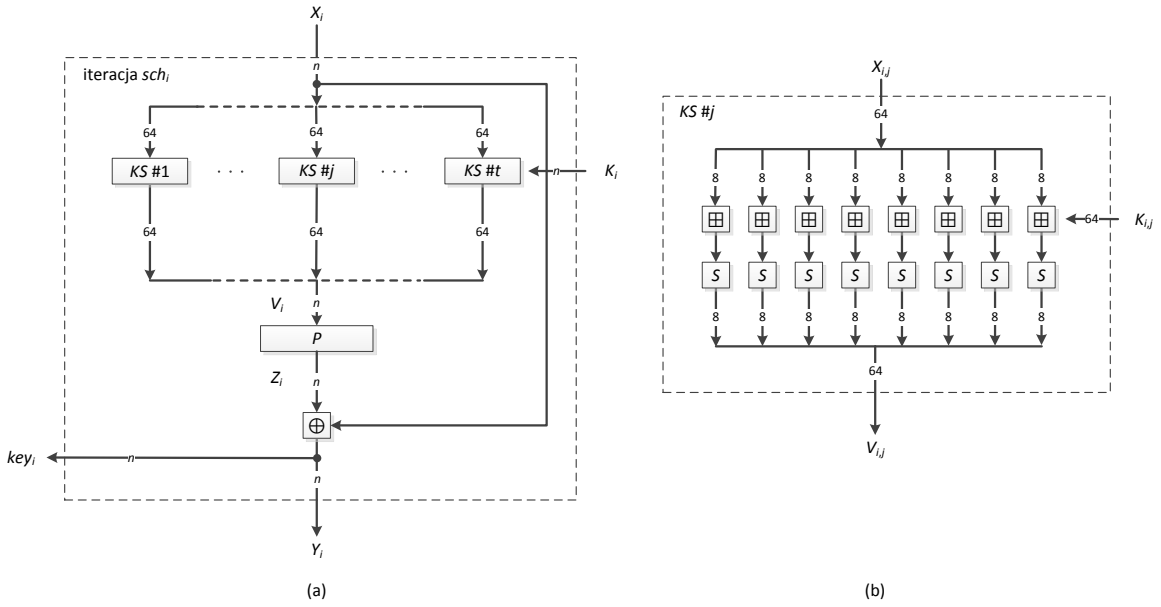
Algorytm generowania kluczy rundowych szyfru PP-2 (algorytm 3.13) przetwarza *n*-bitową stałą X_1, w *R* iteracjach, z wykorzystaniem kluczy pomocniczych K_i (i = 1, 2, …, *R*), zależnych od klucza głównego *k* szyfru. Wejściem tego algorytmu jest klucz *k*, a wyjściem są klucze rundowe $k_1, k_2, \ldots, k_r$.

W algorytmie (algorytm 3.13) i w funkcji iteracji, przedstawionej na rysunku (rys. 3.41), wykonywane są następujące operacje:

- *KS* – funkcja zawierająca operacje:
  - $\boxplus$ – dodawanie modulo 2^8,

- S – funkcja podstawień, S-blok o rozmiarze 8 × 8 bitów (identyczna z S w rundzie szyfru),

- P – skalowalna permutacja bitowa, P-blok, definiowana z wykorzystaniem wielokrotnych rotacji (identyczna z P w rundzie szyfru),
- $\oplus$ – suma wyłączająca (XOR) słów n-bitowych,
- $RR(x, b)$ – cykliczne przesunięcie (rotacja) w prawo słowa x o b bitów.



Rys. 3.41. PP-2 – struktura: (a) iteracji sch_i ($i = 1, 2, \ldots, R$) funkcji generowania kluczy rundowych, (b) elementu *KS #j* ($j = 1, 2, \ldots, t$)

Algorytm 3.13. Algorytm generowania kluczy rundowych szyfru PP-2

WEJŚCIE: klucz główny k o długości $|k| = d \cdot 64$ bity

WYJŚCIE: klucze rundowe $k_1, k_2, \ldots, k_r$ o długości n bitów

PROCEDURA:

1. Na podstawie 64-bitowych stałych $c_{0,1}$ = 0xE3729424EDBC5389 i $c_{1,1}$ = 0x59F0E217D8AC6B43 oblicz n-bitowe stałe:
   - $c_0 = RR(c_{0,1}, 0)||RR(c_{0,1}, 1)||\ldots||RR(c_{0,1}, t–1)$,
   - $c_1 = RR(c_{1,1}, 0)||RR(c_{1,1}, 1)||\ldots||RR(c_{1,1}, t–1)$

   oraz jako X_1 podstaw c_1, tj. $X_1 = c_1$.
2. Na podstawie stałej c_0 i klucza głównego k o długości $d \cdot 64$ bity, uzupełnionego zerami do długości $\lceil d / t \rceil \cdot n$ bitów i po uzupełniniu traktowanego jako konkatenacja n-bitowych podkluczy, tj. $k = \kappa_1||\kappa_2||\ldots||\kappa_{\lceil d/t \rceil}$, wykonaj:
   - oblicz liczbę iteracji: $R = r + \lceil d / t \rceil \cdot t$,

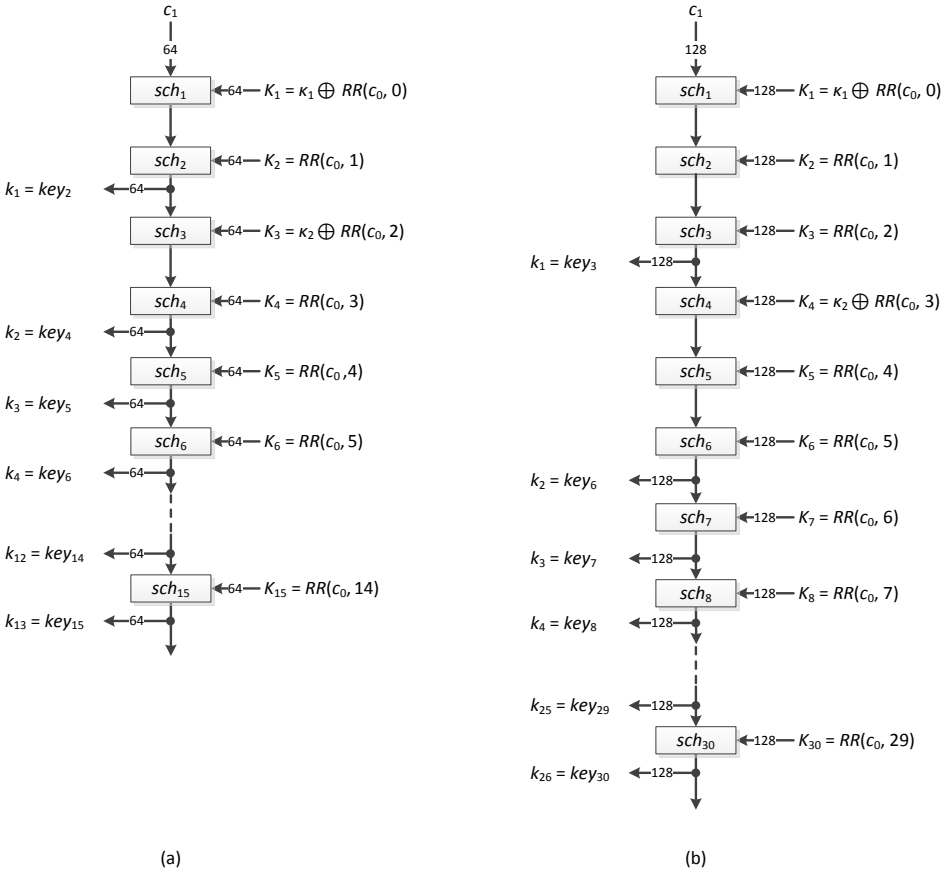
- utwórz ciąg:

$$(K_i^*)_{i=1}^{R} = \left(\kappa_1, \underbrace{0^n, 0^n, \dots, 0^n}_{t}, \kappa_2, \underbrace{0^n, 0^n, \dots, 0^n}_{t}, \dots, \kappa_{\lceil d/t \rceil}, \underbrace{0^n, 0^n, \dots, 0^n}_{t}, \underbrace{0^n, 0^n, \dots, 0^n}_{r - \lceil d/t \rceil}\right),$$

- oblicz klucze pomocnicze: $K_i = K_i^* \oplus RR(c_0, i-1)$, dla $i = 1, 2, \dots, R$.

3. Dla $i = 1$ do R powtarzaj:
   - na podstawie $X_i = X_{i,1}||X_{i,2}||\dots||X_{i,j}||\dots||X_{i,t}$ oraz $K_i = K_{i,1}||K_{i,2}||\dots||K_{i,j}||\dots||K_{i,t}$ oblicz $V_i = V_{i,1}||V_{i,2}||\dots||V_{i,j}||\dots||V_{i,t}$, w następujący sposób: $V_{i,j} = KS(X_{i,j}, K_{i,j})$, dla $j = 1, 2, \dots, t$,
   - na podstawie V_i oraz X_i oblicz $Z_i = P(V_i)$ oraz $Y_i = Z_i \oplus X_i$, a także jako key_i podstaw Y_i, tj. $key_i = Y_i$,
   - jako X_{i+1} podstaw Y_i, tj. $X_{i+1} = Y_i$.
4. Na podstawie kluczy key_i ($i = 1, 2, \dots, R$), wyznacz klucze rundowe k_l ($l = 1, 2, \dots, r$) następująco:

$$k_l = \begin{cases} key_{l(t+1)}, & \text{dla } l \le \lceil d/t \rceil \\ key_{\lceil d/t \rceil \cdot t + l}, & \text{dla } l > \lceil d/t \rceil \end{cases}.$$



Rys. 3.42. PP-2 – struktura funkcji generowania kluczy rundowych dla szyfru: (a) 64/128, (b) 128/256

Na rysunku (rys. 3.42) przedstawiono strukturę funkcji generowania kluczy rundowych szyfru PP-2, dla dwóch przypadków:

- w przypadku (a), 64/128, rozmiar bloku n = 64 bity i długość klucza głównego $|k|$ = 128 bitów – dla takiego szyfru zachodzi: t = 1, d = 2, r = 13, R = 15,
- w przypadku (b), 128/256, mamy n = 128 bitów i $|k|$ = 256 bitów, a zatem: t = 2, d = 4, r = 26, R = 30.

Tab. 3.39. PP-2 – zestawienie cech szyfru

rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego *|k|*	liczba kluczy rundowych *nrk*	długość klucza rundowego *|rk|*	długość próbki kluczy rundowych *|rkp|*
64b	11	64b	11	64b	704b
64b	13	128b	13	64b	832b
…	…	…	…	…	…
128b	22	128b	22	128b	2816b
128b	24	192b	24	128b	3072b
128b	26	256b	26	128b	3328b
…	…	…	…	…	…
256b	43	256b	43	256b	11008b
…	…	…	…	…	…

W tabeli (tab. 3.39) przedstawiono zestawienie podstawowych cech skalowalnego szyfru PP-2 w postaci liczbowej, natomiast w tabeli (tab. 3.40) przedstawiono te cechy w postaci ogólnej.

Tab. 3.40. PP-2 – zestawienie cech szyfru

rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego *|k|*	liczba kluczy rundowych *nrk*	długość klucza rundowego *|rk|*	długość próbki kluczy rundowych *|rkp|*
$t \cdot 64$ $t = 1, 2, 3, \ldots$	$\lfloor n/6 \rfloor + 2(d - t)$	$d \cdot 64$ $d = t, t + 1, t + 2, \ldots$	r	n	$r \cdot n$

W tabeli (tab. 3.41) przedstawiono listę operacji funkcji szyfrującej i algorytmu generowania kluczy rundowych. Do wspólnych operacji funkcji i algorytmu należą: $\oplus$, $\boxplus$, S, P, RR. Operacje $\boxplus$, $\boxminus$, S są nieliniowe, a operacje $\oplus$, P, RR są liniowe (formuła (3.17)).

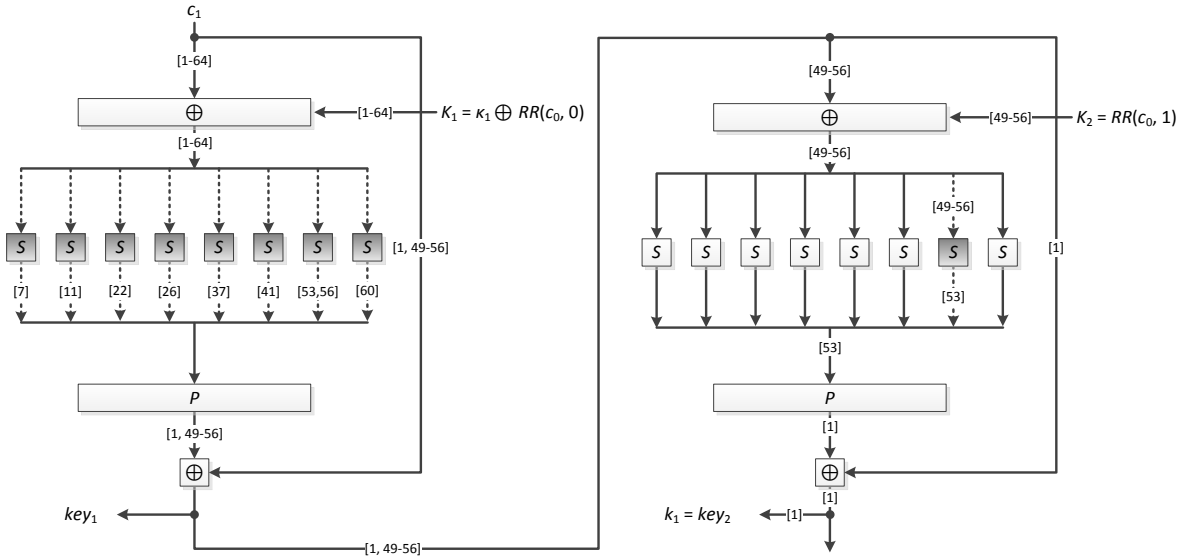
Tab. 3.41. PP-2 – lista operacji

rodzaj	operacja	funkcja szyfrująca/deszyfrująca	algorytm generowania kluczy rundowych
XOR	$\oplus$	tak	tak
dodawanie modulo 256	$\boxplus$	tak	tak
odejmowanie modulo 256	$\boxminus$	tak	nie
podstawienie	S	tak	tak
permutacja z rotacjami	P	tak	tak
rotacja	$RR(x, b)$	tak	tak

Wniosek 3.25.

W algorytmie generowania kluczy rundowych szyfru PP-2 wykorzystywane są operacje liniowe i nieliniowe (w szczególności S-blok S o rozmiarze 8 × 8 bitów). Przez zwiększenie liczby iteracji tego algorytmu możliwe jest zwiększenie liczby rund szyfru PP-2 (metoda B).

■

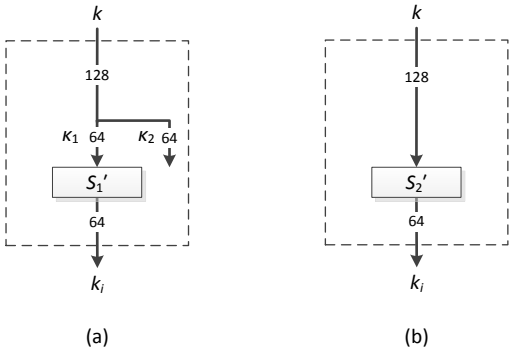


Rys. 3.43. PP-2 64/128 – zależność bitu $k_1[1]$ klucza rundowego od bitów klucza głównego $k = \kappa_1 \| \kappa_2$

Na rysunku (rys. 3.43) przedstawiono zależność bitu $k_1[1]$ klucza rundowego od bitów klucza głównego $k = \kappa_1 \| \kappa_2$. Operację $\boxplus$ zastąpiono operacją $\oplus$, bez wpływu na wynik analizy, w celu uproszczenia rysunku. Przy określeniu tej zależności uwzględniono następujące własności funkcji elementarnych:

- $\oplus$ – bit i wyjścia (wartości) funkcji XOR zależy od bitu i wszystkich wejść (argumentów),
- S – bit i wyjścia S-bloku S zależy od wszystkich bitów wejścia,
- P – bit i wyjścia permutacji bitowej P, jako jedyny, zależy od pewnego bitu j wejścia,

Ostatecznie bit $k_1[1]$ zależy od bitów $\kappa_1[1\text{-}64]$, a więc zależy od 64 bitów 128-bitowego klucza głównego k, co stanowi 50.00%. Dzięki właściwej konstrukcji permutacji P, po dwóch iteracjach (każdy) bit wyjścia zależy od wszystkich bitów wejścia.



Rys. 3.44. PP-2 64/128 – schemat zastępczy funkcji generowania klucza rundowego k_i: (a) $i = 1$, (b) $i = 2, 3, \ldots, r = 13$

Na rysunku (rys. 3.44) przedstawiono schemat zastępczy funkcji generowania klucza rundowego k_i. Każdy bit klucza k_2 i następnych zależy od wszystkich bitów klucza głównego k.

Wniosek 3.26.

W szyfrze PP-2 64/128, klucz rundowy k_i (i = 1, 2, …, r) zależy od liczby bitów 128-bitowego klucza głównego k: 64 (i = 1) lub 128 (i = 2, 3, …, r), przy czym każdy bit klucza k_i zależy od liczby bitów klucza k: 64 (i = 1) lub 128 (i = 2, 3, …, r).

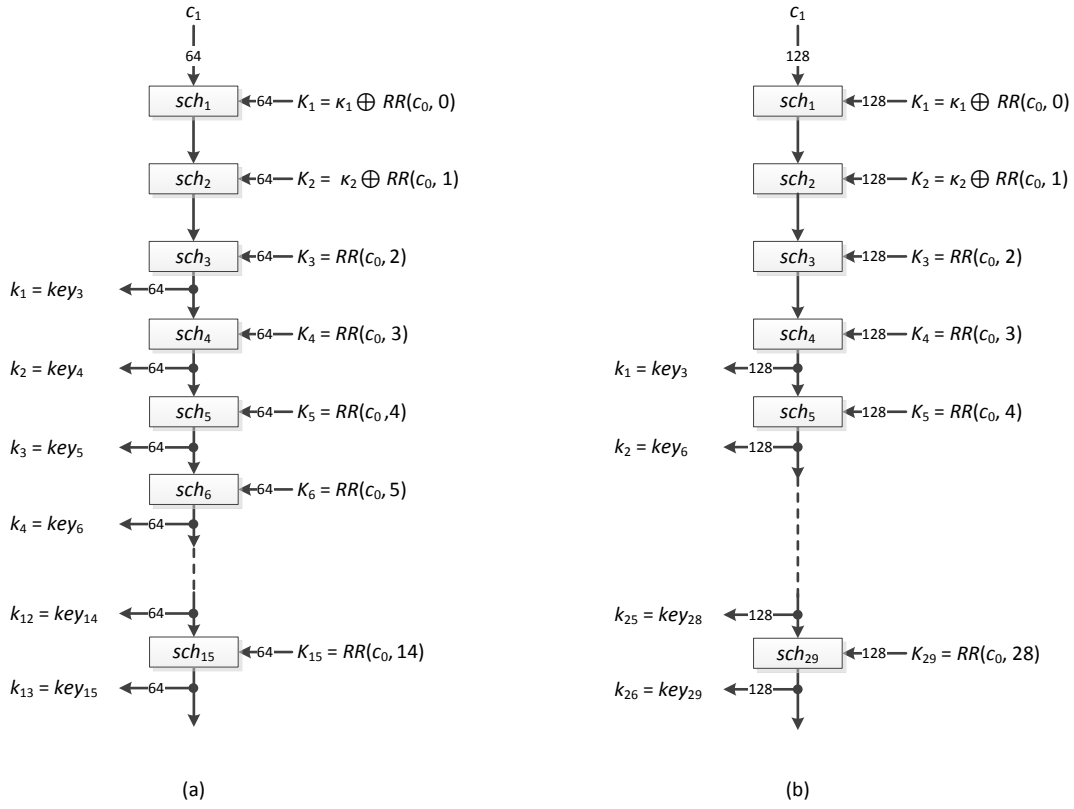
■

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru PP-2, przedstawiono w postaci charakterystyki (charakterystyka 3.13).

Charakterystyka 3.13. Charakterystyka algorytmu generowania kluczy rundowych szyfru PP-2

1. Długość klucza głównego ($d = t, t+1, t+2, \ldots; t = 1, 2, 3, \ldots$): $\boldsymbol{|k| = d \cdot 64b}$.
2. Długość klucza rundowego ($n = t \cdot 64$): $\boldsymbol{|k_i| = n\ (i = 1, 2, \ldots, r)}$.
3. Liczba iteracji: $\boldsymbol{R = r + \lceil d / t \rceil \cdot t}$.
4. Zależność k_i od bitów k (PP-2 64/128):
   - $\boldsymbol{k_1}$**: 64b / 128b (50.00%)**,
   - $\boldsymbol{k_2, k_3, \ldots, k_r}$**: 128b / 128b (100.00%)**.
5. Zależność bitu k_i od bitów k (PP-2 64/128):
   - $\boldsymbol{k_1[j]}$**: 64b / 128b (50.00%)**, $\boldsymbol{j = 1, 2, \ldots, n}$,
   - $\boldsymbol{k_2[j], k_3[j], \ldots, k_r[j]}$**: 128b / 128b (100.00%)**.
6. Zależność grupy bitów k_i od grupy bitów k: **NIE**.
7. Operacja liniowe: $\boldsymbol{\oplus, P, RR}$.

8. Operacje nieliniowe: $\boxplus$, ***S*** **(8b × 8b)**.
9. Możliwości zwiększenia liczby iteracji: **TAK**.
10. Indywidualizacja iteracji: **przez klucze pomocnicze K_i**.
11. Teoretyczna ocena jakości algorytmu: **90.00%**.



Rys. 3.45. PP-2′ – struktura funkcji generowania kluczy rundowych dla szyfru: (a) 64/128, (b) 128/256

Na rysunku (rys. 3.45) przedstawiono strukturę funkcji generowania kluczy rundowych dla szyfru PP-2′. Klucz główny $k = \kappa_1 || \kappa_2$ podawany jest w dwóch pierwszych iteracjach, po czym następuje odpowiednia liczba iteracji rozbiegowych. Liczba ta, dla rozmiaru bloku n = 64 (t = 1) jest równa 1, dla n = 128 (t =2) jest równa 2, ale dla większych wartości $n = t \cdot 64$ jest różna od t.

W szyfrze PP-2′ każdy bit klucza rundowego k_1 i następnych, zależy od wszystkich bitów klucza głównego k. Szyfr PP-2′ 128/256 ma o 1 iterację mniej od szyfru PP-2 128/256.

Wniosek 3.27.

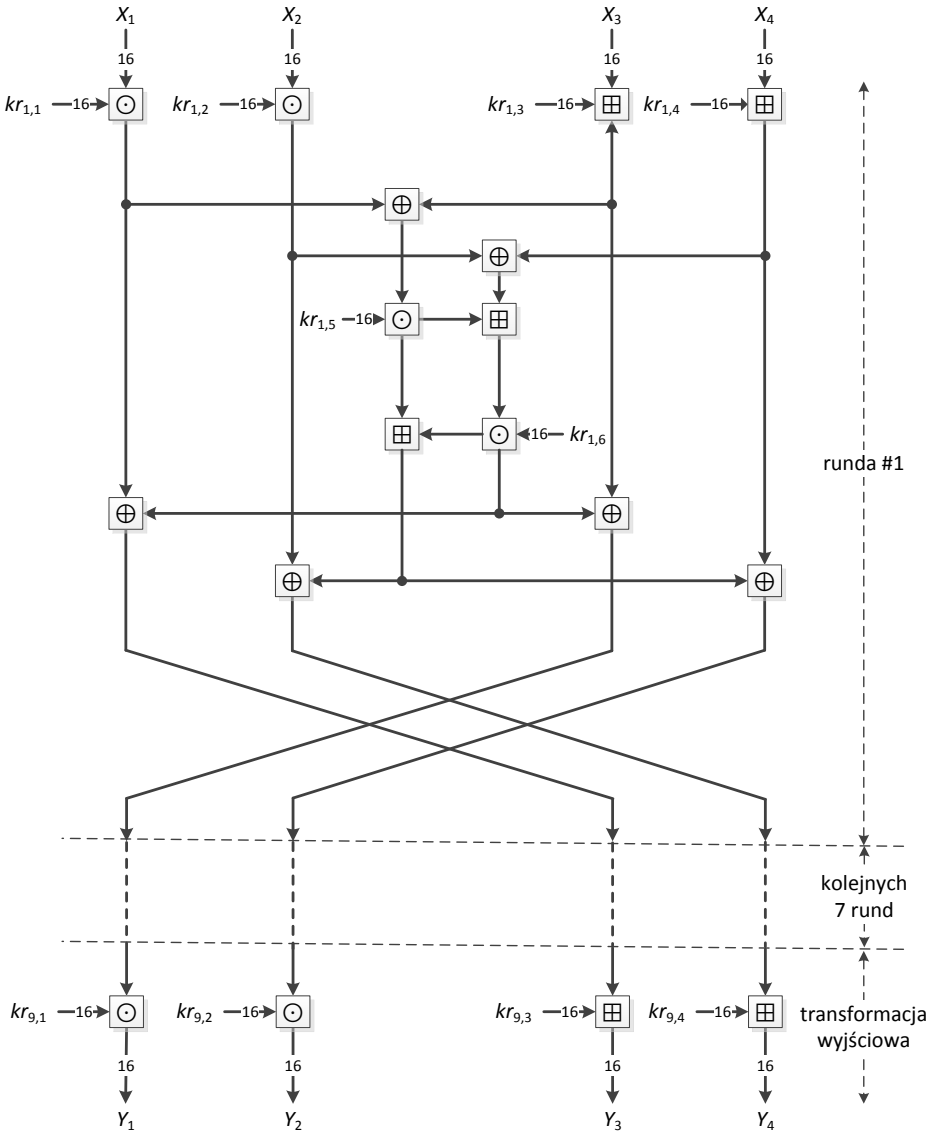
W szyfrze PP-2′, n-bitowy klucz rundowy k_i (i = 1, 2, …, r) zależy od wszystkich bitów ($d \cdot 64$)-bitowego klucza głównego k, przy czym każdy bit klucza k_i zależy od wszystkich bitów klucza k, gdzie $n = t \cdot 64$ oraz $d \geq t$.

■

3.4. Szyfry blokowe o konstrukcji innej

3.4.1. IDEA (1991)

Szyfr blokowy IDEA (ang. *International Data Encryption Algorithm*) [74] operuje na 64-bitowych blokach danych, a klucz główny, wykorzystywany w szyfrze IDEA, ma długość 128 bitów. Szyfr IDEA przetwarza bloki danych, podzielone na 16 bitowe słowa, w 9 rundach. Niekiedy mowa jest o 8.5 rundach, gdyż w pierwszych ośmiu, tożsamych konstrukcyjnie, rundach wykorzystywanych jest sześć 16-bitowych kluczy rundowych, a dziewiąta runda, o innej, prostszej strukturze, nazywana transformacją wyjściową, wykorzystuje tylko cztery klucze rundowe.



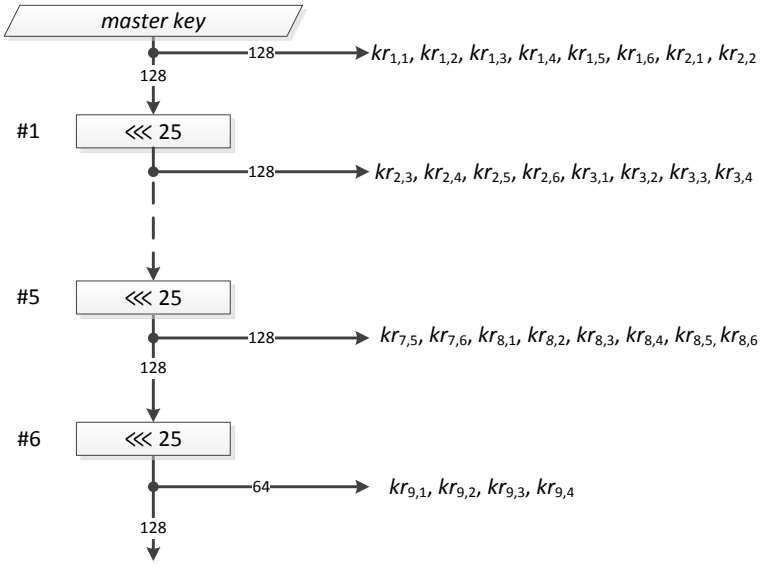
Rys. 3.46. IDEA – struktura funkcji szyfrującej/deszyfrującej

Strukturę funkcji szyfrującej/deszyfrującej przedstawiono na rysunku (rys. 3.46). Podczas deszyfrowania wykorzystywana jest funkcja szyfrująca z odwrotną kolejnością kluczy rundowych, rozumianych jako ciągi podkluczy ($kr_{1,1}$ … $kr_{1,6}$), ($kr_{2,1}$ … $kr_{2,6}$), …, ($kr_{8,1}$ ... $kr_{8,6}$), ($kr_{9,1}$… $kr_{9,4}$), a także z zamianą wartości podkluczy $kr_{i,1}$, $kr_{i,2}$ na $kr_{i,1}^{-1}$, $kr_{i,2}^{-1}$ oraz podkluczy $kr_{i,3}$, $kr_{i,4}$ na $-kr_{i,3}$, $-kr_{i,4}$, gdzie $i = 1, 2,\ldots, 9$.

W funkcji szyfrujące/deszyfrującej wykonywane są następujące operacje:

- $\oplus$ – suma wyłączająca (XOR) słów 16-bitowych,
- $\boxplus$ – dodawanie modulo 2^{16},
- $\odot$ – mnożenie modulo ($2^{16} + 1$), gdzie 16-bitowa wartość 0 reprezentuje liczbę 2^{16}.

Szyfr IDEA zaliczony został do grupy innych szyfrów blokowych, tzn. różniących się od szyfrów Feistela i szyfrów SPN, ponieważ nie wykorzystuje S-bloków reprezentowanych w postaci tablicy. Spośród stosowanych w szyfrze 16-bitowych operacji, największą nieliniowość ma operacja mnożenia $\odot$, której odpowiada S-blok o rozmiarze 32 × 16 bitów – zbyt duży do reprezentowania za pomocą tablicy. Rezultat mnożenia musi być zatem obliczany na bieżąco. Autorzy szyfru określają ideę jego konstrukcji jako „mieszanie operacji z trzech różnych grup algebraicznych”.



Rys. 3.47. IDEA – funkcja generowania kluczy rundowych

W algorytmie generowania kluczy rundowych (algorytm 3.14) i w funkcji przedstawionej na rysunku (rys. 3.47), wykonywana jest operacja:

- ⋘ 25 – cykliczne przesunięcie (rotacja) w lewo słowa 128-bitowego o 25 bitów.

Algorytm 3.14. Algorytm generowania kluczy rundowych szyfru IDEA

WEJŚCIE: klucz główny k o długości 128 bitów

WYJŚCIE: 52 klucze rundowe $(kr_{1,1} \ldots kr_{1,6})$, $(kr_{2,1} \ldots kr_{2,6})$, …, $(kr_{8,1} \ldots kr_{8,6})$, $(kr_{9,1} \ldots kr_{9,4})$ o długości 16 bitów

PROCEDURA:

1. Zdefiniuj funkcję: $I(s) = ((s-1)$ **div** $6 + 1, (s-1)$ **mod** $6 + 1)$, gdzie $s = 1, 2, \ldots, 52$.
2. Podstaw: $K = k$.
3. Dla $i = 0$ do 5 wykonaj:
   - $kr_{I(8i+1)}||kr_{I(8i+2)}||\ldots||kr_{I(8i+8)} = K_1||K_2||\ldots||K_8$,
   - $K \lll 25$.
4. Podstaw: $kr_{9,1}||kr_{9,2}||kr_{9,3}||kr_{9,4} = K_1||K_2||K_3||K_4$.

Wynikiem działania algorytmu generowania kluczy rundowych są 52 klucze rundowe o długości 16 bitów, $kr_{1,1} \ldots kr_{1,6}$, $kr_{2,1} \ldots kr_{2,6}$, …, $kr_{8,1} \ldots kr_{8,6}$, $kr_{9,1} \ldots kr_{9,4}$, uzyskane z klucza głównego.

Tab. 3.42. IDEA – zestawienie cech szyfru

rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego *|k|*	liczba kluczy rundowych *nrk*	długość klucza rundowego *|rk|*	długość próbki kluczy rundowych *|rkp|*
64b	9	128b	52	16b	832b

W tabeli (tab. 3.42) przedstawiono zestawienie podstawowych cech szyfru IDEA.

Tab. 3.43. IDEA – lista operacji

rodzaj	operacja	funkcja szyfrująca/deszyfrująca	algorytm generowania kluczy rundowych
XOR	$\oplus$	tak	nie
dodawanie	$\boxplus$	tak	nie
mnożenie	$\odot$	tak	nie
rotacja	$\lll 25$	nie	tak

W tabeli (tab. 3.43) przedstawiono listę operacji szyfru i algorytmu generowania kluczy rundowych. Operacje $\boxplus$, $\odot$ są operacjami nieliniowymi, a XOR i rotacja są liniowe (formuła (3.17)).

W algorytmie generowania kluczy rundowych szyfru IDEA możliwe jest zwiększenie liczby iteracji. Możliwe jest zatem zwiększenie liczby rund w szyfrze IDEA.

Wniosek 3.28.

W algorytmie generowania kluczy rundowych szyfru IDEA wykorzystywana jest jedna funkcja liniowa. Zwiększenie liczby iteracji tego algorytmu umożliwia zwiększenie liczby rund szyfru IDEA (metoda B).

■

Zdefiniujmy dla rundy i złożony klucz (nadklucz) rundowy $k_i = kr_{i,1}||\ kr_{i,2}||\ldots||kr_{i,6}$, dla $i = 1, 2, \ldots, 8$ oraz $k_i = kr_{i,1}||\ kr_{i,2}||\ldots||kr_{i,4}$, dla $i = 9$.

Klucze rundowe k_i można określić w następujący sposób:

$$k_1 = k[1\text{-}6], \tag{3.21}$$

$$k_2 = k[7\text{-}8]||(k \lll 25)[1\text{-}4], \tag{3.22}$$

$$k_3 = (k \lll 25)[5\text{-}8]||(k \lll 50)[1\text{-}2], \tag{3.23}$$

$$k_4 = (k \lll 50)[3\text{-}8], \tag{3.24}$$

$$k_5 = (k \lll 75)[1\text{-}6], \tag{3.25}$$

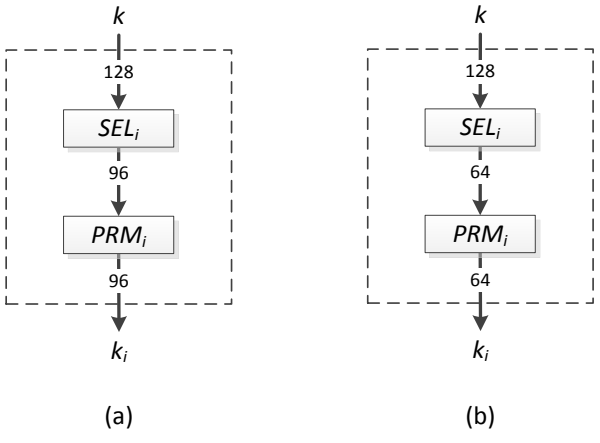
$$k_6 = (k \lll 75)[7\text{-}8]||(k \lll 100)[1\text{-}4], \tag{3.26}$$

$$k_7 = (k \lll 100)[5\text{-}8]||(k \lll 125)[1\text{-}2], \tag{3.27}$$

$$k_8 = (k \lll 125)[3\text{-}8], \tag{3.28}$$

$$k_9 = (k \lll 150)[1\text{-}4]. \tag{3.29}$$

Klucz rundowy k_i, składa się z jednego (i = 1, 4, 5, 8, 9) lub dwóch (i = 2, 3, 6, 7) podciągów, ciągu bitów klucza głównego k, o długości będącej wielokrotnością liczby 16. Każdy bit klucza k_i jest innym bitem klucza głównego – w przypadku dwóch podciągów, podciągi te są rozłączne.



Rys. 3.48. IDEA – schemat zastępczy funkcji generowania klucza rundowego k_i: (a) i = 1, 2, …, 8, (b) i = 9

Niech $k = b_1 b_2 \ldots b_{128}$. W schemacie zastępczym funkcji generowania klucza rundowego k_i, przedstawionym na rysunku (rys. 3.48), wykonywane są operacje:

- SEL_i – funkcja wyboru 96 albo 64 bitów ze 128-bitowego klucza k, np. dla $i = 1$ wybiera $b_1 b_2 \ldots b_{96}$, a dla $i = 2$ wybiera $b_{26} b_{27} \ldots b_{89}$ i $b_{97} b_{98} \ldots b_{128}$,
- PRM_i – permutacja uporządkowania wybranych bitów na właściwych pozycjach klucza k_i, np. dla $i = 1$: $k_1 = b_1 b_2 \ldots b_{96}$, a dla $i = 2$: $k_2 = b_{97} b_{98} \ldots b_{128} || b_{26} b_{27} \ldots b_{89}$.

Wniosek 3.29.

W szyfrze IDEA, klucz rundowy k_i zależy od 96 ($i = 1, 2, \ldots, 8$) lub 64 ($i = 9$) bitów 128-bitowego klucza głównego k, przy czym każdy bit klucza k_i jest pewnym bitem klucza k, a tym samym jest zależny od jednego bitu klucza k.

■

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru IDEA, przedstawiono w postaci charakterystyki (charakterystyka 3.14).

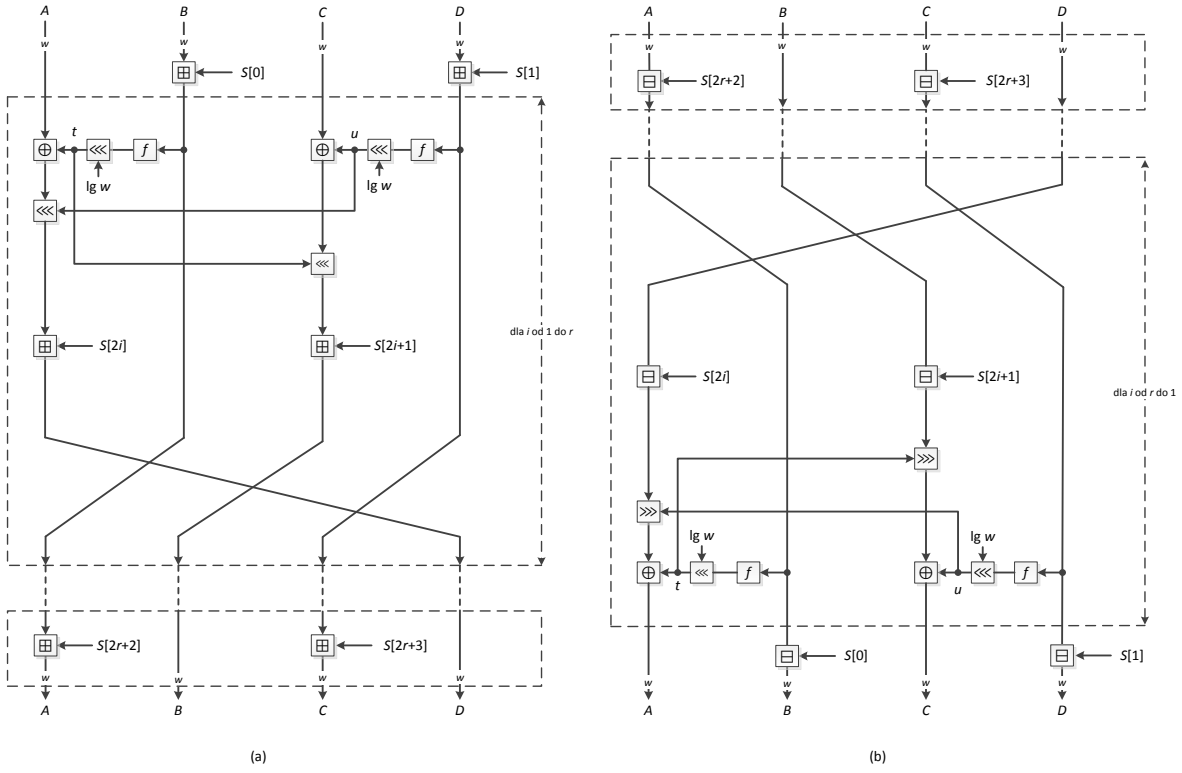
Charakterystyka 3.14. Charakterystyka algorytmu generowania kluczy rundowych szyfru IDEA

1. Długość klucza głównego: $|k| = 128b$.
2. Długość klucza rundowego: $|k_i| = 96b$ ($i = 1, 2, \ldots, 8$) lub $|k_i| = 64b$ ($i = 9$).
3. Liczba iteracji: $R = 6$.
4. Zależność k_i od bitów k:
   - $k_1, k_2, \ldots, k_8$: 96b / 128b (75.00%),
   - k_9: 64b / 128b (50.00%).
5. Zależność bitu k_i od bitów k:
   - $k_1, k_2, \ldots, k_9$: 1b / 128b (0.78%).
6. Zależność grupy bitów k_i od grupy bitów k: **32b, 64b, 96b**.
7. Operacja liniowe: $\lll b$.
8. Operacje nieliniowe: **BRAK**.
9. Możliwości zwiększenia liczby iteracji: **TAK**.
10. Indywidualizacja iteracji: **NIE**.
11. Teoretyczna ocena jakości algorytmu: **0.78%**.

3.4.2. RC6 (1998)

Szyfr RC6 [94], jeden z pięciu finalistów konkursu na standard AES, jest skalowalnym szyfrem blokowym, określonym przez parametry: *w*/*r*/*b*, gdzie 4*w* określa rozmiar *n* bloku (*A*, *B*, *C*, *D*) w bitach, *r* określa liczbę rund szyfru, a *b* określa długość klucza głównego, *k*, wyrażoną w bajtach – |*k*| = 8*b*. Na przykład wariantem szyfru rozpatrywanym w konkursie na standard AES był RC6-32/20/*b*, gdzie *b* = 16, 24, 32, tj. szyfr RC6 przetwarzający 128-bitowe bloki danych, o dwudziestu rundach, z wykorzystaniem 128-, 192-, 256-bitowego klucza głównego. Pozostałe możliwe wartości parametrów *w*/*r*/*b* to:

- *w* = 8, 16, 32, 64, …, zależnie od architektury CPU (w konkursie na AES *w* = 32),
- *r* = 1, 2, …, 255 (w konkursie na AES *r* = 20),
- *b* = 0, 1, …, 255, tj. 0 – 2040 bitów (w konkursie na AES *b* = 16, 24, 32), dla *b* = 0 klucz *k* jest ciągiem pustym, tzn. o zerowej długości i odpowiada mu wartość 0.



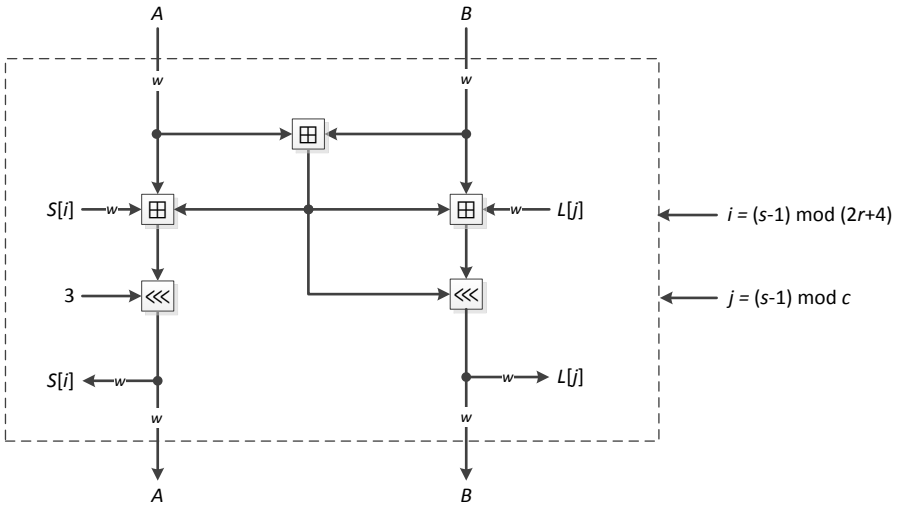
Rys. 3.49. RC6 – struktura funkcji: (a) szyfrującej (b) deszyfrującej

Strukturę funkcji szyfrującej w szyfrze RC6 przedstawiono na rysunku (rys. 3.49 (a)). W rundzie *i* (*i* = 1, 2, …, *r*) wykorzystywane są dwa klucze rundowe *S*[2*i*] i *S*[2*i*+1], w transformacji początkowej – klucze *S*[0] i *S*[1], a w transformacji końcowej – klucze *S*[2*r*+2] i *S*[2*r*+3]. W sumie więc wykorzystywane są 2*r* + 4 klucze rundowe o długości *w* bitów.

Pomimo podobieństwa do sieci Feistela, podczas deszyfrowania wykorzystywana jest funkcja deszyfrująca, przedstawiona na rysunku (rys. 3.49 (b)). W funkcji szyfrującej wykonywane są następujące operacje:

- $\boxplus$ – dodawanie modulo 2^w słów w-bitowych,
- $\oplus$ – suma wyłączająca (XOR) słów w-bitowych,
- f – funkcja nieliniowa taka, że: $f(x) = x \otimes (2 \otimes x \boxplus 1)$,
- $\otimes$ – mnożenie modulo 2^w słów w-bitowych,
- lg w – logarytm o podstawie 2 z liczby w, tj. $\log_2 w$,
- $\lll b$ – cykliczne przesunięcie (rotacja) w lewo słowa w-bitowego o b bitów, gdzie $b = \lg w$,
- $\lll_d b$ – cykliczne przesunięcie (rotacja) w lewo, sterowana danymi, słowa w-bitowego o liczbę bitów $b = 0, 1, \ldots, 2^d - 1$, określoną przez wartość $d = \lg w$ najmniej znaczących bitów słowa sterującego.

Szyfr RC6 zaliczono do innych szyfrów blokowych z powodu zastąpienia S-bloków reprezentowanych w postaci tablicy, obliczaną na bieżąco nieliniową funkcją f. Nowatorskim rozwiązaniem Autorów jest również wykorzystanie rotacji sterowanych danymi.



Rys. 3.50. RC6 – iteracja s funkcji generowania kluczy rundowych ($s = 1, 2, \ldots, v$), gdzie $v = 3 \cdot \max(c, 2r + 4)$ i $c = \lceil 8b / w \rceil$

W algorytmie generowania kluczy rundowych (algorytm 3.15) i w funkcji przedstawionej na rysunku (rys. 3.50), wykonywane są operacje:

- $\boxplus$ – dodawanie modulo 2^w słów w-bitowych,

- $\lll b$ – cykliczne przesunięcie (rotacja) w lewo słowa w-bitowego o b bitów, gdzie $b = 3$,
- $\lll_d b$ – cykliczne przesunięcie (rotacja) w lewo, sterowana danymi, słowa w-bitowego o liczbę bitów $b = 0, 1, \ldots, 2^d - 1$, określoną przez wartość $d = \lg w$ najmniej znaczących bitów słowa sterującego,
- **max** – operacja wyznaczania największej wartości w zbiorze liczb,
- **mod** – operacja wyznaczania reszty z dzielenia całkowitego.

Ponadto w algorytmie (algorytm 3.15) wykorzystuje się dwie tzw. „magiczne stałe" (ang. „*magic constants*"):

- P_{32} = 0xB7E15163 – uzyskana z binarnego rozwinięcia $e - 2$, gdzie e to liczba Eulera,
- Q_{32} = 0x9E3779B9 – uzyskana z binarnego rozwinięcia $\Phi - 1$, gdzie Φ to „złota liczba" (ang. *golden ratio*).

Algorytm 3.15. Algorytm generowania kluczy rundowych szyfru RC6-*w*/*r*/*b*

WEJŚCIE: klucz główny k o długości b bajtów, w postaci tablicy o rozmiarze c w-bitowych słów $L[0], L[1], \ldots, L[c-1]$, gdzie $c = \lceil 8 \cdot b / w \rceil$ dla $b > 0$ i $c = 1$, dla $b = 0$

WYJŚCIE: klucze rundowe $S[0], S[1], \ldots, S[2r+3]$ o długości w bitów

PROCEDURA:

1. Uzupełnij $L[c-1]$ (klucz główny) zerami, na najbardziej znaczących pozycjach, do długości w ($c \cdot w$) bitów.
2. Przypisz do $S[0]$ wartość stałej P_w, tj. $S[0] = P_w$.
3. Dla $i = 1$ do $2r + 3$ przypisz do $S[i]$ wartość sumy $S[i-1]$ i stałej Q_w:
   - $S[i] = S[i-1] \boxplus Q_w$.
4. Wyzeruj zmienne A, B, i, j, tj. $A = B = i = j = 0$.
5. Oblicz v takie, że $v = 3 \cdot \mathbf{max}\{c, 2r + 4\}$.
6. W iteracjach od $s = 1$ do v oblicz:
   - $A = S[i] = (S[i] \boxplus A \boxplus B) \lll 3$,
   - $B = L[j] = (L[j] \boxplus A \boxplus B) \lll_d (A \boxplus B)$,
   - $i = (i + 1)$ **mod** $(2r + 4)$,
   - $j = (j + 1)$ **mod** c.

W tabeli (tab. 3.44) przedstawiono zestawienie podstawowych cech szyfru RC6-32/20/*b*, tj. wariantu zgłoszonego do konkursu na standard AES.

Tab. 3.44. RC6-32/20/*b* – zestawienie cech szyfru

rozmiar bloku danych *n*	liczba rund *r*	długość klucza głównego |*k*|	liczba kluczy rundowych *nrk*	długość klucza rundowego |*rk*|	długość próbki kluczy rundowych |*rkp*|
128b	20	128b	$2r + 4$	$n / 4$	$n(2r + 4) / 4$
		192b			
		256b			

W tabeli (tab. 3.45) przedstawiono listę operacji funkcji szyfrującej i algorytmu generowania kluczy rundowych.

Tab. 3.45. RC6 – lista operacji

rodzaj	operacja	funkcja szyfrująca	algorytm generowania kluczy rundowych
XOR	$\oplus$	tak	nie
dodawanie modulo 2^w	$\boxplus$	tak	tak
mnożenie modulo 2^w	$\otimes$	tak	nie
logarytm	lg *w*	tak	tak
rotacja	$\lll b$	tak	tak
rotacja sterowana danymi	$\lll_d b$	tak	tak
wartość maksymalna	**max**	nie	tak
reszta z dzielenia całkowitego	**mod**	nie	tak

Wniosek 3.30.

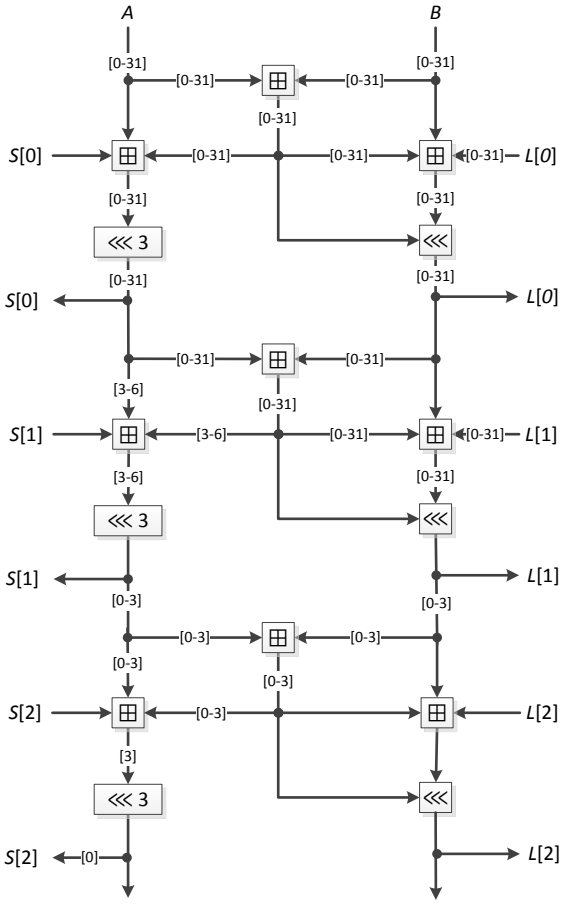
W algorytmie generowania kluczy rundowych szyfru RC6 wykorzystywane są operacje nieliniowe, tj. dodawanie modulo 2^w i rotacja sterowana danymi oraz liniowe – rotacja. Operacje wyznaczania reszty z dzielenia całkowitego i wartości maksymalnej, jako służące do obliczania indeksów, pominięto. Podobnie pominięto operację lg *w*, definiującą liczbę bitów. W algorytmie generowania kluczy rundowych możliwe jest zwiększenia liczby iteracji (metoda B) i możliwe jest odpowiednie zwiększenie liczby rund szyfru.

■

Na rysunku (rys. 3.51) przedstawiono zależność bitu *S*[2][0] podklucza rundowego od bitów klucza głównego $k = L[0]||L[1]||L[2]||L[3]$. Przy określeniu tej zależności uwzględniono następujące własności funkcji elementarnych:

- $\boxplus$ – bit *i* wyjścia funkcji dodawania modulo 2^l zależy od bitów *i*, *i* – 1, …, 0 wejść, gdzie bit numer 0 jest najmniej znaczący,

- $\lll b$ – bit i wyjścia rotacji w lewo o b bitów słowa l-bitowego, w przypadku LSB = 0 po lewej, zależy od bitu (i + b) **mod** l wejścia,
- $\lll_d b$ – bit i wyjścia rotacji w lewo, sterowanej d-bitowymi danymi, słowa l-bitowego, w przypadku LSB = 0 po lewej, zależy od bitów: i, (i + 1) **mod** l, (i + + 2) **mod** l, …, (i + 2^d – 1) **mod** l wejścia.



Rys. 3.51. RC6-32/20/16 – zależność bitu *S*[2][0] podklucza rundowego od bitów klucza głównego *k* = *L*[0]||*L*[1]||*L*[2]||*L*[3]

Ostatecznie bit $S[2][0]$ zależy od bitów $L[0][0\text{-}31]$ i $L[1][0\text{-}31]$, a ponieważ $L[0]||L[1]||L[2]||L[3] = k$ to bit $S[2][0]$ zależy od 64 bitów 128-bitowego klucza głównego k, co stanowi 50%. Z rysunku (rys. 3.51) można też odczytać, że:

- bit $S[1][0]$ zależy od 32 bitów,
- bit $S[0][0]$ zależy od 0 bitów

oraz wnioskować, że:

- bit $S[3][0]$ zależy od 96 bitów,
- bit $S[4][0]$, $S[5][0]$, …, $S[2r+3][0]$ zależy od 128 bitów.

Te same wyniki uzyskiwane są dla dowolnego bitu kluczy rundowych $S[0]$, $S[1]$, …, $S[2r+3]$.

Wniosek 3.31.

W szyfrze RC6-32/20/16 (nad)klucz rundowy $k_i = S[2i]||S[2i+1]$ ($i = 0, 1, \ldots, r + 1$) zależy od liczby bitów: 32 ($i = 0$), 96 ($i = 1$) lub 128 ($i = 2, 3, \ldots, r + 1$) 128-bitowego klucza głównego k, przy czym bit (nad)klucza k_i, zależy od liczby bitów klucza k: 32 ($i = 0$), 96 ($i = 1$) lub 128 ($i = 2, 3, \ldots, r + 1$).

■

W przedstawionej analizie rozważono pierwszą z trzech faz obliczania kluczy rundowych ($R = 3(2r + 4)$). Po trzech fazach, bit (nad)klucza k_i zależy od wszystkich bitów klucza głównego k dla $i = 0, 1, \ldots, r + 1$.

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru RC6-w/r/b przedstawiono w postaci charakterystyki (charakterystyka 3.15).

Charakterystyka 3.15. Charakterystyka algorytmu generowania kluczy rundowych szyfru RC6-*w*/*r*/*b*

1. Długość klucza głównego: $|k| = 8 \cdot b$ **bitów**.
2. Długość klucza rundowego: $|k_i| = 2 \cdot w$ **bitów** ($i = 0, 1, \ldots, r + 1$)
3. Liczba iteracji ($c = \lceil 8 \cdot b / w \rceil$): $R = 3 \cdot \max\{c, 2r + 4\}$.
4. Zależność $k_i = S[2i]||S[2i+1]$ od bitów k (RC6-32/20/16):
   - k_0: **32b / 128b (25.00%),**
   - k_1: **96b / 128b (75.00%),**
   - $k_2, k_3, \ldots, k_{r+1}$: **128b / 128b (100.00%).**
5. Zależność bitu $k_i = S[2i]||S[2i + 1]$ od bitów k (RC6-32/20/16):
   - k_0: **32b / 128b (25.00%),**
   - k_1: **96b / 128b (75.00%),**
   - $k_2, k_3, \ldots, k_{r+1}$: **128b / 128b (100.00%).**
6. Zależność grupy bitów K_i od grupy bitów k: **NIE**.
7. Operacja liniowe: $\lll b$.
8. Operacje nieliniowe: $\boxplus$, $\lll_d b$.
9. Możliwości zwiększenia liczby iteracji: **TAK**.
10. Indywidualizacja iteracji: **przez stałe** $P_w \boxplus i \cdot Q_w$.
11. Teoretyczna ocena jakości algorytmu: **80.00%**.

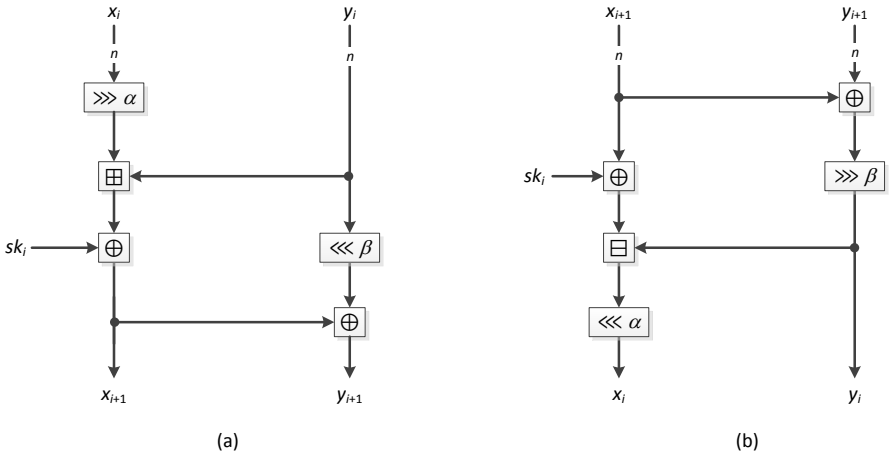
3.4.3. Speck (2013)

Szyfr Speck [12][13] jest szyfrem typu ARX (ang. *Addition, Rotation, XOR*), został opublikowany przez NSA w 2013 roku i należy do grupy lekkich szyfrów blokowych o dużej wydajności, z przeznaczeniem do zastosowań sprzętowych. Szyfr Speck operuje na blokach o rozmiarze $n' = 32, 48, 64, 96$ lub 128 bitów, dzielonych zawsze na dwa równe, n-bitowe podbloki. Klucz główny k szyfru może przyjmować długość $|k| = m \cdot n$, gdzie $m = 2, 3$ lub 4. Możliwe warianty szyfru, oznaczane jako Speck2n/mn oraz ich parametry, przedstawiono w tabeli (tab. 3.46).

Tab. 3.46. Speck – parametry zależne od wariantu szyfru

wariant szyfru	rozmiar bloku danych	długość klucza	liczba rund r	m	α	β
Speck32/64	2 · 16b = 32b	4 · 16b = 64b	22	4	7	2
Speck48/72	2 · 24b = 48b	3 · 24b = 72b	22	3	8	3
Speck48/96		4 · 24b = 96b	23	4	8	3
Speck64/96	2 · 32b = 64b	3 · 32b = 96b	26	3	8	3
Speck64/128		4 · 32b = 128b	27	4	8	3
Speck96/96	2 · 48b = 96b	2 · 48b = 96b	28	2	8	3
Speck96/144		3 · 48b = 144b	29	3	8	3
Speck128/128	2 · 64b = 128b	2 · 64b = 128b	32	2	8	3
Speck128/192		3 · 64b = 192b	33	3	8	3
Speck128/256		4 · 64b = 256b	34	4	8	3

Liczba r rund szyfru zależy od rozmiaru bloku n i długości klucza głównego $|k|$, a w rundzie i ($i = 0, 1, \ldots, r - 1$) wykorzystywany jest jeden klucz rundowy sk_i, o długości n bitów. Szyfr Speck nie jest inwolucyjny – podczas deszyfrowania stosowane są funkcje odwrotne i wykonywane są one w odwrotnej kolejności. Klucze rundowe wówczas podawane są też w odwrotnej kolejności.

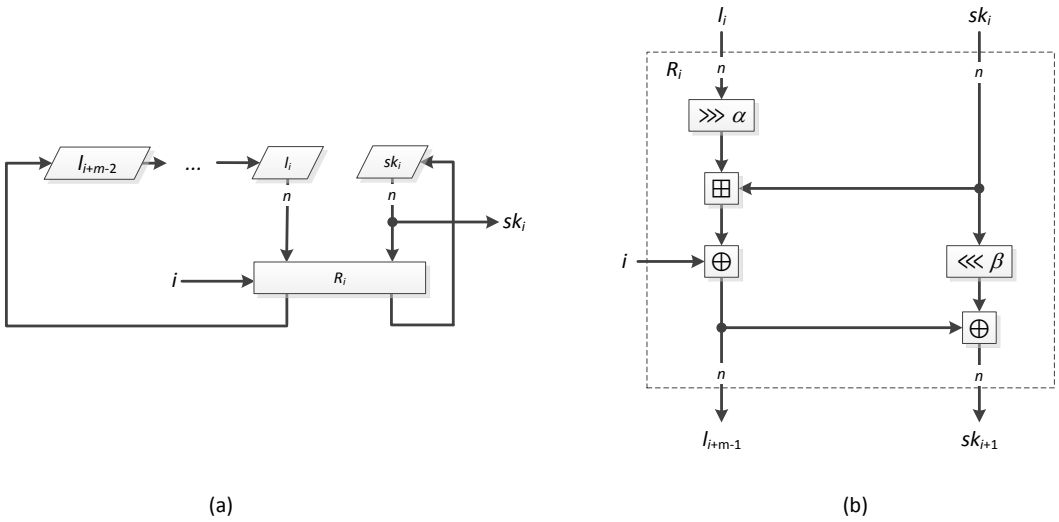


Rys. 3.52. Speck – struktura rundy i ($i = 0, 1, \ldots, r - 1$): (a) szyfrowanie, (b) deszyfrowanie

Strukturę pojedynczej rundy szyfru Speck, podczas szyfrowania i deszyfrowania, przedstawiono na rysunku (rys. 3.52). W funkcji szyfrujące stosowane są wyłącznie operacje ARX:

- $\boxplus$ – dodawanie modulo 2^n słów n-bitowych,
- $\ggg \alpha$ – cykliczne przesunięcie (rotacja) w prawo słowa n-bitowego o α bitów (tab. 3.46),
- $\lll \beta$ – cykliczne przesunięcie (rotacja) w lewo słowa n-bitowego o β bitów (tab. 3.46),
- $\oplus$ – suma wyłączająca (XOR) słów n-bitowych.

Szyfr Speck należy do innych szyfrów blokowych ponieważ jest szyfrem ARX – nie wykorzystuje S-bloków o dużej nieliniowości.



Rys. 3.53. Speck – struktura: (a) iteracji *i* funkcji generowania kluczy rundowych, (b) funkcji R_i (i = 0, 1, …, r – 1)

Na rysunku (rys. 3.53) przedstawiono strukturę pojedynczej iteracji funkcji generowania kluczy rundowych w szyfrze Speck. Struktura ta składa się z zestawu m rejestrów n-bitowych i z funkcji R_i – identycznej z funkcją szyfrującą rundy (rys. 3.52 (a)). Początkowo do rejestrów wpisywany jest klucz główny $k = k_{m-1}||k_{m-2}||\ldots||k_0$, którego n najmniej znaczących bitów, tj. podklucz k_0, stanowi klucz rundowy sk_0. Funkcja R_0 oblicza wartości l_{m-1} i sk_1, wpisywane następnie do odpowiednich rejestrów z jednoczesnym przesunięciem ich zawartości, zgodnie ze schematem. Klucz rundowy sk_1 zależy od podkluczy k_1 i k_0, podobnie jak wartość l_{m-1}. Obliczenie jest kontynuowane aż do wpisania do rejestrów wartości l_{r+m-3} i sk_{r-1}.

Algorytm 3.16. Algorytm generowania kluczy rundowych szyfru Speck

WEJŚCIE: klucz główny $k = k_{m-1}||k_{m-2}||\ldots||k_0$ o długości $m \cdot n$ bitów

WYJŚCIE: klucze rundowe $sk_0, sk_1, \ldots, sk_{r-1}$ o długości n bitów

PROCEDURA:

1. Jako $l_{m-2}||l_{m-3}||\ldots||l_0||sk_0$ podstaw k.
2. Dla $i = 0$ do $r - 2$ oblicz:
   - $l_{i+m-1} = (sk_i \boxplus (l_i \ggg \alpha)) \oplus i$,
   - $sk_{i+1} = (sk_i \lll \beta) \oplus l_{i+m-1}$.

W algorytmie generowania kluczy rundowych szyfru Speck (algorytm 3.16) wykorzystywane są te same operacje ARX co w funkcji szyfrującej.

Tab. 3.47. Speck – zestawienie cech szyfru

rozmiar bloku danych n'	liczba rund r	długość klucza głównego $\|k\|$	liczba kluczy rundowych nrk	długość klucza rundowego $\|rk\|$	długość próbki kluczy rundowych $\|rkp\|$
32b	(22)	(64b)	r	16b	$r \cdot n' / 2$
48b	(22, 23)	(72b, 96b)		24b	
64b	(26, 27)	(96b,128b)		32b	
96b	(28, 29)	(96b,144b)		48b	
128b	(32, 33, 34)	(128b,192b, 256b)		64b	

W tabeli (tab. 3.47) przedstawiono zestawienie podstawowych cech szyfru Speck.

Tab. 3.48. Speck – lista operacji

rodzaj	operacja	funkcja szyfrująca	algorytm generowania kluczy rundowych
XOR	$\oplus$	tak	tak
dodawanie modulo 2^n	$\boxplus$	tak	tak
rotacja	$\ggg \alpha$	tak	tak
rotacja	$\lll \beta$	tak	tak

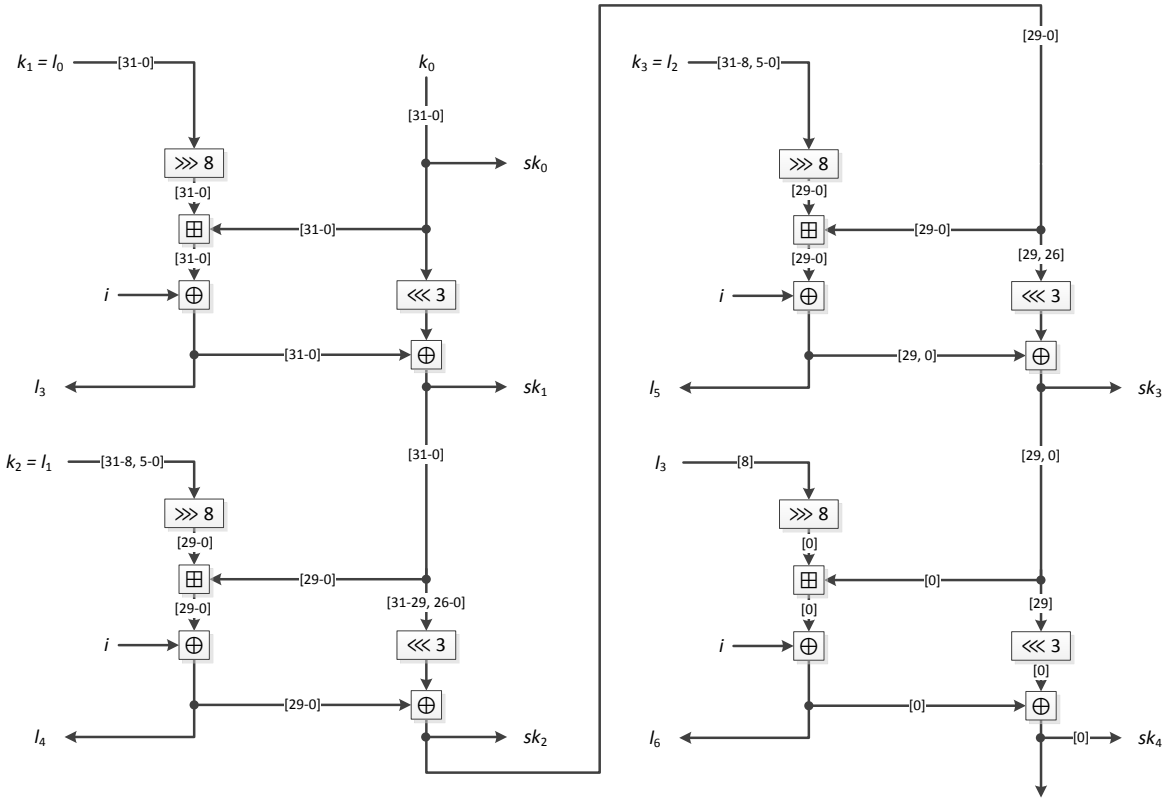
W tabeli (tab. 3.48) przedstawiono listę operacji funkcji szyfrującej i algorytmu generowania kluczy rundowych. Operacja $\boxplus$ jest nieliniowa, a XOR i rotacje są liniowe (formuła (3.17)).

W algorytmie generowania kluczy rundowych szyfru Speck możliwe jest zwiększenie liczby iteracji. Możliwe jest zatem zwiększenie liczby rund w szyfrze Speck.

Wniosek 3.32.

W szyfrze Speck stosowane są operacje nieliniowa ($\boxplus$) i liniowe ($\ggg \alpha$, $\lll \beta$, $\oplus$). Zwiększenie liczby iteracji algorytmu generowania kluczy rundowych szyfru Speck umożliwia zwiększenie liczby rund tego szyfru (metoda B).

■



Rys. 3.54. Speck64/128 – zależność bitu $sk_4[0]$ klucza rundowego od bitów klucza głównego $k = k_3||k_2||k_1||k_0$

Na rysunku (rys. 3.54) przedstawiono zależność bitu $sk_4[0]$ klucza rundowego od bitów klucza głównego $k = k_3||k_2||k_1||k_0$. Przy określeniu tej zależności uwzględniono następujące własności funkcji elementarnych:

- $\boxplus$ – bit i wyjścia funkcji dodawania modulo 2^l zależy od bitów i, $i - 1$, …, 0 wejść, gdzie bit numer 0 jest najmniej znaczący,
- $\ggg \alpha$ – bit i wyjścia rotacji w prawo o b bitów słowa l-bitowego, w przypadku LSB = 0 po prawej, zależy od bitu $(i + \alpha)$ **mod** l wejścia,
- $\lll \beta$ – bit i wyjścia rotacji w lewo o b bitów słowa l-bitowego, w przypadku LSB = 0 po prawej, zależy od bitu $(i + (l - \beta))$ **mod** l wejścia,
- $\oplus$ – bit i wyjścia (wartości) funkcji XOR zależy od bitu i wszystkich wejść (argumentów).

Ostatecznie bit $sk_4[0]$ zależy od bitów k_0[31-0], k_1[31-0], k_2[31-8, 5-0], k_3[31-8, 5-0], a ponieważ $k_3||k_2||k_1||k_0 = k$ to bit $sk_4[0]$ zależy od 124 bitów 128-bitowego klucza głównego k, co stanowi 96.88%.

Z rysunku (rys. 3.54) można też odczytać, że:

- bit $sk_3[0]$ zależy od 93 bitów,
- bit $sk_2[0]$ zależy od 61 bitów,
- bit $sk_1[0]$ zależy od 3 bitów,
- bit $sk_0[0]$ zależy od 1 bitu.

Bit $sk_4[0]$, z uwagi na operację ⊞, stanowi najgorszy przypadek, tj. zależy od możliwie najmniejszej liczby bitów klucza głównego k.

Wniosek 3.33.

W szyfrze Speck64/128 klucz rundowy sk_i (i = 0, 1, …, 26) zależy od liczby bitów: 32 (i = 0), 64 (i = 1), 96 (i = 2) lub 128 (i = 3, 4, …, 26) 128-bitowego klucza głównego k, przy czym bit klucza sk_i, w najgorszym przypadku, zależy od liczby bitów klucza k: 1 (i = 0), 3 (i = 1), 61 (i = 2), 93 (i = 3), 124 (i = 4).

■

Zestawienie podstawowych cech algorytmu generowania kluczy rundowych szyfru Speck przedstawiono w postaci charakterystyki (charakterystyka 3.16).

Charakterystyka 3.16. Charakterystyka algorytmu generowania kluczy rundowych szyfru Speck

1. Długość klucza głównego: $|\boldsymbol{k}| = \boldsymbol{m} \cdot \boldsymbol{n}$ **bitów**.
2. Długość klucza rundowego: $|\boldsymbol{sk_i}| = \boldsymbol{n}$ **bitów** ($\boldsymbol{i = 0, 1, \ldots, r - 1}$).
3. Liczba iteracji: $\boldsymbol{R = r - 1}$.
4. Zależność sk_i = od bitów k (Speck64/128):
   - $\boldsymbol{sk_0}$**: 32b / 128b (25.00%),**
   - $\boldsymbol{sk_1}$**: 64b / 128b (50.00%),**
   - $\boldsymbol{sk_2}$**: 96b / 128b (75.00%),**
   - $\boldsymbol{sk_3, sk_4, \ldots, sk_{26}}$**: 128b / 128b (100.00%).**
5. Zależność bitu sk_i od bitów k (Speck64/128):
   - $\boldsymbol{sk_0}$**[0]: 1b / 128b (0.78%),**
   - $\boldsymbol{sk_1}$**[0]: 3b / 128b (2.34%),**
   - $\boldsymbol{sk_2}$**[0]: 61b / 128b (47.66%),**

- **sk_3[0]: 93b/128b (72.66%),**
- **sk_4[0]: 124b / 128b (96.68%).**

6. Zależność grupy bitów sk_i od grupy bitów k: **NIE**.
7. Operacja liniowe: $\oplus$, $\ggg \alpha$, $\lll \beta$.
8. Operacje nieliniowe: $\boxplus$.
9. Możliwość zwiększenia liczby iteracji: **TAK**.
10. Indywidualizacja iteracji: **przez licznik iteracji *i***.
11. Teoretyczna ocena jakości algorytmu: **44.06%**.

3.5. Analiza porównawcza algorytmów generowania kluczy rundowych

W tabelach (tab. 3.49, tab. 3.50) przedstawiono zestawienie cech algorytmów generowania kluczy rundowych, nazywanych dalej krótko algorytmami, w rozpatrywanych w niniejszym rozdziale szyfrach.

Tab. 3.49. Zestawienie cech algorytmów generowania kluczy rundowych w szyfrach o konstrukcji sieci Feistela (*n* – rozmiar bloku, *r* – liczba rund)

Lp.	Cecha	DES	3DES	LOKI97	SM4	KASUMI	FeW	LCB-IoT
1.	długość klucza głównego |*k*|	64b	192b	128b 192b 256b	128b	128b	80b 128b	80b
2.	liczba kluczy rundowych *nrk*	16	48	48	32	64	32	32
3.	długość klucza rundowego |*rk*|	48b	48b	64b	32b	16b	32b	16b
4.	liczba iteracji *R*	16	48	48	32	8	64	32
5.	długość słowa |*wd*|	2·28b	2·28b	4·64b	4·32b	8·16b	80b 128b	2·16b
6.	uzupełnienie klucza głównego	nie	nie	tak	nie	nie	nie	nie
7.	indywidualizacja iteracji	β_i	β_i	C_i	CK_i	Cj	i	-
8.	iteracje nadmiarowe lub rozbiegowe	nie	nie	nie	nie	nie	nie	nie
9.	bezpośrednie użycie klucza głównego	nie	nie	nie	nie	nie	tak	tak
10.	skalowalność	nie	nie	nie	nie	nie	nie	nie
11. operacje liniowe	⊕ – XOR	nie	nie	tak	tak	tak	tak	tak
	P – permutacja bitowa	nie	nie	tak	nie	nie	nie	tak
	⋘ *b* – rotacja	tak	tak	nie	tak	tak	tak	tak
	P_w – permutacja z wyborem	tak	tak	nie	nie	nie	nie	nie
	P_k – permutacja z kluczem	nie	nie	tak	nie	nie	nie	nie
	E – rozszerzenie	nie	nie	tak	nie	nie	nie	nie
12. operacje nieliniowe	*S* – podstawienie	nie	nie	tak	tak	nie	tak	tak
	⊞ – dodawanie modulo	nie	nie	tak	nie	nie	nie	nie
	⊟ – odejmowanie modulo	nie	nie	nie	nie	nie	nie	nie
	^ - AND	nie	nie	tak	nie	nie	nie	nie
	v - OR	nie	nie	tak	nie	nie	nie	nie
	$\lll_d$ *b* – rotacja sterowana danymi	nie	nie	nie	nie	nie	nie	nie
13.	zależność klucza rundowego k_i (5 kluczy)	75%	25%	100%	100%	100%	47.97%	20%
14.	zależność bitu klucza rundowego k_i (5 kluczy)	1.56%	0.52%	92.81%	90.16%	0.78%	9.53%	1.25%
15.	zależność grupy bitów klucza rundowego k_i	nie	nie	nie	nie	16b	nie	nie
16.	teoretyczna ocena jakości algorytmu	1.56%	0.52%	92.81%	90.16%	0.78%	9.53%	1.25%
17.	dł. próbki kluczy rundowych w bitach |*rkp*|	768b	2304b	3072b	1024b	1024b	1024b	512b
18.	możliwość zwiększenia liczby iteracji	nie	nie	tak	nie	nie	war.	tak

Tab. 3.50. Zestawienie cech algorytmów generowania kluczy rundowych w szyfrach o konstrukcji SPN lub innej (*n* – rozmiar bloku, *r* – liczba rund)

Lp.	Cecha	AES	Serpent	SAFER+	PRESENT	PP-1	PP-2	IDEA	RC6	Speck
1.	**długość klucza głównego |*k*|**	128b 192b 256b	128b 192b 256b	128b 192b 256b	80b 128b	n $2n$	$d{\cdot}64$	128b	128b 192b 256b	$m{\cdot}n'$
2.	**liczba kluczy rundowych *nrk***	11 13 15	33	17 25 33	32	$2r$	r	52	44	r
3.	**długość klucza rundowego |*rk*|**	128b	128b	128b	64b	n	n	16b	32b	$n'=n/2$
4.	**liczba iteracji *R***	40 46 52	132	16 24 32	31	$2r+1$	$r+\lceil d/t \rceil{\cdot}t$	6	132	$r-1$
5.	**długość słowa |*wd*|**	4·32b	8·32b	16·8b	80b 128b	n	n	128b	2·32b	$2n'$
6.	**uzupełnienie klucza głównego**	nie	tak	nie	nie	nie	tak	nie	tak	nie
7.	**indywidualizacja iteracji**	$Rcon_i$	$\Phi \oplus i$	B_i	i	K_i	K_i	-	$P_w \boxplus i \cdot Q_w$	i
8.	**iteracje nadmiarowe lub rozbiegowe**	nie	nie	nie	nie	tak	tak	nie	tak	nie
9.	**bezpośrednie użycie klucza głównego**	tak	nie	tak	tak	nie	nie	tak	nie	tak
10.	**skalowalność**	nie	nie	nie	nie	tak	tak	nie	tak	tak
11. operacje liniowe	**⊕ – XOR**	tak	tak	tak	tak	tak	tak	nie	nie	tak
	***P* – permutacja bitowa**	nie	tak	nie	nie	tak	tak	nie	nie	nie
	⋘ *b* – rotacja	tak	tak	tak	tak	tak	tak	tak	nie	tak
	***P*$_w$ – permutacja z wyborem**	nie	nie	nie	nie	nie	nie	nie	nie	nie
	***P*$_k$ – permutacja z kluczem**	nie	nie	nie	nie	nie	nie	nie	nie	nie
	***E* – rozszerzenie**	nie	nie	tak	nie	nie	nie	nie	nie	nie
12. operacje nieliniowe	***S* – podstawienie**	tak	tak	nie	tak	tak	tak	nie	nie	nie
	⊞ – dodawanie modulo	nie	nie	tak	nie	tak	tak	nie	tak	tak
	⊟ – odejmowanie modulo	nie	nie	nie	nie	tak	nie	nie	nie	nie
	^ - AND	nie	nie	nie	nie	tak	nie	nie	nie	nie
	v - OR	nie	nie	nie	nie	nie	nie	nie	nie	nie
	⋘$_d$ *b* – rotacja sterowana danymi	nie	nie	nie	nie	tak	nie	nie	tak	nie
13.	**zależność klucza rundowego k_i (5 kluczy)**	100%	100%	100%	50%	100%	90%	75%	80%	70%
14.	**zależność bitu klucza rundowego k_i (5 kluczy)**	50.63%	39.22%	0.78%	0.78%	83.75%	90%	0.78%	80%	44.06%
15.	**zależność grupy bitów klucza rundowego k_i**	nie	nie	8b	nie	nie	nie	32b 64b 96b	nie	nie
16.	**teoretyczna ocena jakości algorytmu**	50.63%	39.22%	0.78%	0.78%	83.75%	90%	0.78%	80%	44.06%
17.	**długość próbki kluczy rundowych w bitach |*rkp*|**	1408b 1664b 1920b	4224b	2176b 3200b 4224b	2048b	$2r{\cdot}n$	$r{\cdot}n$	832b	1408b	$r{\cdot}n$
18.	**możliwość zwiększenia liczby iteracji**	tak	tak	tak	war.	tak	tak	tak	tak	tak

W dalszej części niniejszego podrozdziału, omówiono cechy przedstawione w powyższych tabelach.

Cecha 1. Długość klucza głównego w bitach – algorytmy są zaprojektowane dla długości klucza:

- jednej (DES, 3DES, SM4, KASUMI, LCB-IoT, IDEA),
- kilku (LOKI97, FeW, AES, Serpent, SAFER+, PRESENT),
- wielu (PP-1, PP-2, RC6, Speck).

Cecha 2. Liczba kluczy rundowych:

- stała (DES, 3DES, LOKI97, SM4, FeW, LCB-IoT, Serpent, PRESENT, IDEA),
- zależna od długości klucza (AES, SAFER+),
- zależna tylko od rozmiaru n bloku (PP-1),
- zależna od rozmiaru bloku n i długości klucza głównego k (PP-2, Speck – przez zależność liczby rund r),
- dowolna (RC6 – liczba rund $r = 1, 2, \ldots, 255$).

Ponadto na rundę szyfru może przypadać klucz rundowy:

- jeden (DES, 3DES, SM4, FeW, LCB-IoT, AES, Serpent, PRESENT, PP-2, Speck),
- dwa (SAFER+, PP-1, RC6),
- trzy (LOKI97),
- cztery lub sześć (IDEA),
- osiem (KASUMI).

Cecha 3. Długość klucza rundowego:

- stała, równa n (AES, Serpent, SAFER+, PRESENT),
- stała, pomiędzy $n / 2$ i n (DES, 3DES),
- stała, równa $n / 2$ (LOKI97, FeW, LCB-IoT),
- stała, równa $n / 4$ (SM4, IDEA),
- stała, równa $n / 8$ (KASUMI),
- zmienna, równa n (PP-1, PP-2),
- zmienna, równa $n / 2$ (Speck),
- zmienna, równa $n / 4$ (RC6).

Cecha 4. Liczba iteracji algorytmu:

- równa $4 \cdot nrk$ (Serpent),

- równa $4 \cdot nrk - Nk$ (AES),
- równa $3 \cdot nrk$ (RC6),
- równa $nrk + \lceil d / t \rceil \cdot t$ (PP-2),
- równa $nrk + 1$ (PP-1),
- równa nrk (DES, 3DES, LOKI97, SM4, LCB-IoT),
- równa $nrk - 1$ (SAFER+, PRESENT, Speck),
- równa $nrk / 2$ (FeW),
- równa $nrk / 8$ (KASUMI),
- równa $(nrk - 4) / 8$ (IDEA).

Cecha 5. Długość przetwarzanego słowa algorytmu:

- równa maksymalnej wartości $|k|$ (LOKI97, Serpent),
- równa $|k|$ (SM4, KASUMI, FeW, SAFER+, PRESENT, IDEA),
- równa $2 \cdot |rk|$ (LCB-IoT, RC6, Speck),
- pomiędzy $|rk|$ i $|k|$ (DES, 3DES),
- równa $|rk|$ (AES, PP-1, PP-2).

Ponadto przetwarzanie może być realizowane w podsłowach:

- jednym (FeW, PRESENT, PP-1, PP-2, IDEA),
- dwóch (DES, 3DES, LCB-IoT, RC6, Speck),
- czterech (LOKI97, SM4, AES),
- ośmiu (KASUMI, Serpent),
- szesnastu (SAFER+).

Cecha 6. Uzupełnienie klucza głównego do długości:

- 256b (LOKI97 – $f(A, B)$, $f(B, A)$; Serpent – $1||0^{127}$, $1||0^{63}$),
- $\lceil d / t \rceil \cdot n$ (PP-2 – zerami),
- $c \cdot w$ (RC6 – zerami).

Cecha 7. Indywidualizacja iteracji – poprzez:

- stałe o zdefiniowanych wartościach (DES, 3DES, SM4, KASUMI),
- stałe określone formułą (LOKI97, AES, SAFER+, RC6),
- licznik iteracji o małej liczbie bitów (FeW – 8b, PRESENT – 5b),
- licznik iteracji o dużej liczbie bitów (Serpent – 32b, Speck – 16b, 24b, …),
- klucze pomocnicze określone formułą (PP-1, PP-2).

Cecha 8. Iteracje nadmiarowe lub rozbiegowe – stosowane w celu:

- uzależnienia pierwszego klucza rundowego, k_1, od całego klucza głównego, $k = k_H||k_L$ (PP-1 – iteracja #0),
- wpływu (dyfuzji) każdego bitu podklucza głównego, κ_i, na cały klucz rundowy, k_i (PP-2 – t rund rozbiegowych po każdym kluczu κ_i),
- uzależnienia każdego bitu klucza rundowego od wszystkich bitów klucza głównego (zależność 100%) i wpływu każdego bitu klucza głównego na wszystkie bity klucza rundowego (RC6 – trzy fazy obliczania kluczy rundowych, $R = 3 \cdot nrk$, z czego $2 \cdot nrk$ iteracji jest nadmiarowych).

Cecha 9. Bezpośrednie (bez przekształcenia) użycie klucza głównego k w kluczach rundowych k_i – stosowane przypadki:

- połowa klucza rundowego (FeW – 16b/80b, 16b/128b),
- cały klucz główny (LCB-IoT – 80b/80b, AES – 128b/128b, 192b/192b, 256b/256b; SAFER+ – 128b/128b, 192b/192b, 256b/256b; IDEA – 128b/128b),
- cały klucz rundowy (PRESENT – 64b/80b, 64/128b; Speck – n' / ($m \cdot n'$)).

Cecha 10. Skalowalność algorytmu – stosowane przypadki:

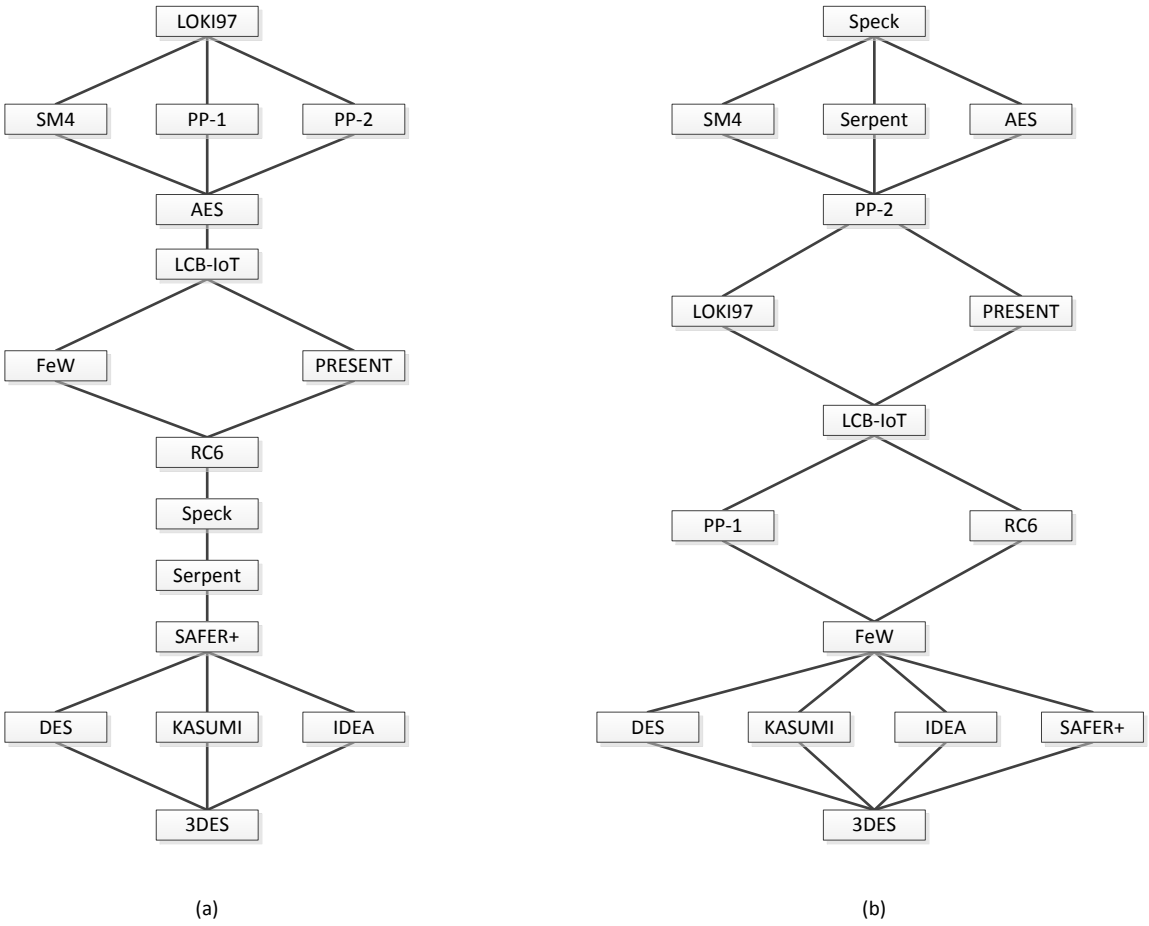
- skalowalny klucz rundowy ($|rk| = n = t \cdot 64$ dla $t = 1, 2, 3, \ldots$) i dwie długości klucza głównego, $|k| = n$ lub $|k| = 2n$ (PP-1),
- skalowalny klucz rundowy ($|rk| = n = t \cdot 64$ dla $t = 1, 2, 3, \ldots$) i wiele długości klucza głównego, $|k| = d \cdot 64$ dla $d \geq t$ (PP-2),
- skalowalny klucz rundowy ($|rk| = w = 8, 16, 32, 64, \ldots$) i niezależnie skalowalny klucz główny, $|k| = 8 \cdot b$ dla $b = 0, 1, \ldots, 255$ (RC6),
- skalowalny klucz rundowy ($|rk| = n' = 16, 24, 32, 48, 64$) i m długości klucza głównego, $|k| = m \cdot n'$ dla $m = 2, 3, 4$ (Speck).

Cecha 11, 12. Operacje liniowe i nieliniowe stosowane w algorytmach przedstawiono w odniesieniu do warstw funkcji szyfrującej *KA* – dodania klucza, *S* – podstawień i *P* – dyfuzji, omówionych w podrozdziale 3.1.

Warstwie *KA* odpowiada w algorytmie dodanie klucza pomocniczego K_i', w celu indywidualizacji funkcji iteracji. Stosowane sposoby indywidualizacji, punkt 7 tabel (tab. 3.49, tab. 3.50), można wyrazić w kategoriach kluczy pomocniczych. Klucz pomocniczy K_i':

- nie jest dodawany (LCB-IoT, IDEA),

- dodawany jest przez parametr rotacji (DES i 3DES, $K_i' = \beta_i$),
- dodawany operacją $\oplus$ (SM4 – $K_i' = CK_i$, KASUMI – $K_i' = Cj$, FeW – $K_i' = i$, AES – $K_i' = Rcon_i$, Serpent – $K_i' = \Phi \oplus i$, PRESENT – $K_i' = i$, Speck – $K_i' = i$),
- dodawany operacją $\boxplus$ (LOKI97 – $K_i' = C_i = \Delta \cdot i$, SAFER+ – $K_i' = B_i$, PP-2 – $K_i' = K_i$, RC6 – $K_i' = P_w \boxplus i \cdot Q_w$),
- dodawany operacjami $\oplus$, $\boxplus$, $\boxminus$ **(**PP-1 – $K_i' = K_i$**).**



Rys. 3.55. Uporządkowanie algorytmów generowania kluczy rundowych ze względu na warstwę: (a) podstawień, (b) dyfuzji

Warstwie *S* odpowiada w algorytmie wykorzystanie operacji nieliniowych, najczęściej w postaci S-bloków. Z uwagi na operacje nieliniowe można wyróżnić następujące przypadki (rys. 3.55 (a)):

- brak operacji nieliniowych (DES, 3DES, KASUMI, IDEA),
- operacja $\boxplus$ o znikomym znaczeniu (SAFER+),
- warstwa małych S-bloków (4b × 4b) o znikomym znaczeniu (Serpent),
- operacja $\boxplus$ (Speck),

- operacje $\lll_d b$ i $\boxplus$ (RC6),
- małe S-bloki (4b × 4b) dla podsłowa (FeW – (3 · 4b) / 80b, (4 · 4b) / 128b; PRESENT – (1 · 4b) / 80b, (2 · 4b) / 128b),
- warstwa małych S-bloków (4b × 4b) dla słowa (LCB-IoT – (4 · 4b) / 16b),
- duże S-bloki (8b × 8b) dla podsłowa (AES – (4 · 8b) / 128b),
- warstwa dużych S-bloków (8b × 8b) dla słowa (SM4 – (4 · 8b) / 32b, PP-1 – (8 · 8b) / 64b, PP-2 – (8 · 8b) / 64b),
- jedna lub dwie warstwy bardzo dużych S-bloków (11b × 8b, 13b × 8b) dla słowa (LOKI97 – jedna (32b/128b), dwie (96b/128b)).

Warstwie *P* odpowiada w algorytmie wykorzystanie elementów liniowych i nieliniowych w celu zapewnienia dyfuzji. Elementy nieliniowe, np. S-bloki, zapewniają tzw. dyfuzję lokalną. Każde wejście S-bloku posiada wpływ na każde jego wyjście. Zadaniem warstwy *P* jest zapewnienia tzw. dyfuzji globalnej, tzn. wpływu każdego bitu na wszystkie bity przetwarzanego słowa. Wszystkie bity wyjścia S-bloku w iteracji *i* powinny być rozproszone na wejścia, jak największej liczby, S-bloków w iteracji *i* + 1.

Dla ilustracji dyfuzji, dobrym przykładem struktury *S*-*P*-*S* (podstawienie-permutacja-podstawienie) jest funkcja *f*(*A*, *B*) szyfru LOKI97 (rys. 3.10). Każdy bit wejścia [4-0, 63-56] S-bloku *S*1 w warstwie podstawień S_a, ma wpływ na bity [63-56] wejścia permutacji *P*. Zgodnie z tablicą permutacji *P* (tab. 3.7), bity [63-56] wejścia *P* przekształcane są w bity [56, 48, 40, 32, 24, 16, 08, 00] wyjścia *P*, a więc w bity wejściowe wszystkich S-bloków warstwy podstawień S_b. W rezultacie każdy bit wejścia, górnego S-bloku *S*1 w warstwie S_a ma wpływ na (wszystkie) bity [63-0] wyjścia funkcji *f*(*A*, *B*). Podobnie jest dla pozostałych S-bloków warstwy S_a. Ostatecznie każdy bit wejścia warstwy S_a ma wpływ na wszystkie bity wyjścia warstwy S_b, co odpowiada S-blokowi *S*4′, o rozmiarze 64b × 64b, w schemacie zastępczym funkcji *f*(*A*, *B*), przedstawionym na rysunku (rys. 3.12).

Przy rozpatrywaniu dyfuzji należy uwzględnić następujące własności funkcji elementarnych:

- $\oplus$ – bit *i* wejścia (argumentu) funkcji XOR ma wpływ na bit *i* wyjścia (wartości),
- *P* – bit *i* wejścia permutacji bitowej *P*, jako jedyny, ma wpływ na bit $j = P(i)$ wyjścia,
- $\lll b$ – bit *i* wejścia rotacji w lewo o *b* bitów słowa *l*-bitowego, w przypadku LSB = 0 po prawej, ma wpływ na bit $(i + b)$ **mod** *l* wyjścia,
- *S* – bit *i* wejścia S-bloku *S*, ma wpływ na wszystkie bity wyjścia,

- $\boxplus$ – bit i wejścia funkcji dodawania modulo 2^l ma wpływ na bity $l-1$, $l-2$, …, i wyjścia, gdzie bit numer 0 jest najmniej znaczący,
- $\lll_d b$ – bit i wejścia rotacji w lewo, sterowanej d-bitowymi danymi, słowa l-bitowego, w przypadku LSB = 0 po prawej, ma wpływ na bity $(i + 2^d - 1)$ **mod** l, $(i + 2^d - 2)$ **mod** l, …, $(i + 1)$ **mod** l, i wyjścia.

Z uwagi na dyfuzję w algorytmie generowania kluczy rundowych wyróżnić można następujące przypadki (rys. 3.55 (b)):

- brak dyfuzji (DES, 3DES, KASUMI, IDEA, SAFER+),
- rotacja jako warstwa dyfuzji (FeW),
- rotacja sterowana danymi jako warstwa dyfuzji (PP-1, RC6),
- permutacja bitowa i rotacja jako warstwa dyfuzji (LCB-IoT),
- permutacja bitowa jako warstwa dyfuzji (LOKI97, PRESENT),
- skalowalna permutacja bitowa (wielokrotne rotacje) jako warstwa dyfuzji (PP-2),
- operacja XOR i rotacja jako warstwa dyfuzji (SM4, Serpent, AES),
- operacje ARX jako warstwa dyfuzji (Speck).

Niektóre algorytmy generowania kluczy rundowych inspirowane są funkcją szyfrującą (/deszyfrującą). Jest to widoczne na rysunkach ich struktury i w tabelach z listami operacji. Następujące algorytmy są podobne do funkcji szyfrującej (/deszyfrującej):

- LOKI97 (rys. 3.9, tab. 3.9),
- SM4 (rys. 3.13, tab. 3.12),
- PP-1 (rys. 3.36, rys. 3.37, tab. 3.36),
- PP-2 (rys. 3.40, rys. 3.41, tab. 3.41),
- Speck (rys. 3.52, rys. 3.53, tab. 3.48).

W szyfrach KASUMI, SAFER+ i IDEA, jakościowa różnica pomiędzy funkcją szyfrującą (/deszyfrującą) i algorytmem generowania kluczy rundowych jest największa.

Punkty 13 – 16 tabel (tab. 3.49, tab. 3.50) związane są z całościową, teoretyczną oceną jakości algorytmu.

Cecha 13. Zależność klucza rundowego k_i od bitów klucza głównego k – zależność ta jest:

- pełna <100%> (LOKI97, SM4, KASUMI, AES, Serpent, SAFER+, PP-1),
- duża <75%, 100%) (DES, PP-2, IDEA, RC6),
- średnia <50%, 75%) (PRESENT, Speck),

- mała <25%, 50%) (3DES, FeW),
- bardzo mała <0%, 25%) (LCB-IoT).

Przedstawiona, w tym punkcie tabel, syntetyczna miara zależności, stanowi średnią arytmetyczną procentowych wartości podanych w punkcie 4 charakterystyki algorytmu, dla pierwszych pięciu kluczy rundowych.

Cecha 14. Zależność bitu klucza rundowego k_i od bitów klucza głównego k – zależność ta jest:

- pełna <100%> (PP-2′),
- duża <75%, 100%) (LOKI97, SM4, PP-1, PP-2, RC6),
- średnia <50%, 75%) (AES),
- mała <25%, 50%) (Serpent, Speck),
- bardzo mała <0%, 25%) (DES, 3DES, KASUMI, FeW, LCB-IoT, SAFER+, PRESENT, IDEA).

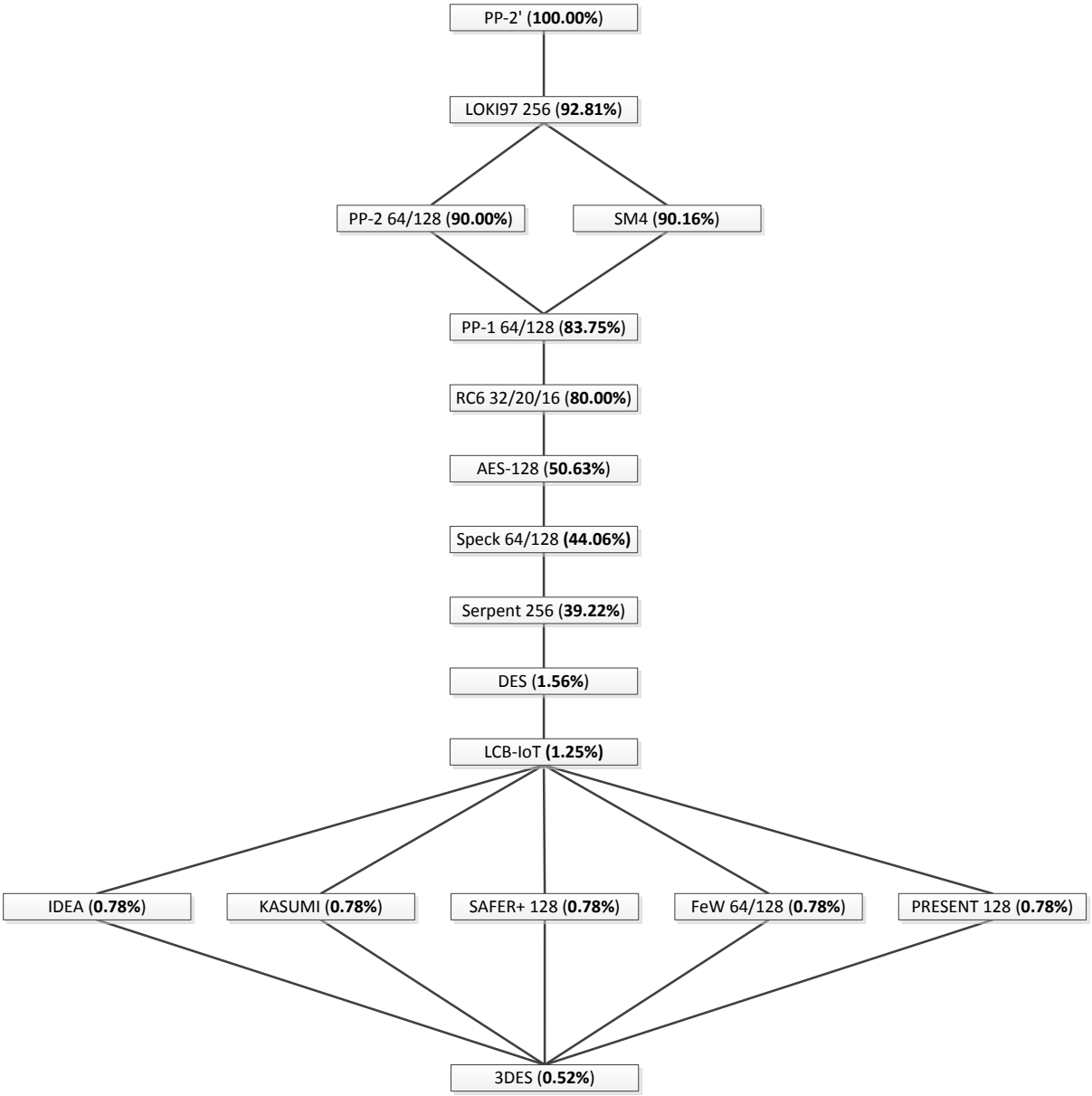
Podobnie jak w punkcie 13, przedstawione w punkcie 14 tabel wartości, stanowią średnią arytmetyczną wartości z punktu 5 charakterystyki algorytmu dla pierwszych pięciu kluczy rundowych.

Cecha 15. Zależność grupy (sąsiadujących) bitów klucza rundowego k_i od grupy (sąsiadujących) bitów klucza głównego k – zależność ta występuje w algorytmach:

- KASUMI (grupa 16b),
- SAFER+ (grupa 8b),
- IDEA (grupy 32b, 64b, 96b).

Powyższe wartości zostały pobrane z punktu 6 charakterystyki algorytmu.

Cecha 16. Teoretyczna ocena jakości algorytmu – oparta na zależności bitu klucza rundowego k_i od bitów klucza głównego k (punkt 14 tabel). Przedstawione wartości pobrano z punktu 11 charakterystyki algorytmu i według tego kryterium uporządkowano algorytmy generowania kluczy rundowych na rysunku (rys. 3.56).



Rys. 3.56. Uporządkowanie algorytmów generowania kluczy rundowych według teoretycznego kryterium jakości

Cecha 17. Długość próbki kluczy rundowych w bitach ($|rkp| = nrk \cdot |rk|$) – parametr z tabel zawierających zestawienie cech szyfru, wykorzystywany w metodzie A (punkt 4.3.2).

Cecha 18. Możliwość zwiększenia liczby iteracji algorytmu – parametr z wniosków wieńczących opis szyfrów, istotny w metodzie B (punkt 4.3.3).

4. Testowanie i ocena jakości algorytmów generowania kluczy rundowych

Rozdział 4 poświęcony jest metodom testowania i oceny jakości generatorów ciągów losowych oraz ciągów pseudolosowych. Przedstawiono w nim metodologię testów oraz pakiet testów statystycznych NIST SP800-22. W rozdziale zaprezentowano też wyniki badań autora rozprawy nad oceną jakości algorytmów generowania kluczy rundowych z wykorzystaniem testów NIST SP800-22.

4.1. Metodologia testów

Testy statystyczne i ich zastosowanie rozważane są zazwyczaj w odniesieniu do badania generatorów ciągów losowych bądź pseudolosowych i wytwarzanych przez nie sekwencji. Mogą być one również używane do oceny jakości szyfrów, a także algorytmów generowania kluczy rundowych. Jakość algorytmów generowania kluczy rundowych ocenia się po tym, jak wygenerowana przez algorytm sekwencja bitów kluczy jest różna od sekwencji losowej. Do oceny jakości algorytmów wykorzystać można specjalnie opracowane testy statystyczne, które z pewnym prawdopodobieństwem pozwalają stwierdzić czy badana sekwencja posiada pewne statystyczne własności charakterystyczne dla sekwencji losowej lub czy nie występują w niej pewne defekty statystyczne. Dodatkowo należy pamiętać, że wynikiem działania szyfru jest sekwencja binarna będąca szyfrogramem i zgodnie z zaleceniami Shannona taka sekwencja powinna mieć następujące własności [97][98]:

- dyfuzję (rozproszenie) – informacja z pojedynczego znaku wiadomości szyfrowanej powinna być rozproszona po całym szyfrogramie,
- konfuzję (nieregularność) – szyfrogram powinien być statystycznie niezależny od klucza.

Zatem w szyfrogramie nie powinny występować żadne regularności ani inne zależności między kolejnymi bitami czy nawet między kolejnymi szyfrogramami (tworzonymi przy użyciu tego samego klucza). Występowanie takich defektów ułatwia kryptoanalizę. Dlatego w procesie projektowania szyfrów oraz algorytmów generowania kluczy rundowych jednym z koniecznych etapów oceny jest ocena statystyczna. W konkursie na standard AES stosowano takie podejście i poddano ocenie statystycznej każdy z 15 szyfrów, zakwalifikowanych do konkursu, z wykorzystaniem pakietu NIST SP800-22 [103][104].

Należy zauważyć, że nie ma jednoznacznej metody, aby określić czy badany algorytm generowania kluczy lub wygenerowana przez algorytm sekwencja są dobre czy złe. Do momentu gdy nie znajdzie się test, w którym testowana sekwencja wygenerowana przez badany algorytm nie zakończy się sukcesem, to taką sekwencję uznaje się za losową.

Testowanie algorytmów generowania kluczy rundowych (dotyczy to również generatorów ciągów pseudolosowych) to testowanie pewnych hipotez statystycznych o wygenerowanej przez nie sekwencji. Stwierdzenie, że badana sekwencja jest losowa, jest badaną hipotezą zerową, oznaczaną jako H_0 (ang. *null hypothesis*). Test mówi jak duże jest prawdopodobieństwo tego, że ta hipoteza jest prawdziwa. Hipoteza alternatywna H_a (ang. *alternative hypothesis*) dotyczy sekwencji nielosowej [95].

Ważnym parametrem przy testowaniu hipotez statystycznych jest poziom istotności (PI), oznaczany jako α, który określa prawdopodobieństwo odrzucenia hipotezy H_0, jeśli jest ona prawdziwa. Przykładowo jeśli dla $\alpha = 0.01$ badany ciąg nie spełnia testu, to z jednoprocentowym prawdopodobieństwem ten ciąg jest jednak losowy. W praktyce parametr α przyjmuje wartość z przedziału <0.001; 0.01>, a najczęściej wartość 0.01.

Tab. 4.1. Przypadki występowania błędów I i II rodzaju [95]

sytuacja (w rzeczywistości)	**wniosek (wynik testu)**	
	akceptowana H_0	**akceptowana H_a (odrzucona H_0)**
sekwencja jest losowa (H_0 jest prawdziwa)	brak błędu	błąd I rodzaju
sekwencja nie jest losowa (H_a jest prawdziwa)	błąd II rodzaju	brak błędu

Podczas testowania hipotez występują dwa rodzaje błędów (tab. 4.1):

- I rodzaju – hipotezę H_0 uznaje się za fałszywą, gdy jest prawdziwa (na podstawie wyniku testu odrzucany jest ciąg losowy),
- II rodzaju – hipotezę H_0 uznaje się za prawdziwą, gdy jest fałszywa (na podstawie wyniku testu akceptowany jest ciąg, który ma pewne defekty statystyczne).

Prawdopodobieństwo błędu II rodzaju oznaczane jest przez β, natomiast prawdopodobieństwo I rodzaju to α czyli poziom istotności. W ujęciu kryptograficznym błąd II rodzaju jest równie ważny jak błąd I rodzaju ponieważ dopuszczenie sekwencji nielosowej z defektami jako klucza szyfrującego może skutkować gorszymi konsekwencjami aniżeli odrzucenie sekwencji o dobrych własnościach.

4.2. Pakiet testów statystycznych NIST SP800-22

Pakiet NIST SP800-22 [95] został opracowany przez Narodowy Instytut Standardów i Technologii USA i jest rekomendowanym narzędziem do oceny generatorów ciągów losowych i pseudolosowych stosowanych w kryptografii. Niektóre testy możliwe są do przeprowadzenia wyłącznie na długich próbkach danych tj. próbkach o długości $n \geq 10^6$ bitów. Niewątpliwą zaletą pakietu jest obszerna dokumentacja techniczna wraz ze wskazówkami na temat sposobu interpretacji otrzymanych wyników, natomiast do wad pakietu należy zaliczyć długi czas przetwarzania testowanych sekwencji.

W testach NIST SP800-22 wykorzystywane są następujące funkcje:

- $\Phi()$ – funkcja rozkładu skumulowanego (ang. *Cumulative Distribution Function*):

$$\Phi(z) = \frac{1}{2\pi}\int_{-\infty}^{z} e^{-u^2/2}\, du. \tag{4.1}$$

- ***erfc***() – komplementarna funkcja błędu (ang. *Complementary Error Function*):

$$erfc(z) = \frac{2}{\sqrt{\pi}}\int_{z}^{\infty} e^{-u^2}\, du. \tag{4.2}$$

- $\Gamma()$ – funkcja gamma (ang. *Gamma Function*):

$$\Gamma(z) = \int_{0}^{\infty} t^{z-1}\, e^{-t}\, dt. \tag{4.3}$$

- ***igamc***() – niezupełna funkcja gamma (ang. *Incomplete Gamma Function*):

$$Q(a,x) = \frac{1}{\Gamma(a)}\int_{x}^{\infty} t^{a-1}\, e^{-t}\, dt. \tag{4.4}$$

Wykorzystywane są również w testach NIST SP800-22 następujące oznaczenia:

- α – poziom istotności (ang. *significance level*),
- $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ – próbka (ciąg zero-jedynkowy),
- n – długość próbki w bitach,
- m – liczba próbek,
- M – rozmiar bloku w bitach,
- *p-value – p-wartość,*

W pakiecie NIST SP800-22 zaimplementowano 15 testów statystycznych[6] [95]. Pliki z danymi wejściowymi zawierają m próbek (ciągów zero-jedynkowych) o długość n.

Każdy z testów oblicza *p-wartość* (ang. *p-value*), będącą liczbą z przedziału <0, 1>, która określa prawdopodobieństwo, że doskonały generator liczb losowych wytworzy sekwencję bitów mniej losową od sekwencji testowanej. *P-wartość* = 1 oznacza doskonałą

[6] W nawiasach klamrowym umieszczono skrócone nazwy testów wykorzystywane w dalszej części pracy.

losowość, *p-wartość* = 0 oznacza zupełną nielosowość, *p-wartość* ≥ α oznacza losowość, a *p-wartość* < α oznacza nielosowość. Jeśli α = 0.01 to 1 / 100 losowych sekwencji jest odrzuconych przez test. *P-wartość* ≥ 0.01 oznacza losowość sekwencji z ufnością 99%, a *p-wartość* < 0.01 oznacza nielosowość z ufnością 99%.

Test 1. Test częstości (ang. *Frequency Test*) **{frequency}** – sprawdza czy liczba jedynek (zer) w próbce, odpowiada sekwencji losowej.

FUNKCJA: *Frequency*(*n*)

WEJŚCIE: $n \geq 100$

WYJŚCIE: (*p-value* ≥ α)

PROCEDURA:

1. Dla $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ oblicz sumę: $S_n = X_1 + X_2 + \ldots + X_n$, gdzie $X_i = 2\varepsilon_i - 1 = \pm 1$.
2. Oblicz statystykę testu: $s(obs) = |S_n|/\sqrt{n}$.
3. Oblicz *p-wartość*: *p-value* = $erfc(s(obs)/\sqrt{2})$.
4. Oblicz wynik testu: ACCEPT/REJECT = (*p-value* ≥ α).

UWAGI:

- rekomendowana długość próbki $n \geq 100$;
- pozostałe testy uzależnione są od pozytywnego wyniku tego testu;
- rozkład odniesienia – półnormalny (ang. *half normal*);
- dla liczby jedynek równej liczbie zer w próbce zachodzi *s*(*obs*) = 0;
- jako nielosowa uznawana jest próbka o zbyt dużej lub zbyt małej liczbie jedynek.

Test 2. Test częstości w blokach (ang. *Frequency Test within a Block*) {**block-frequency**} – sprawdza czy liczby jedynek (zer) w *M*-bitowych blokach próbki, odpowiadają sekwencji losowej.

FUNKCJA: *BlockFrequency*(*M*, *n*)

WEJŚCIE: $n \geq 100$, $M \geq 20$, $M > 0.01n$

WYJŚCIE: (*p-value* ≥ α)

PROCEDURA:

1. Podziel $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ na *M*-bitowe bloki $B_1, B_2, \ldots, B_N$, gdzie $N = \lfloor n / M \rfloor$.
2. Oblicz proporcję π_i jedynek w bloku B_i ($i = 1, 2, \ldots, N$):
$$\pi_i = (\sum_{j=1}^{M} \varepsilon_{(i-1)M+j})/M.$$
3. Oblicz statystykę testu:
$$\chi^2(obs) = 4M \sum_{i=1}^{N} (\pi_i - 1/2)^2.$$

4. Oblicz *p-wartość*: *p-value* = $igamc(N/2, \chi^2(obs)/2)$.
5. Oblicz wynik testu: ACCEPT/REJECT = (*p-value* ≥ α).

UWAGI:

- rekomendacja $M > 0.01n$ implikuje liczbę bloków $N < 100$;
- jeśli $MN < n$ to bity ε_i próbki dla $MN < i \leq n$, są w teście ignorowane;
- dla $M = 1$ ten test redukuje się do testu częstości (test 1);
- rozkład odniesienia – χ^2 (ang. *chi-square*);
- jako nielosowa uznawana jest próbka o zbyt dużej lub zbyt małej liczbie jedynek w co najmniej jednym bloku.

Test 3. Test serii (ang. *Runs Test)* {**runs**} – sprawdza czy liczba ciągów samych zer (serii zer) i samych jedynek (serii jedynek) w próbce, odpowiada sekwencji losowej.

FUNKCJA: *Runs*(*n*)

WEJŚCIE: $n \geq 100$

WYJŚCIE: (*p-value* ≥ α)

PROCEDURA:

1. Dla $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ oblicz proporcję jedynek:
$$\pi = (\sum_{i=1}^{n} \varepsilon_i)/n.$$
2. Jeśli $|\pi - 1/2| \geq \tau = 2/\sqrt{n}$ to podstaw *p-value* = 0 i przejdź do kroku 5.
3. Oblicz statystykę testu:
$$V_n(obs) = (\sum_{i=1}^{n-1} r(i)) + 1,$$
gdzie $r(i) = 0$ dla $\varepsilon_i = \varepsilon_{i+1}$ oraz $r(i) = 1$, w przeciwnym przypadku.
4. Oblicz *p-wartość*: *p-value* = $erfc(\frac{|V_n(obs) - 2n\pi(1-\pi)|}{2\sqrt{2n}\,\pi(1-\pi)})$.
5. Oblicz wynik testu: ACCEPT/REJECT = (*p-value* ≥ α).

UWAGI:

- test sprawdza czy oscylacje pomiędzy seriami zer i jedynek nie są zbyt szybkie lub zbyt wolne;
- oczekiwaną liczbą serii jest $n/2$;
- statystyka testu $V_n(obs)$ równa jest liczbie serii (liczbie oscylacji +1);
- rozkład odniesienia – normalny (ang. *normal*).

Test 4. Test na najdłuższą serię jedynek w bloku (ang. *Test for the Longest Run of Ones in a Block*) {**longest-runs**} – sprawdza czy najdłuższe serie jedynek w *M*-bitowych blokach próbki, odpowiadają sekwencji losowej.

FUNKCJA: *LongestRunOfOnes*(*n*)

WEJŚCIE: $n \geq 128$

WYJŚCIE: (*p-value* $\geq \alpha$)

PROCEDURA:

1. Podziel $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ na *M*-bitowe bloki $B_1, B_2, \ldots, B_N$, gdzie $N = \lfloor n / M \rfloor$ i $M = 8$ ($128 \leq n < 6272$), 128 ($6272 \leq n < 750000$), 10^4 ($n \geq 75 \cdot 10^4$).
2. Podziel przedział $<0, M>$ na podprzedziały $C_0, C_1, \ldots, C_K$.
3. Dla $j = 1, 2, \ldots, N$ oblicz: $MR_j = MaxRun(B_j)$.
4. Dla $i = 0, 1, \ldots, K$ oblicz: $v_i = \#(MR_j \in C_i)$.
5. Oblicz statystykę testu:

$$\chi^2(obs) = \sum_{i=0}^{K} \frac{(v_i - N\pi_i)^2}{N\pi_i},$$

   gdzie π_i to prawdopodobieństwo teoretyczne.
6. Oblicz *p-wartość*: $p\text{-}value = igamc(K/2, \chi^2(obs)/2)$.
7. Oblicz wynik testu: ACCEPT/REJECT = (*p-value* $\geq \alpha$).

UWAGI:

- liczba podprzedziałów (klas) $K + 1$, wynosi odpowiednio 4, 6, 7, dla $M = 8, 128, 10^4$;
- prawdopodobieństwa teoretyczne $\pi_0, \pi_1, \ldots, \pi_K$, zależne od *M*, przedstawiono w dokumentacji pakietu w formie tabeli;
- odstępstwo najdłuższych serii jedynek, od wartości oczekiwanych, implikuje takie odstępstwo najdłuższych serii zer;
- duża wartość $\chi^2(obs)$ implikuje małą wartość *p-value*.

Test 5. Test rzędów macierzy binarnych (ang. *Binary Matrix Rank Test*) {**rank**} – sprawdza czy rzędy, utworzonych z próbki, macierzy binarnych o *M* wierszach i *Q* kolumnach, odpowiadają sekwencji losowej.

FUNKCJA: *Rank*(*n*)

WEJŚCIE: $n \geq 38912$

WYJŚCIE: (*p-value* $\geq \alpha$)

PROCEDURA:

1. Wpisz $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$, wierszami, do binarnych macierzy $A_1, A_2, \ldots, A_N$, o $M = 32$ wierszach i $Q = 32$ kolumnach, gdzie $N = \lfloor n / (MQ) \rfloor$.
2. Dla $i = 1, 2, \ldots, N$ oblicz rząd: $R_i = rank(A_i)$.
3. Oblicz częstości: $F_M = \#(R_i = M)$, $F_{M-1} = \#(R_i = M-1)$.
4. Oblicz statystykę testu:

   $$\chi^2(obs) = \frac{(F_M - P_M N)^2}{P_M N} + \frac{(F_{M-1} - P_{M-1} N)^2}{P_{M-1} N} + \frac{((N - F_M - F_{M-1}) - (1 - P_M - P_{M-1})N)^2}{(1 - P_M - P_{M-1})N},$$

   gdzie $P_M = 0.2888$ i $P_{M-1} = 0.5776$ to prawdopodobieństwa teoretyczne.
5. Oblicz *p-wartość*: $p\text{-}value = igamc(1, \chi^2(obs)/2)$.
6. Oblicz wynik testu: ACCEPT/REJECT = ($p\text{-}value \geq \alpha$).

UWAGI:

- ten test występuje również w pakiecie testów DIEHARD [82];
- jeśli $MQN < n$ to bity $\varepsilon_{i'}$, dla $MQN < i' \leq n$, są w teście ignorowane;
- funkcja obliczania rzędu macierzy, $rank(A_i)$, przedstawiona jest w dodatku A dokumentacji pakietu;
- test sprawdza liniową zależność pomiędzy wierszami macierzy A_i;
- wiersze losowej macierzy są niezależne z prawdopodobieństwem $P_M = 0.2888$, jeden jest zależny z prawdopodobieństwem $P_{M-1} = 0.5776$, a dwa lub więcej – z prawdopodobieństwem $1 - P_M - P_{M-1} = 0.1336$.

Test 6. Test (widmowy) dyskretnej transformaty Fouriera (ang. *Discrete Fourier Transform (Spectral) Test*) {**fft**} – sprawdza czy wysokość pików, dyskretnej transformaty Fouriera próbki, odpowiada sekwencji losowej.

FUNKCJA: *DiscreteFourierTransform*(n)

WEJŚCIE: $n \geq 1000$

WYJŚCIE: ($p\text{-}value \geq \alpha$)

PROCEDURA:

1. Dla $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ utwórz ciąg $X = X_1, X_2, \ldots, X_n$, gdzie $X_k = 2\varepsilon_k - 1$.
2. Zastosuj transformatę DFT do X ($S = DFT(X)$):

   $s_j = \sum_{k=1}^{n} x_k exp(2\pi \boldsymbol{i}(k-1)j/n)$, dla $j = 0, 1, \ldots, n-1$ oraz $\boldsymbol{i} = \sqrt{-1}$,

   gdzie $exp(2\pi \boldsymbol{i}(k-1)j/n) = \cos(2\pi(k-1)j/n) + \boldsymbol{i} \sin(2\pi(k-1)j/n)$.
3. Oblicz sekwencję wysokości pików: $M = modulus(S')$, gdzie $S' = S[0, n/2 - 1]$.

4. Oblicz 95% próg wysokości pików: $T = \sqrt{nlog(1/0.05)}$.
5. Oblicz liczbę pików mniejszych od *T*: oczekiwaną $N_0 = 0.95n/2$ i obserwowaną N_1.
6. Oblicz statystykę testu: $d = (N_1 - N_0)/\sqrt{n(0.95)(0.05)/4}$.
7. Oblicz *p-wartość*: *p-value* = $erfc(|d|/\sqrt{2})$.
8. Oblicz wynik testu: ACCEPT/REJECT = (*p-value* ≥ α).

UWAGI:

- celem testu jest wykrycie pojawiających się okresowo w próbce, sekwencji bitów (wzorców);
- intencją testu jest sprawdzenie czy liczba pików wyższych od progu 95% znacząco różni się od 5%;
- ciąg *S′* wyznaczany jest z uwagi na symetrię ciągu *S;*
- rozkład odniesienia – normalny.

Test 7. Test dopasowania nienachodzących wzorców (ang. *Non-overlapping Template Matching Test*) {**non-overlapping-templates**} – sprawdza czy liczby *m*-bitowych nienachodzących wzorców w *M*-bitowych blokach próbki, odpowiadają sekwencji losowej.

FUNKCJA: *NonOverlappingTemplateMatching*(*m*, *n*)

WEJŚCIE: $n \geq 16$, $2 \leq m \leq 10$ (rekomendowana długość wzorca $m = 9$ lub 10)

WYJŚCIE: (*p-value* ≥ α) – wiele wyników

PROCEDURA:

1. Podziel $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ na *M*-bitowe bloki $B_1, B_2, \ldots, B_N$, gdzie $N = 8$, $M = \lfloor n / N \rfloor$.
2. Oblicz liczbę W_i nienachodzących wystąpień wzorca *B* w bloku B_i ($i = 1, 2, \ldots, N$).
3. Oblicz teoretyczne wartości średniej i wariancji:

$$\mu = \frac{M-m+1}{2^m}, \qquad \sigma^2 = M(\frac{1}{2^m} - (2m-1)/2^{2m}).$$

4. Oblicz statystykę testu:

$$\chi^2(obs) = \sum_{i=1}^{N}((W_i - \mu)^2/\sigma^2)$$

5. Oblicz *p-wartość*: *p-value* = $igamc(N/2, \chi^2(obs)/2)$.
6. Oblicz wynik testu: ACCEPT/REJECT = (*p-value* ≥ α) – wiele wyników

UWAGI:

- celem testu jest wykrycie w próbce, powtarzających się zbyt wiele lub zbyt mało razy, nieokresowych wzorców, zawartych w bibliotece wzorców;

- w celu wykrycia *m*-bitowego wzorca, przesuwane jest wzdłuż próbki *m*-bitowe okno, które w przypadku wykrycia wzorca przesuwane jest nie na kolejną pozycję (nachodzące wzorce), a poza koniec wzorca (nienachodzące wzorce);
- w odróżnieniu od testu 6 (DFT), ten test wykrywa również wzorce powtarzające się w próbce nieokresowo;
- dla n =9 liczba *p-wartości* (wzorców) może być równa 148, a dla m = 10 może być równa 284.

Test 8. Test dopasowania nachodzących wzorców (ang. *Overlapping Template Matching Test*) {**overlapping-templates**} – sprawdza czy liczby *m*-bitowych nachodzących wzorców w *M*-bitowych blokach próbki, odpowiadają sekwencji losowej.

FUNKCJA: *OverlappingTemplateMatching*(*m*, *n*)

WEJŚCIE: $n \geq 10^6$, rekomendowana długość wzorca m = 9 lub 10

WYJŚCIE: (*p-value* $\geq \alpha$)

PROCEDURA:

1. Podziel $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ na M-bitowe bloki $B_1, B_2, \ldots, B_N$, gdzie M = 1032, N = 968.
2. Podziel przedział <0, M> na podprzedziały $C_0, C_1, \ldots, C_K$, gdzie K = 5, C_0 = <0, 0>, C_1 = <1, 1>, …, C_{K-1} = <K – 1, K – 1>, C_K = <K, M>.
3. Oblicz liczbę w_j nachodzących wystąpień wzorca B w bloku B_j (j = 1, 2, …, N).
4. Dla i = 0, 1, …, K oblicz: $v_i = \#(w_j \in C_i)$.
5. Oblicz statystykę testu: $\chi^2(obs) = \sum_{i=0}^{K}(v_i - N\pi_i)^2/N\pi_i$, gdzie π_i to prawdopodobieństwo teoretyczne.
6. Oblicz *p-wartość*: *p-value* = $igamc(K/2, \chi^2(obs)/2)$.
7. Oblicz wynik testu: ACCEPT/REJECT = (*p-value* $\geq \alpha$).

UWAGI:

- celem testu jest wykrycie w blokach próbki *m*-bitowych serii jedynek powtarzających się zbyt mało lub zbyt wiele razy;
- okno *m*-bitowe przesuwane wzdłuż próbki, po wykryciu wzorca przesuwane jest na kolejną pozycję (nachodzące wzorce);
- zmienna K = 5 jest liczbą stopni swobody (ang. *number of degrees of freedom*);
- prawdopodobieństwa π_i (i = 0, 1, …, K) obliczane są w oparciu o rozkład Poissona.

Test 9. „Uniwersalny statystyczny" test Maurera (ang. *Maurer's „Universal Statistical" Test*) {**universal**} – sprawdza czy potencjalna możliwość kompresji próbki, odpowiada sekwencji losowej.

FUNKCJA: *Universal*(*M*, *Q*, *n*)

WEJŚCIE: $n \geq 387840$, długość bloku M, $6 \leq M \leq 16$, liczba bloków segmentu inicjalizującego $Q = 10 \cdot 2^M$

WYJŚCIE: (*p-value* $\geq \alpha$)

PROCEDURA:

1. Podziel $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ na M-bitowe bloki $B_1, B_2, \ldots, B_N$, gdzie $N = \lfloor n / M \rfloor$. Pierwszych Q bloków tworzy segment inicjalizujący, kolejnych $K = N - Q$ bloków – segment testowy.
2. Utwórz ciąg (tablicę) $T_0, T_1, \ldots, T_{2^M-1}$ i podstaw $T_j = 0$ dla $j = 0, 1, \ldots, 2^M - 1$.
3. Dla $i = 1$ do Q wykonaj: $j = B_i$, $T_j = i$.
4. Podstaw *sum* = 0 oraz dla $i = Q + 1$ do N wykonaj: $j = B_i$, *sum* = *sum* + $\log_2(i - T_j)$, $T_j = i$.
5. Oblicz statystykę testu: $f_n = \frac{1}{(N-Q)} \sum_{i=Q+1}^{N} \log_2(i - T_j) = \frac{sum}{(N-Q)}$.
6. Oblicz *p-wartość*: *p-value* = $erfc(|f_n - expectedValue(M)/\sqrt{2}\,\sigma)$, gdzie *expectedValue*(*M*) i σ, również zależne od M, są stabelaryzowane.
7. Oblicz wynik testu: ACCEPT/REJECT = (*p-value* $\geq \alpha$).

UWAGI:

- test bada liczbę bitów pomiędzy wystąpieniami M-bitowego wzorca w próbce, tj. bada miarę wykorzystywaną w kompresji ciągów binarnych;
- analizowane są wszystkie wzorce M-bitowe, tzn. liczby 0, 1, …, 2^M–1;
- próbka o dużej możliwości kompresji uznawana jest za nielosową;
- rozkład odniesienia – półnormalny, podobnie jak w teście częstości (test 1).

Test 10. Test złożoności liniowej (ang. *Linear Complexity Test*) {**linear-complexity**} – sprawdza czy długości najkrótszych LFSR (ang. *Linear Feedback Shift Register*), generujących M-bitowe bloki próbki, odpowiadają sekwencji losowej.

FUNKCJA: *LinearComplexity*(*M*, *n*)

WEJŚCIE: $n \geq 10^6$, długość bloku M, $500 \leq M \leq 5000$

WYJŚCIE: (*p-value* $\geq \alpha$)

PROCEDURA:

1. Podziel $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ na M-bitowe bloki $B_1, B_2, \ldots, B_N$, gdzie $N = \lfloor n / M \rfloor$.
2. Dla $j = 1, 2, \ldots, N$ oblicz złożoność liniową L_j, tj. długość najkrótszego LFSR generującego blok B_j.
3. Oblicz teoretyczną średnią: $\mu = M / 2 + (9 + (-1)^{M+1}) / 36 - (M / 3 + 2 / 9) / 2^M$.
4. Dla $j = 1, 2, \ldots, N$ oblicz: $T_j = (-1)^M \cdot (L_j - \mu) + 2 / 9$.
5. Podziel przedział $(-\infty, \infty)$, na podprzedziały $C_0, C_1, \ldots, C_K$, gdzie $K = 6$ i $C_0 = (-\infty, -2.5>$, $C_1 = (-2.5, -1.5>$, $C_2 = (-1.5, -0.5>$, $C_3 = (-0.5, 0.5>$, $C_4 = (0.5, 1.5>$, $C_5 = (1.5, 2.5>$, $C_6 = (2.5, \infty)$.
6. Dla $i = 0, 1, \ldots, K$ oblicz: $v_i = \#(T_j \in C_i)$.
7. Oblicz statystykę testu: $\chi^2(obs) = \sum_{i=0}^{K} (v_i - N\pi_i)^2 / N\pi_i$, gdzie π_i to prawdopodobieństwo teoretyczne.
8. Oblicz *p-wartość*: *p-value* $= igamc(K/2, \chi^2(obs)/2)$.
9. Oblicz wynik testu: ACCEPT/REJECT = (*p-value* $\geq \alpha$).

UWAGI:

- ciągi losowe wymagają długich LFSR, a nielosowe – krótkich;
- zmienna $K = 6$ jest liczbą stopni swobody, podobnie jak w teście 8;
- złożoność liniowa L_j obliczana jest algorytmem Berlekampa-Masseya [85];
- prawdopodobieństwa π_i ($i = 0, 1, \ldots, K$) obliczane są zgodnie z prostymi wzorami, przedstawionymi w dokumentacji pakietu.

Test 11. Test seryjny (ang. *Serial Test*) {**serial**} – sprawdza czy częstość wszystkich możliwych m-bitowych wartości, w (nachodzących na siebie) blokach próbki na każdej z n pozycji, odpowiada sekwencji losowej.

FUNKCJA: *Serial*(m, n)

WEJŚCIE: długość próbki n i długość bloku m takie, że: $m < \lfloor \log_2 n \rfloor - 2$

WYJŚCIE: (*p-value* $\geq \alpha$) – dwa wyniki

PROCEDURA:

1. Dla $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ utwórz powiększony ciąg: $\varepsilon' = \varepsilon \,||\, \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{m-1}$.
2. Niech B_j^l oznacza l-bitowy blok na pozycji j ciągu ε', gdzie $l = m, m-1, m-2$, a $j = 1, 2, \ldots, n$. Dla wszystkich możliwych binarnych wartości: $i_1 \ldots i_m$, $i_1 \ldots i_{m-1}$, $i_1 \ldots i_{m-2}$, oblicz częstości:

$$v_{i_1 \ldots i_m} = \#(B_j^m = i_1 \ldots i_m),$$

$$v_{i_1 \dots i_{m-1}} = \#(B_j^{m-1} = i_1 \dots i_{m-1}),$$

$$v_{i_1 \dots i_{m-2}} = \#(B_j^{m-2} = i_1 \dots i_{m-2}),$$ gdzie $j = 1, 2, \dots, n$.

3. Oblicz:

$$\Psi_m^2 = \frac{2^m}{n}\sum_{i_i \dots i_m}\left(v_{i_1 \dots i_m} - \frac{n}{2^m}\right)^2 = \frac{2^m}{n}\sum_{i_i \dots i_m} v_{i_1 \dots i_m}^2 - n$$

$$\Psi_{m-1}^2 = \frac{2^{m-1}}{n}\sum_{i_i \dots i_{m-1}}(v_{i_1 \dots i_{m-1}} - \frac{n}{2^{m-1}})^2 = \frac{2^{m-1}}{n}\sum_{i_i \dots i_{m-1}} v_{i_1 \dots i_{m-1}}^2 - n,$$

$$\Psi_{m-2}^2 = \frac{2^{m-2}}{n}\sum_{i_i \dots i_{m-2}}(v_{i_1 \dots i_{m-2}} - \frac{n}{2^{m-2}})^2 = \frac{2^{m-2}}{n}\sum_{i_i \dots i_{m-2}} v_{i_1 \dots i_{m-2}}^2 - n.$$

4. Oblicz statystykę testu:

$$\nabla\Psi_m^2(obs) = \Psi_m^2 - \Psi_{m-1}^2,$$

$$\nabla^2\Psi_m^2(obs) = \Psi_m^2 - 2\Psi_{m-1}^2 + \Psi_{m-2}^2.$$

5. Oblicz *p-wartość*:

$$p\text{-}value1 = igamc(2^{m-2}, \nabla\Psi_m^2(obs)),$$

$$p\text{-}value2 = igamc\left(2^{m-3}, \nabla^2\Psi_m^2(obs)\right).$$

6. Oblicz wynik testu: ACCEPT/REJECT = (*p-value* ≥ α) – dwa wyniki.

UWAGI:

- każdy *m*-bitowy wzorzec jest tak samo prawdopodobny (równomierność);
- duża wartość $\nabla\Psi_m^2(obs)$ lub $\nabla^2\Psi_m^2(obs)$ świadczy o nierównomierności i cechuje próbkę nielosową;
- wartość Ψ_m^2 nie jest używana jako statystyka testu ponieważ częstości $v_{i_1 \dots i_m}$ nie są niezależne;
- właściwym rozwiązaniem są statystyki $\nabla\Psi_m^2(obs), \nabla^2\Psi_m^2(obs)$ o rozkładach χ^2 z odpowiednio 2^{m-1} i 2^{m-2} stopniami swobody;
- dla $m = 1$ ten test odpowiada testowi częstości (test 1).

Test 12. Test przybliżonej entropii (ang. *Approximate Entropy Test*) {**apen**} – sprawdza czy porównanie entropii dla nachodzących na siebie *m*-bitowych bloków oraz entropii dla (*m*+1)-bitowych bloków próbki, odpowiada sekwencji losowej.

FUNKCJA: *ApproximateEntropy*(*m*, *n*)

WEJŚCIE: długość próbki *n* i długość bloku *m* takie, że: $m < \lfloor \log_2 n \rfloor - 5$

WYJŚCIE: (*p-value* ≥ α)

PROCEDURA:

1. Dla $\varepsilon = \varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$ utwórz powiększony ciąg: $\varepsilon' = \varepsilon \,||\, \varepsilon_1, \varepsilon_2, \dots, \varepsilon_{m-1}$.
2. Niech B_j^m oznacza *m*-bitowy blok na pozycji *j* ciągu ε', gdzie $j = 1, 2, \dots, n$.

3. Dla wszystkich możliwych *m*-bitowych wartości *i* (*i* = 0, 1, …, 2^m–1) oblicz:

$$C_i^m = \#(B_j^m = i)/n.$$

4. Oblicz:

$$\phi^{(m)} = \sum_{i=0}^{2^m-1} \pi_i \log \pi_i \text{ , gdzie } \pi_i = C_i^m.$$

5. Powtórz kroki 1-4 dla *m* + 1.
6. Oblicz statystykę testu: $\chi^2(obs) = 2n(\log 2 - ApEn(m))$,
   gdzie $ApEn(m) = \phi^{(m)} - \phi^{(m+1)}$.
7. Oblicz *p-wartość*: *p-value* = $igamc(2^{m-1}, \chi^2(obs)/2)$.
8. Oblicz wynik testu: ACCEPT/REJECT = (*p-value* ≥ α).

UWAGI:

- podobnie jak w teście seryjnym (test 11), wyznaczana jest częstość wszystkich możliwych *m*-bitowych wartości w blokach próbki, na każdej z *n* pozycji;
- mała wartość *ApEn*(*m*) świadczy o dużej regularności, a duża wartość – o nieregularności (losowości) próbki;
- dla ustalonej długości *m* bloku, w długiej losowej sekwencji binarnej, wartość *ApEn*(*m*) ≈ log2.

Test 13. Test kumulowanych sum (ang. *Cumulative Sums (Cusum) Test*) {**cumulative-sums**} – sprawdza czy maksymalna, bezwzględna wartość kumulowanej sumy dla próbki, odpowiada sekwencji losowej.

FUNKCJA: *CumulativeSums*(*mode, n*)

WEJŚCIE: *n* ≥ 100, *mode* = forward/backward (0/1)

WYJŚCIE: (*p-value* ≥ α)

PROCEDURA:

1. Dla $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ utwórz ciąg: $X = X_1, X_2, \ldots, X_n$, gdzie $X_i = 2\varepsilon_i - 1$.
2. Dla $k = 1, 2, \ldots, n$ oblicz kumulowane sumy:

$$S_k = X_1 + X_2 + \ldots + X_k, \text{ gdy } mode = 0,$$

$$S_k = X_n + X_{n-1} + \ldots + X_{n-k+1}, \text{ gdy } mode = 1.$$

3. Oblicz statystykę testu:

$$z(obs) = \max_{1 \le k \le n} |S_k|$$

4. Oblicz *p-wartość*: *p-value* =

$$1 - \sum_{k=(-n/z(obs)+1)/4}^{(n/z(obs)-1)/4}\left[\phi\left((4k+1)z(obs)/\sqrt{n}\right) - \phi\left((4k-1)z(obs)/\sqrt{n}\right)\right]$$
$$+ \sum_{k=(-n/z(obs)-3)/4}^{(n/z(obs)-1)/4}\left[\phi\left((4k+3)z(obs)/\sqrt{n}\right) - \phi\left((4k+1)z(obs)/\sqrt{n}\right)\right]$$

5. Oblicz wynik testu: ACCEPT/REJECT = (*p-value* ≥ α).

UWAGI:

- kumulowana suma, przy kodowaniu bitów ±1, może być traktowana jako losowa wycieczka;
- dla losowego ciągu trasa wycieczki (wartość S_k) powinna oscylować blisko zera;
- rozkład odniesienia – normalny;
- duża wartość statystyki testu świadczy o zbyt wielu zerach lub zbyt wielu jedynkach na początku (*mode* = 0) albo końcu (*mode* = 1) próbki;
- mała wartość statystyki testu świadczy o zbyt równomiernym przemieszaniu zer i jedynek.

Test 14. Test losowych wycieczek (ang. *Random Excursions Test*) {**random-excursions**} – sprawdza czy liczby cykli o *k* wystąpieniach stanu *x* w losowej wycieczce, określonej przez próbkę, odpowiadają sekwencji losowej.

FUNKCJA: *RandomExcursions*(*n*)

WEJŚCIE: $n \geq 10^6$

WYJŚCIE: (*p-value* ≥ α) dla stanu $x = -4, -3, -2, -1, 1, 2, 3, 4$ (8 wyników)

PROCEDURA:

1. Dla $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ utwórz ciąg: $X = X_1, X_2, \ldots, X_n$, gdzie $X_i = 2\varepsilon_i - 1$.
2. Dla $i = 1, 2, \ldots, n$ oblicz częściowe sumy: $S_i = X_1 + X_2 + \ldots + X_i$.
3. Utwórz ciąg $S' = 0, S_1, \ldots, S_n, 0 = (S'_{i'})$, dla $i' = 0, 1, \ldots, n + 1$.
4. Oblicz liczbę J cykli i określ cykle C_j ($j = 1, 2, \ldots, J$):

   $$J = \#\,(S'_{i'} = 0) - 1,\ S' = 0, C_1, 0, C_2, \ldots, 0, C_J, 0.$$

   Jeśli $J < 500$ to podstaw *p-value* = 0 i przejdź do kroku 9.
5. Dla każdego stanu $x = -4, -3, -2, -1, 1, 2, 3, 4$ oraz dla każdego cyklu C_j ($j = 1, 2, \ldots, J$) oblicz częstość x w cyklu C_J:

   $$w_j(x) = \#\,(C_{j,j'} = x), \text{ gdzie } j' = 1, 2, \ldots, |C_j|.$$
6. Podstaw $K = 5$ i dla każdego stanu $x = -4, -3, -2, -1, 1, 2, 3, 4$ oraz dla $k = 0, 1, \ldots, K$ oblicz liczbę cykli, w których stan x występuje k razy:

   $$v_k(x) = \#\,(C_j: w_j(x) = k), \text{ jeśli } k = 0, 1, 2, 3, 4,$$
   $$v_k(x) = \#\,(C_j: w_j(x) \geq k), \text{ jeśli } k = 5.$$

7. Dla każdego stanu $x = -4, -3, -2, -1, 1, 2, 3, 4$ oblicz statystykę testu:

$$\chi^2(obs) = \sum_{k=0}^{K}(v_k(x) - J\pi_k(x))^2 / J\pi_k(x),$$

gdzie $\pi_k(x)$ to prawdopodobieństwo teoretyczne.

8. Dla każdego stanu $x = -4, -3, -2, -1, 1, 2, 3, 4$ oblicz *p-wartość*:

$$p\text{-}value = igamc(5/2, \chi^2(obs)/2).$$

9. Dla każdego stanu $x = -4, -3, -2, -1, 1, 2, 3, 4$ oblicz wynik testu:

ACCEPT/REJECT = (p-$value \geq \alpha$) – 8 wyników.

UWAGI:

- przez cykl w ciągu S' rozumiany jest podciąg niezerowych wartości pomiędzy kolejnymi wartościami zero;
- przy założeniu, że ε nie jest ciągiem pustym, w ciągu S' nie ma sąsiadujących zer, a zatem wszystkie cykle w S' są niepuste;
- liczba cykli w S' jest równa liczbie zer w S' pomniejszonej o jeden;
- wartości prawdopodobieństw teoretycznych π_k oraz sposób ich obliczania, przedstawiono w dokumentacji pakietu.

Test 15. Test wariantu losowych wycieczek (ang. *Random Excursions Variant Tes*t) {**random-excursions-variant**} – sprawdza czy sumaryczna liczba wystąpień stanu x w cyklach losowej wycieczki, określonej przez próbkę, odpowiada sekwencji losowej.

FUNKCJA: *RandomExcursionsVariant*(n)

WEJŚCIE: $n \geq 10^6$

WYJŚCIE: (p-$value \geq \alpha$) dla stanu $x = -9, -8, \ldots, -1, 1, 2, \ldots, 9$ (18 wyników)

PROCEDURA:

1. Dla $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ utwórz ciąg: $X = X_1, X_2, \ldots, X_n$, gdzie $X_i = 2\varepsilon_i - 1$.
2. Dla $i = 1, 2, \ldots, n$ oblicz częściowe sumy: $S_i = X_1 + X_2 + \ldots + X_i$.
3. Utwórz ciąg $S' = 0, S_1, \ldots, S_n, 0 = (S'_{i'})$, dla $i' = 0, 1, \ldots, n + 1$.
4. Oblicz liczbę J cykli i określ cykle C_j ($j = 1, 2, \ldots, J$):

$$J = \#(S'_{i'} = 0) - 1,\ S' = 0, C_1, 0, C_2, \ldots, 0, C_J, 0.$$

Jeśli $J < 500$ to podstaw p-$value = 0$ i przejdź do kroku 7.

5. Dla każdego stanu $x = -9, -8, \ldots, -1, 1, 2, \ldots, 9$ oblicz statystykę testu:

$$\xi(x)(obs) = \#(S'_{i'} = x),$$

tj. częstość x w S' (sumaryczną w cyklach $C_1, C_2, \ldots, C_J$).

6. Dla każdego stanu $x = -9, -8, \ldots, -1, 1, 2, \ldots, 9$ oblicz *p-wartość*:

$$p\text{-}value = erfc(|\xi(x)(obs) - J| \,/\, \sqrt{2J(4|x| - 2)}).$$

7. Dla każdego stanu x = –9, –8, …, –1, 1, 2, …, 9 oblicz wynik testu:

$$\text{ACCEPT/REJECT} = (p\text{-}value \geq \alpha) \text{ – 18 wyników}$$

UWAGI:

- kroki od 1 do 4 testu są takie same jak w teście 14;
- definicję cyklu podano w uwagach do testu 14;
- rozkład odniesienia – półnormalny.

```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------

------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
108  92 105 111  95 122  68 115  83 101  0.005721   0.9860    frequency
 91 114 131  95 102 105  94 102  81  85  0.023705   0.9900    block-frequency
112  92  80 102 106 108 113  93  77 117  0.041709   0.9890    cumulative-sums
110  91  88 113 107 108 106  87  84 106  0.272977   0.9870    cumulative-sums
115  93  97 107  99 110  76 105 107  91  0.234373   0.9880    runs
110  69  92 109 163  79  89  95  92 102  0.000000 * 0.9860    fft
 97  98 108 104  95  97 104  90 118  89  0.649612   0.9880    apen
100 109  97  84 124  85 102 119  91  89  0.046568   0.9880    serial
102 100 109  98 110 110  90  85  94 102  0.684890   0.9880    serial


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
The minimum pass rate for each statistical test with the exception of the random
excursion (variant) test is approximately = 0.980561 for a sample size = 1000
binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Rys. 4.1. Przykład raportu z testów NIST SP800-22 – zawartość pliku *finalAnalysisReport.txt*

Zbiorcze rezultaty testów zapisywane są w pliku *finalAnalysisReport.txt* (rys. 4.1). Wyniki dotyczą dwóch kryteriów oceny losowości/pseudolosowości testowanego generatora. Pierwsze kryterium to iloraz liczby próbek spełniających dany test do liczby wszystkich próbek (ang. *proportion of squences passing a test*). Iloraz ten nazywany jest też krótko proporcją i oznaczany jako p. Np. dla m = 1000, α = 0.01 i 996 sekwencjach o *p-wartościach* $\geq \alpha$, wartość p = 996 / 1000 = 0.9960.

W ramach danego testu statystycznego otrzymany wynik (proporcja) musi należeć do przedziału akceptacji (PA), inaczej przedział ufności (ang. *confidence interval*), by uznać testowany generator za losowy/pseudolosowy. Dla wszystkich przeprowadzonych w rozprawie testów poziom istotności (PI) wynosi α = 0.01, zatem przedział akceptacji wynosi PA = <0.9805607; 0.9994392> – wyliczany zgodnie z formułą [95]:

$$\text{PA} = \hat{p} \pm 3\sqrt{\frac{\hat{p}\,(1-\hat{p})}{m}} \tag{4.5}$$

gdzie $\hat{p} = 1 - \alpha$, a m to liczba próbek.

Ze względu na przedstawianie wartości proporcji p z dokładnością do 4 cyfr po przecinku, jako przedział akceptacji przyjmujemy PA = <0.9806; 0.9994>. Przedział akceptacji został obliczony z wykorzystaniem rozkładu normalnego jako aproksymacji rozkładu dwumianowego, co jest uzasadnione dla dużej liczby próbek, np. $n \geq 1000$.

Drugie kryterium oceny losowości/pseudolosowości generatora to równomierny rozkład *p-wartości* (ang. *uniform distribution of p-values*). W obliczeniach wykorzystywany jest wzór:

$$\chi^2 = \sum_{i=1}^{10} \frac{(C_i - m/10)^2}{m/10} \tag{4.6}$$

gdzie s oznacza liczbę próbek, a C_i – liczbę *p-wartości* w przedziale $[(i - 1) / 10, i / 10]$, $i = 1, 2, \ldots, 10$. Stosowaną miarą losowości/pseudolosowości generatora jest *p-wartość*, obliczona zgodnie ze wzorem:

$$p\text{-}wartość = igamc(9 / 2, \chi^2 / 2). \tag{4.7}$$

Rozkład *p-wartości* uznawany jest za równomierny jeśli obliczona zgodnie ze wzorem (4.7) *p-wartość* ≥ 0.0001.

Alternatywne, wykorzystywane pakiety testów statystycznych to m.in.:

- pakiet DIEHARD [82] – autorstwa Georga Marsagalie, który zawiera 19 testów, a minimalny wymagany rozmiar danych wejściowych to plik o wielkości rzędu 10-11MB, czyli o rozmiarze co najmniej 88 milionów bitów,
- pakiet ENT [106] – autorstwa Johna Walkera (autora AutoCada i Autodeska), zawierający 6 testów statystycznych,
- pakiet DIEHARDER [37] – opracowany przez Roberta G. Browna, zawierający zbiór testów obejmujący zmodyfikowane (ulepszone) testy pakietu DIEHARD, 3 testy z pakietu NIST SP800-22 oraz testy własne.

4.3. Badania i eksperymenty

W podrozdziale przedstawiono wyniki badań przeprowadzonych przez autora rozprawy. Zaproponowano pięć metod testowania i oceny jakości algorytmów generowania kluczy rundowych. Przedstawiono wyniki wybranych testów statystycznych NIST SP800-22 algorytmu generowania kluczy rundowych szyfru DES, IDEA, KASUMI, PP-1, PP-2 oraz wyniki badań nad różnymi wariantami (modyfikacjami) algorytmu generowania kluczy rundowych szyfru PP-1 oraz IDEA. Przedstawiono również wyniki wszystkich testów statystycznych pakietu dla algorytmu generowania kluczy rundowych szyfru PP-1 oraz PP-2

możliwe do osiągniecia dzięki zmianie warunku stopu w algorytmach. Zaproponowano metodę pozwalającą na wykonanie wszystkich testów pakietu NIST SP800-22, niezależną od możliwości zmiany warunku stopu i zastosowano ją do algorytmu generowania kluczy rundowych w szyfrach DES, IDEA, KASUMI, PP-1 i PP-2. Dla tych algorytmów rozważono również metodę wykrywania słabości statystycznych początkowych kluczy rundowych, z wykorzystaniem testów pakietu. Zaproponowano także metodę hybrydową, z wykorzystaniem testów NIST SP800-22 oraz analizy skupień, w celu wyznaczenia optymalnego wariantu badanego algorytmu generowania kluczy rundowych. Stosowalność metody zaprezentowano na przykładzie algorytmów IDEA oraz PP-1.

4.3.1. Charakterystyka ogólna

Jako kryterium oceny jakości algorytmu generowania kluczy rundowych przyjęto jego właściwości statystyczne, zbadane rekomendowanym przez NIST pakietem NIST SP800-22. Przyjmijmy jako dane wejściowe dla pakietu NIST SP800-22, próbki T_1, T_2,…, T_z dla kluczy głównych MK_1, MK_2, …, MK_z, gdzie z oznacza jednocześnie liczbę próbek oraz liczbę kluczy głównych. Pojedyncza próbka T_i (i = 1, 2, …, z) jest sekwencją bitów, będącą konkatenacją wszystkich kluczy rundowych wygenerowanych z pojedynczego klucza głównego MK_i:

$$T_i = k_{i,1}||k_{i,2}|| \dots ||k_{i,nrk}, \tag{4.8}$$

gdzie *nrk* to liczba kluczy rundowych.

Przez wariant losowych kluczy głównych, rozumiany jest sposób wyboru tych kluczy, zgodny z formułą:

$$MK_i = random(0, 2^b - 1), \text{dla } i = 1, 2, \dots, z, \tag{4.9}$$

gdzie $b = |MK_i|$, tj. b jest długością klucza MK_i w bitach.

Przez wariant kolejnych kluczy głównych, rozumiany jest sposób ich wyboru, określony formułą:

$$MK_i = \begin{cases} random(0, 2^b - 1), & \text{dla } i = 1 \\ (MK_{i-1} + 1) \textbf{ mod } 2^b, & \text{dla } i = 2, 3, \dots, z \end{cases}, \tag{4.10}$$

gdzie $b = |MK_i|$.

Funkcja *random*(0, 2^b–1), generuje liczby pseudolosowe z przedziału <0, 2^b–1> i wykorzystuje generator liczb pseudolosowych, zaimplementowany w języku programowania C++.

W kolejnych punktach niniejszego podrozdziału niezmienną wartość mają następujące parametry:

- poziom istotności: $\alpha = 0.01$,
- przedział akceptacji: PA = <0.9806; 0.9994>,
- liczba próbek: $z = 1000$.

Wartości tych parametrów są przypominane w celu ułatwienia analizy wyników, tabel i rysunków.

Przy opisie każdego szyfru w rozdziale 3, w tabelach zawierających zestawienie cech szyfru, podano wartości następujących parametrów:

- *nrk* – liczba kluczy rundowych,
- |*rk*| – długość klucza rundowego w bitach,
- |*rkp*| – długość próbki kluczy rundowych w bitach ($|rkp| = nrk \cdot |rk|$).

W odniesieniu do badanych sekwencji bitów zastosowano następujące określenia:

- standardowa próbka (próbka) – konkatenacja *nrk* kluczy rundowych,
- wydłużona próbka (nadpróbka) – konkatenacja nrk_1 kluczy rundowych dla $nrk_1 > nrk$,
- skrócona próbka (podpróbka) – konkatenacja nrk_2 kluczy rundowych dla $nrk_2 < nrk$,
- złożona próbka (metapróbka) – sekwencja $T_i = T_{i,1}||T_{i,2}||\ldots||T_{i,l}$, gdzie $l = 2, 3, \ldots$, a $T_{i,j}$ ($j = 1, 2, \ldots, l$) jest próbką, nadpróbką lub podpróbką.

Autor rozprawy wyróżnia następujące metody oceny jakości algorytmu generowania kluczy rundowych:

- metoda A (próbek),
- metoda B (nadpróbek),
- metoda C (metapróbek),
- metoda D (podpróbek),
- metoda E (hybrydowa).

4.3.2. Metoda A (próbek)

Metoda A (próbek) polega na przebadaniu testami NIST SP800-22 standardowych próbek, tj. konkatenacji liczby kluczy rundowych określonych w specyfikacji szyfru. Ze

względu na niewielką długości próbki, możliwe do wykonania są tylko niektóre testy, w zależności od rekomendowanej długości próbki.

A1. Ocena algorytmu generowania kluczy rundowych

Przykładem zastosowania metody A są badania autora przeprowadzone nad algorytmami generowania kluczy rundowych szyfrów: DES, IDEA, KASUMI, PP-1 i PP-2 [3]. Dla różnych szyfrów i odpowiadających im algorytmów generowania kluczy rundowych, długość standardowej próbki jest różna, co wynika z różnej liczby i różnej długości kluczy rundowych. Dla rozważanych szyfrów długość próbki jest następująca:

- DES: 768 bitów (konkatenacja 16 kluczy rundowych o długości 48 bitów),
- IDEA: 832 bity (konkatenacja 52 kluczy rundowych o długości 16 bitów),
- KASUMI: 1024 bity (konkatenacja 64 kluczy rundowych o długości 16 bitów),
- PP-1 64/128: 1408 bitów (konkatenacja 22 kluczy rundowych o długości 64 bity) – testowaną wersją szyfru PP-1 była wersja przetwarzająca 64-bitowe bloki danych z wykorzystaniem 128-bitowego klucza głównego,
- PP-2 64/128: 832 bity (konkatenacja 13 kluczy rundowych o długości 64 bity) – testowaną wersją szyfru PP-2 była wersja przetwarzająca 64-bitowe bloki danych z wykorzystaniem 128-bitowego klucza głównego.

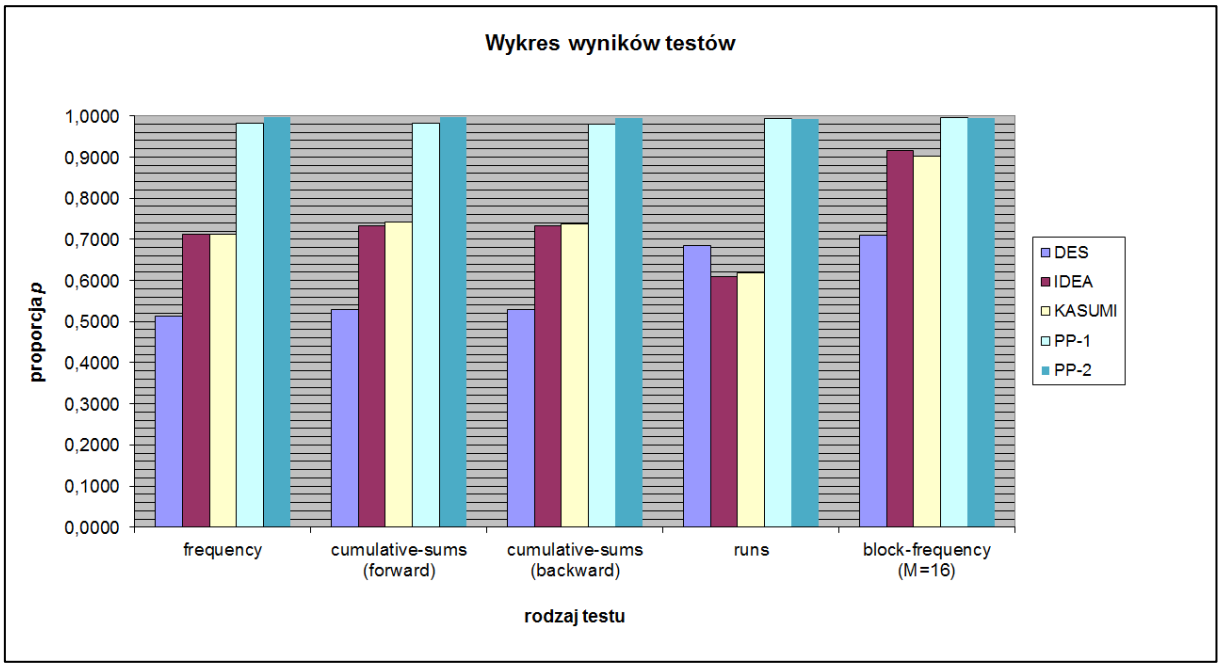
W przeprowadzonych badaniach zastosowano wariant losowych kluczy głównych (formuła (4.9)) i wykonano następujące testy:

- test częstości {frequnecy}, $n \geq 100$,
- test kumulowanych sum (wprzód) {cumulative-sums (forward)}, $n \geq 100$,
- test kumulowanych sum (wstecz) {cumulative-sums (backward)}, $n \geq 100$,
- test serii {runs}, $n \geq 100$,
- test częstości w blokach {block-frequnecy ($M = 16$)}, $n \geq 100$.

Tab. 4.2. Zestawienie wyników *p* testów NIST SP800-22 przy zastosowaniu metody A (PA = <0.9806; 0.9994>)

rodzaj testu	algorytm generowania kluczy rundowych				
	DES	IDEA	KASUMI	PP-1	PP-2
frequency	0.5120	0.7120	0.7120	0.9830	0.9980
cumulative-sums (forward)	0.5290	0.7320	0.7410	0.9820	0.9970
cumulative-sums (backward)	0.5280	0.7320	0.7370	0.9810	0.9950
runs	0.6850	0.6100	0.6180	0.9930	0.9930
block-frequency ($M = 16$)	0.7100	0.9160	0.9010	0.9950	0.9940

Przedstawione w tabeli (tab. 4.2) i na rysunku (rys. 4.2) wyniki (proporcje) *p*, należy interpretować jako iloraz liczby próbek spełniających dany test do liczby wszystkich próbek *z* = 1000. Dla danego szyfru, wartości proporcji *p* muszą należeć do przedziału akceptacji PA, aby algorytm generowania kluczy rundowych tego szyfru można było uznać za generator ciągów pseudolosowych.



Rys. 4.2. Wykres wyników *p* testów NIST SP800-22 przy zastosowaniu metody A (PA = <0.9806; 0.9994>)

Wniosek 4.1.

W metodzie A, przy losowych wartościach kluczy głównych (formuła (4.9)), algorytm generowania kluczy rundowych w szyfrach DES, IDEA, KASUMI, nie spełnia kryterium jakości (losowości), a w szyfrach PP-1 i PP-2 spełnia.

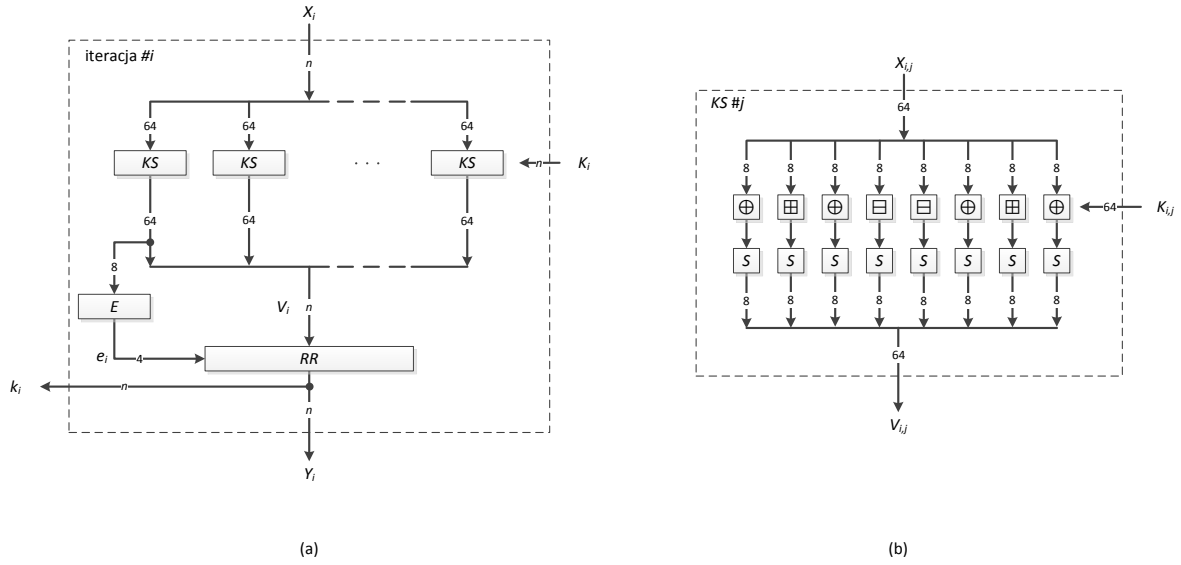
■

Z przeprowadzonych testów wynika, że ciągi generowane przez algorytm generowania kluczy rundowych w różnych szyfrach blokowych posiadają znaczące różnice jakościowe. Dla szyfrów DES, IDEA i KASUMI uzyskano negatywne wyniki wykonanych testów, a dla szyfrów PP-1 i PP-2 – wyniki pozytywne. Należy zwrócić uwagę na fakt, że algorytm generowania kluczy rundowych w szyfrach blokowych DES, IDEA oraz KASUMI opiera się wyłącznie na operacjach liniowych, takich jak rotacja, permutacja bitowa oraz suma wyłączająca (XOR), natomiast algorytm generowania kluczy w szyfrach PP-1 oraz PP-2 wykorzystuje operacje nieliniowe (podstawienia z wykorzystaniem S-bloków).

A2. Projektowanie algorytmu generowania kluczy rundowych

Innym przykładem zastosowania metody A są przeprowadzone eksperymenty nad zmodyfikowaną konstrukcją algorytmu generowania kluczy rundowych szyfru PP-1 64/64 [4]. Celem jest uproszczenie oraz przyspieszenie procesu generowania kluczy rundowych przy jednoczesnym zachowaniu ich dobrych własności statystycznych. Dla wszystkich wersji algorytmu, zastosowano wariant losowych kluczy głównych i przeprowadzono te same testy statystyczne. Podkreśleniem zaznaczono wyniki spoza przedziału akceptacji.

<u>Wersja (0) oryginalna algorytmu</u>



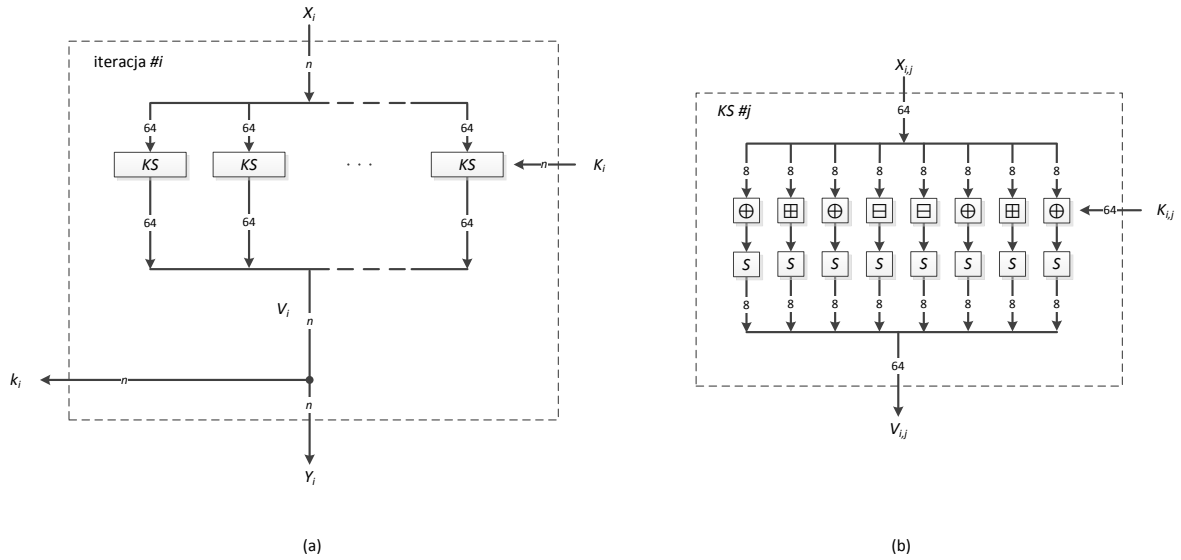
Rys. 4.3. PP-1 – wersja (0) oryginalna algorytmu generowania kluczy rundowych, struktura: (a) iteracji #*i* (*i* = 0, 1, …, 2*r*) funkcji generowania kluczy rundowych, (b) elementu *KS #j* (*j* = 1, 2, …, *t*)

Rysunek (rys. 4.3) przedstawia oryginalny algorytm generowania kluczy rundowych szyfru PP-1. Wszystkie testy, przeprowadzone na wygenerowanych ciągach (próbkach) o długości 1408 bitów, zakończyły się sukcesem (PA = <0.9806; 0.9994>):

- test częstości {frequency}, $n \geq 100$: **0.9890**,
- test częstości w blokach {block-frequency ($M = 4$)}, $n \geq 100$: **0.9970**,
- test kumulowanych sum (wprzód) {umulative-sums (forward)}, $n \geq 100$: **0.9910**,
- test kumulowanych sum (wstecz) {cumulative-sums (backward)}, $n \geq 100$: **0.9900**,
- test serii {runs}, $n \geq 100$: **0.9860**,
- test transformaty Fouriera {fft}, $n \geq 1000$: **0.9890**,
- test przybliżonej entropi {apen ($m = 4$)}, $n \geq 1024$: **0.9940**,
- test seryjny 1 {serial1 ($m = 4$)}, $n \geq 128$: **0.9910**,

- test seryjny 2 {serial2 (m = 4)}, $n \geq 128$: **0.9890**.

Wersja 1 algorytmu bez rotacji *RR*

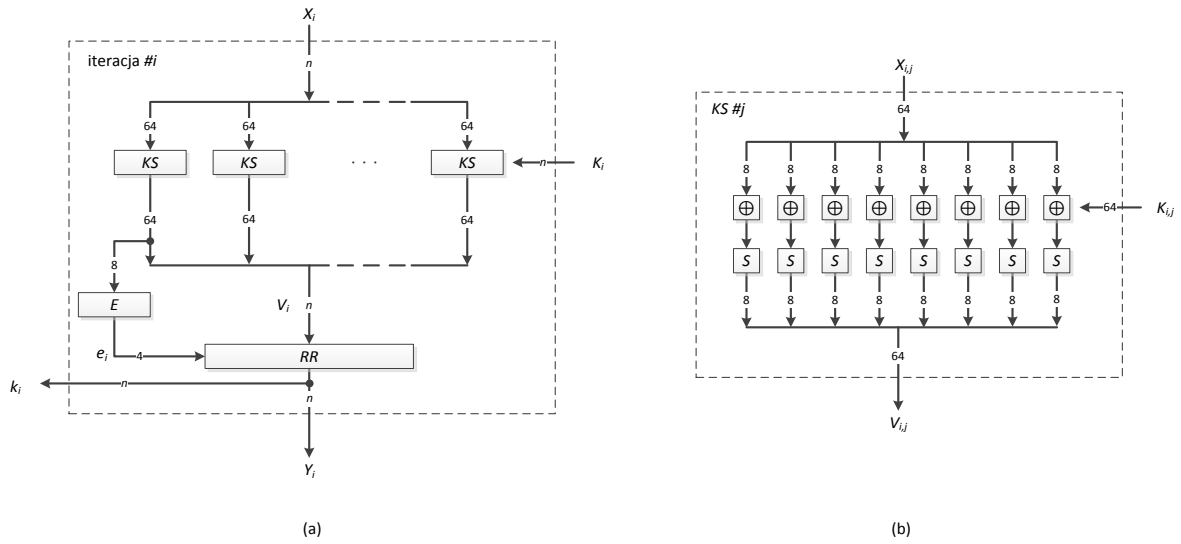


Rys. 4.4. PP-1 – wersja 1 algorytmu generowania kluczy rundowych bez rotacji *RR*, struktura: (a) iteracji #*i* (*i* = 0, 1, …, 2*r*) funkcji generowania kluczy rundowych, (b) elementu *KS #j* (*j* = 1, 2, …, *t*)

Rysunek (rys. 4.4) przedstawia zmodyfikowany algorytm z usuniętą operacją *RR*. Element KS w prezentowanym algorytmie generowania kluczy pozostał bez zmian. Wszystkie testy, które przeprowadzono na wygenerowanych ciągach o długości 1408 bitów zakończyły się sukcesem (PA = <0.9806; 0.9994>):

- frequency: 0.9890,
- block-frequency (M = 4): 0.9880,
- cumulative-sums (forward): 0.9910,
- cumulative-sums (backward): 0.9900,
- runs: 0.9840,
- fft: 0.9850,
- apen (m = 4): 0.9960,
- serial1 (m = 4): 0.9960,
- serial2 (m = 4): 0.9950.

Wersja 2 algorytmu z operacjami XOR



Rys. 4.5. PP-1 – wersja 2 algorytmu generowania kluczy rundowych z operacjami XOR zamiast sumy i różnicy modulo 256, struktura: (a) iteracji #*i* (*i* = 0, 1, …, 2*r*) funkcji generowania kluczy rundowych, (b) elementu *KS* #*j* (*j* = 1, 2, …, *t*)

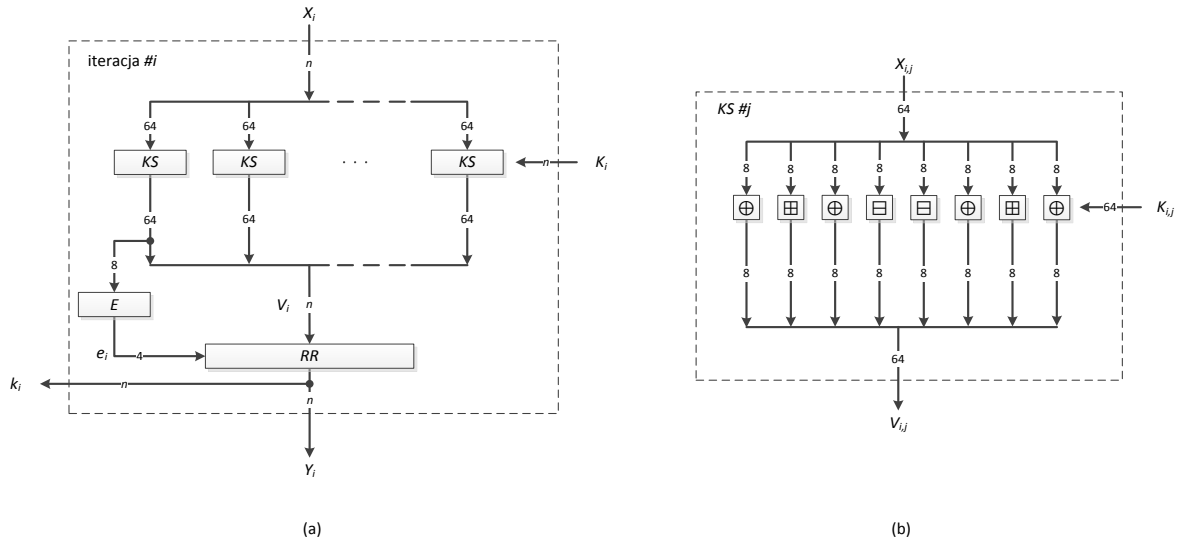
Rysunek (rys. 4.5) przedstawia zmodyfikowany algorytm generowania kluczy, w którym warstwa dodania podklucza pomocniczego $K_{i,j}$ zawiera operacje XOR, zamiast operacji sumy i różnicy modulo 256. Zmiana dotyczy elementu *KS*, a uzyskane wyniki są następujące (PA = <0.9806; 0.9994>):

- frequency: 0.9890,
- block-frequency (*M* = 4): 0.9940,
- cumulative-sums (forwarad): 0.9930,
- cumulative-sums (backward): 0.9880,
- runs: 0.9880,
- fft: 0.9880,
- apen (*m* = 4): 0.9920,
- serial1 (*m* = 4): 0.9950,
- serial2 (*m* = 4): 0.9800.

W przypadku testu serial2, pomimo wyniku (proporcji) spoza przedziału akceptacji PA, w zbiorczym raporcie pakietu NIST SP800-22 ten wynik nie został oznaczony jako negatywny. W związku z tym, w tabeli (tab. 4.3), zaliczono ten wynik jako pozytywny.

Wersja 3 algorytmu ze zredukowaną liczbą S-bloków

W wersji algorytmu ze zredukowaną liczbą S-bloków, zredukowano liczbę operacji podstawień, w stosunku do oryginalnego algorytmu w elemencie KS, całkowicie (rys. 4.6), do czterech (rys. 4.7) oraz do sześciu (rys. 4.8) S-bloków.



Rys. 4.6. PP-1 – wersja 3.1 algorytmu generowania kluczy rundowych bez S-bloków, struktura: (a) iteracji #*i* (*i* = 0, 1, …, 2*r*) funkcji generowania kluczy rundowych, (b) elementu *KS #j* (*j* = 1, 2, …, *t*)

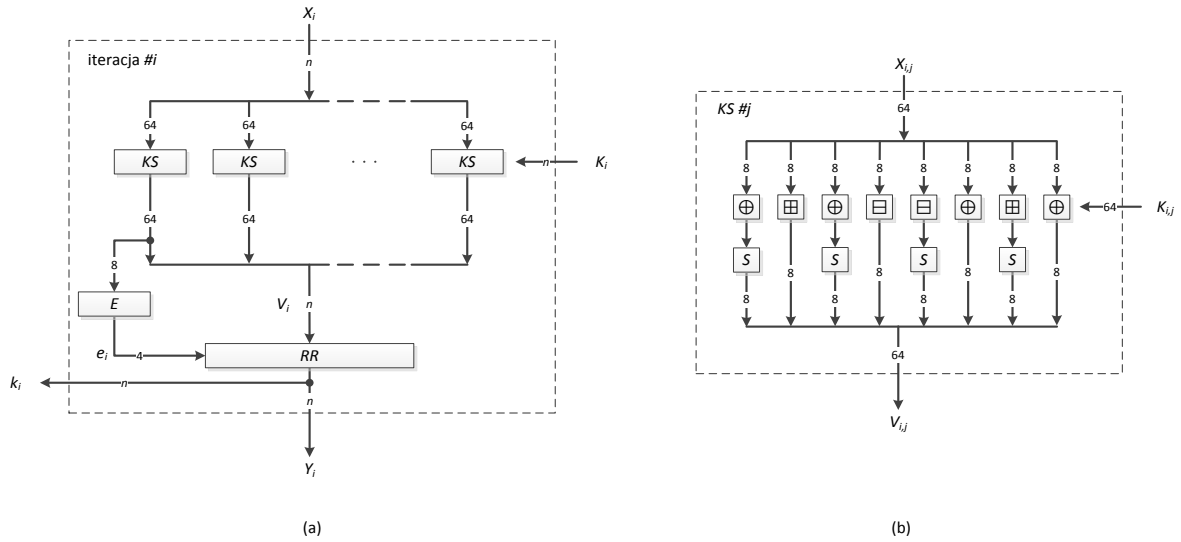
Wersja 3.1 algorytmu bez S-bloków przeszła pomyślnie większość testów, a test entropii oraz test serii z parametrem dały wynik negatywny (PA = <0.9806; 0.9994>):

- frequency: 0.9970,
- block-frequency (*M* = 4): 0.9980,
- cumulative-sums (forward): 0.9960,
- cumulative-sums (backward) 0.9960,
- runs: 0.9960,
- fft: 0.9860,
- apen (*m* = 4): 0.9740,
- serial1 (*m* = 4): 0.9870,
- serial2 (*m* = 4): 0.9730.

Test powtórzono dla wersji 3.2 algorytmu z czterema S-blokami transformującymi 1, 3, 5 i 7 bajt danych (rys. 4.7). Wyniki dwóch testów były negatywne (PA = <0.9806; 0.9994>):

- frequency: 0.9960,
- block-frequency (*M* = 4): 0.9970,

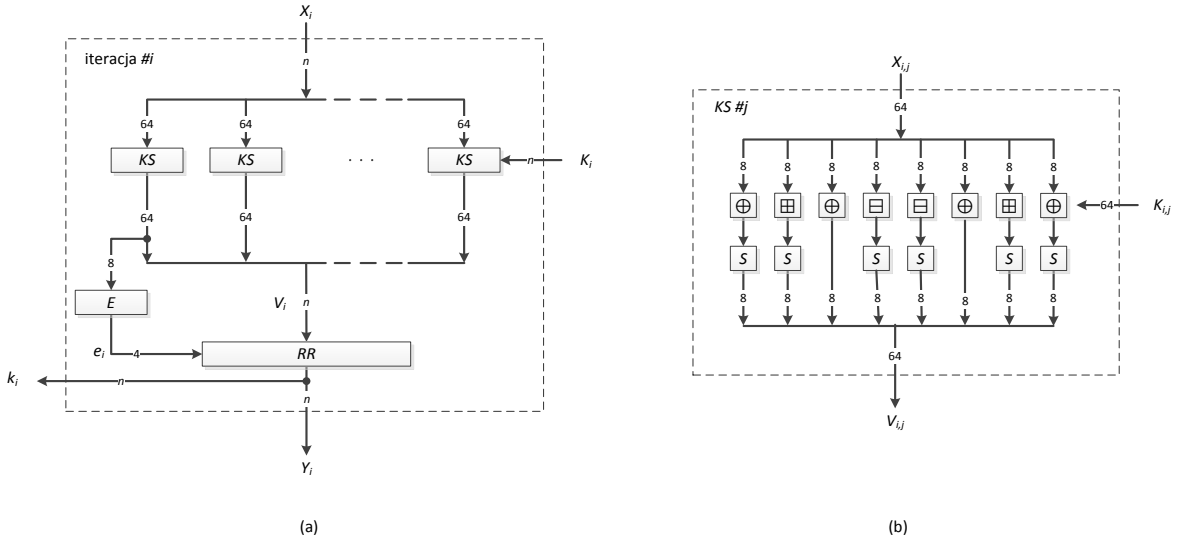
- cumulative-sums (forward): 0.9970,
- cumulative-sums (backward) 0.9940,
- runs: 0.9740,
- fft: 0.9930,
- apen (m = 4): 0.9810,
- serial1 (m = 4): 0.9740,
- serial2 (m = 4): 0.9860.



Rys. 4.7. PP-1 – wersja 3.2 algorytmu generowania kluczy rundowych z 4 S-blokami, struktura: (a) iteracji #*i* (*i* = 0, 1, …, 2*r*) funkcji generowania kluczy rundowych, (b) elementu *KS* #*j* (*j* = 1, 2, …, *t*)

Dla przedstawionej na rysunku (rys. 4.8) wersji 3.3 algorytmu z sześcioma S-blokami wyniki wszystkich przeprowadzonych testów były pozytywne (PA = <0.9806; 0.9994>):

- frequency: 0.9960,
- block-frequency (M = 4): 0.9990,
- cumulative-sums (forward): 0.9960,
- cumulative-sums (backward) 0.9980,
- runs: 0.9950,
- fft: 0.9860,
- apen (m = 4): 0.9910,
- serial1 (m = 4): 0.9890,
- serial2 (m = 4): 0.9880.



Rys. 4.8. PP-1 – wersja 3.3 algorytmu generowania kluczy rundowych z 6 S-blokami, struktura: (a) iteracji #*i* (*i* = 0, 1, …, 2*r*) funkcji generowania kluczy rundowych, (b) elementu *KS #j* (*j* = 1, 2, …, *t*)

Wersja 4 algorytmu z modyfikacją stałej B

Inną z testowanych wersji algorytmu generowania kluczy rundowych jest wersja 4.1 ze zmianą oryginalnej stałej B_1=0x912B4769B2496E7C na wartość B_1=0x0101010101010101. Pozostałe elementy algorytmu pozostały bez zmian. Wyniki większości testów były negatywne (PA = <0.9806; 0.9994>):

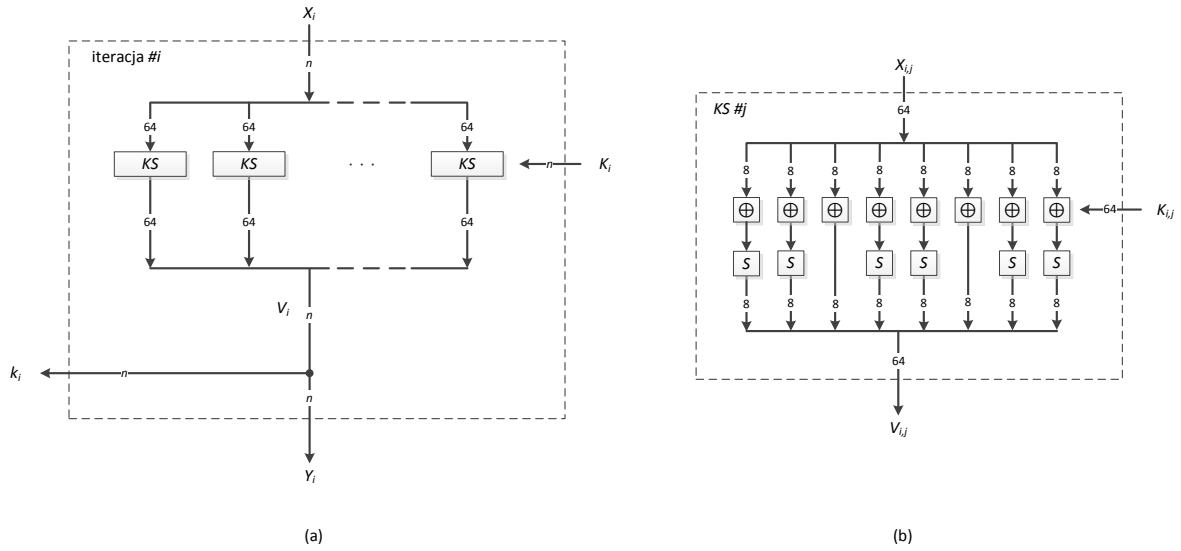
- frequency: 0.9450,
- block-frequency (M = 4): 0.9920,
- cumulative-sums (forward): 0.9380,
- cumulative-sums (forward): 0.9380,
- runs: 0.9950,
- fft: 0.9810,
- apen (m = 4): 0.8770,
- serial1 (m = 4): 0.9160,
- serial2 (m = 4): 0.9620.

W wersji 4.2. testy powtórzono dla stałej o wartości B_1=0x0F0F0F0F0F0F0F0F. Wyniki dwóch testów były negatywne (PA = <0.9806; 0.9994>):

- frequency: 0.9960,
- block-frequency (M = 4): 0.8940,
- cumulative-sums (forward): 0.9970,

- cumulative-sums (backward): 0.9970,
- runs: 0.9960,
- fft: 0.9850,
- apen (m = 4): 0.9580,
- serial1 (m = 4): 0.9920,
- serial2 (m = 4): 0.9870.

Wersja 5 algorytmu z pozytywnymi (prostymi) modyfikacjami



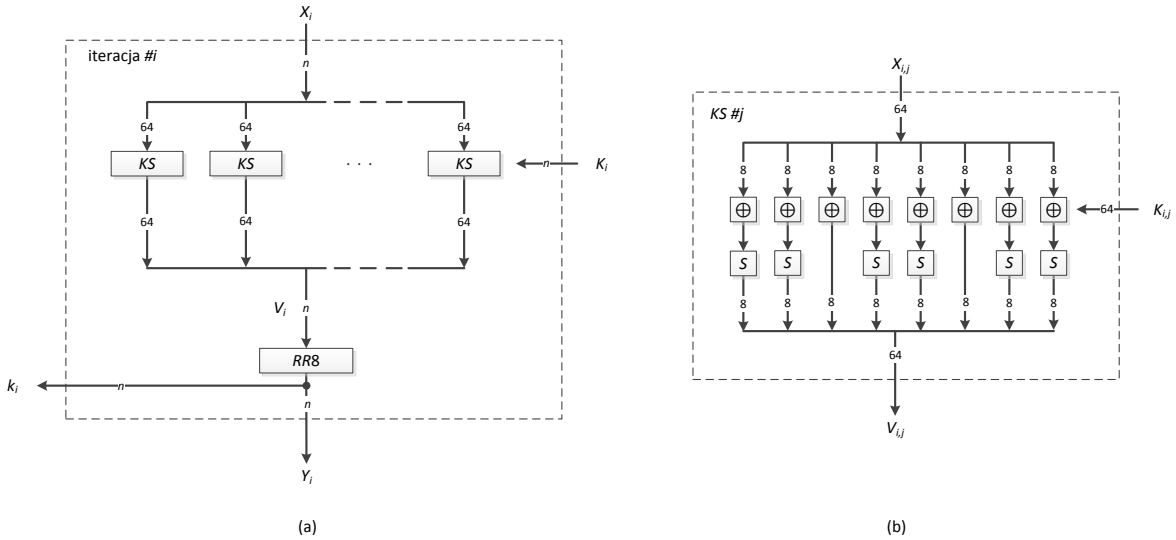
Rys. 4.9. PP1 – wersja 5 algorytmu generowania kluczy rundowych złożona z pozytywnych (prostych) modyfikacji, struktura: (a) iteracji #*i* (*i* = 0, 1, …, 2*r*) funkcji generowania kluczy rundowych, (b) elementu *KS #j* (*j* = 1, 2, …, *t*)

Przedstawiona na rysunku (rys. 4.9) wersja 5 algorytmu generowania kluczy rundowych to złożenie uproszczonych wersji o pozytywnych wynikach wszystkich testów. W wersji 5 algorytmu usunięto rotację *RR*, zastąpiono operacje sumy i różnicy modulo 256 operacjami XOR oraz zredukowano liczbę S-bloków do sześciu. Wszystkie otrzymane wyniki były pozytywne (PA = <0.9806; 0.9994>):

- frequency: 0.9970,
- block-frequency (M = 4): 0.9960,
- cumulative-sums (forward): 0.9990,
- cumulative-sums (backward) 0.9970,
- runs: 0.9880,
- fft: 0.9840,
- apen (m = 4): 0.9900,

- serial1 (m = 4): 0.9900,
- serial2 (m = 4): 0.9920.

Wersja 6 algorytmu złożona z rotacją *RR*8

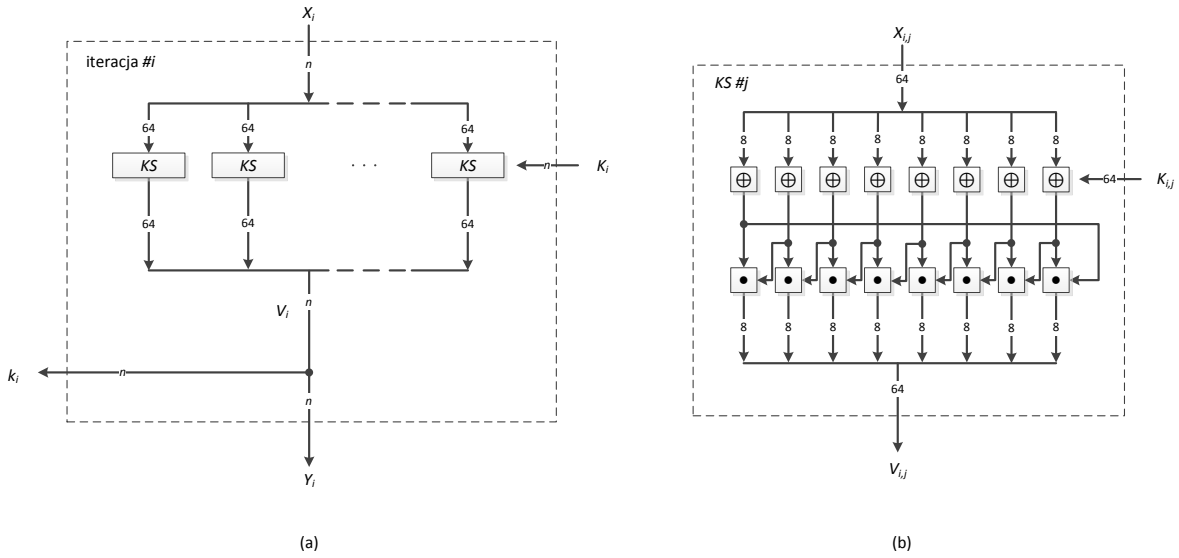


Rys. 4.10. PP-1 – wersja 6 algorytmu generowania kluczy rundowych złożona z operacji *RR*8, XOR i 6 S-bloków, struktura: (a) iteracji #*i* (*i* = 0, 1, …, 2*r*) funkcji generowania kluczy rundowych, (b) elementu *KS* #*j* (*j* = 1, 2, …, *t*)

Przedstawiona na rysunku (rys. 4.10) wersja 6 algorytmu jest analogiczna do poprzedniej wersji, ale dodatkowo z uwzględnieniem stałej rotacji *RR*8 tj. o 8 bitów w prawo. Z przeprowadzonych testów tylko test serii z parametrem dał wynik negatywny (PA = <0.9806; 0.9994>):

- frequency: 0.9940,
- block-frequency (M = 4): 0.9940,
- cumulative-sums (forward): 0.9900,
- cumulative-sums (backward): 0.9850,
- runs: 0.9960,
- fft: 0.9860,
- apen (m = 4): 0.9890,
- serial1 (m = 4): 0.9870,
- serial2 (m = 4): 0.9760.

Wersja 7 algorytmu złożona z mnożeniem



Rys. 4.11. PP-1 – wersja 7 algorytmu generowania kluczy rundowych złożona z mnożeniem, operacją XOR i bez rotacji *RR*, struktura: (a) iteracji #*i* (*i* = 0, 1, …, 2*r*) funkcji generowania kluczy rundowych, (b) elementu *KS* #*j* (*j* = 1, 2, …, *t*)

Przedstawiona na rysunku (rys. 4.11) modyfikacja w elemencie *KS* wykorzystuje operacje XOR w warstwie dodania klucza pomocniczego $K_{i,j}$, następnie wykonywane jest mnożenie (symbol kropki) *ab* **mod** 257 przy założeniu, że zeru odpowiada 256. W algorytmie generowania kluczy rundowych pominięto rotację *RR*. Dla opisywanej wersji 6 uzyskano następujące wyniki (PA = <0.9806; 0.9994>):

- frequency: 0.9860,
- block-frequency (*M* = 4): 0.9900,
- cumulative-sums (forward): 0.9830,
- cumulative-sums (backward): 0.9850,
- runs: 0.9790,
- fft: 0.9870,
- apen (*m* = 4): 0.9790,
- serial1 (*m* = 4): 0.9680,
- serial2 (*m* = 4): 0.9800.

W przypadku testu serial2, pomimo wyniku (proporcji) spoza przedziału akceptacji PA, w zbiorczym raporcie pakietu NIST SP800-22 ten wynik nie został oznaczony jako negatywny. W związku z tym, w tabeli (tab. 4.3), zaliczono ten wynik jako pozytywny.

Porównanie wersji algorytmu pod względem czasu obliczeń

Poniższe zestawienie przedstawia porównanie czasów generowania kluczy rundowych, nazywanych krótko czasami obliczeń, wybranych wersji algorytmów względem oryginalnego algorytmu. Ze względu na specyficzną implementację ukierunkowaną na generowanie próbek na potrzeby testów NIST SP800-22 zmierzone czasy uwzględniają cały proces generowania próbki (czyli czas trwania wygenerowania 1000 sekwencji po 1408 bitów oraz zapis kluczy do pliku) – dlatego należy zwrócić uwagę na różnice w czasach a nie same czasy generowania. Niech Δt oznacza różnice czasów pomiędzy oryginalnym algorytmem i jego uproszczoną wersją. Prezentowane czasy to średnia arytmetyczna z 5 uruchomień generatora kluczy:

- wersja (0) oryginalna: 2,091s,
- wersja 1 bez rotacji *RR*: 2,057s (Δt = 0,034s),
- wersja 2 z operacjami XOR, zamiast sumy i różnicy: 2,075s (Δt = 0,016s),
- wersja 3.1 bez S-bloków: 2,041s (Δt = 0,050s),
- wersja 3.2 z czterema S-blokami: 2,061s (Δt = 0,030s),
- wersja 3.3 z sześcioma S-blokami : 2.074s (Δt = 0,017s),
- wersja 4 z modyfikacją stałej *B*: 2,090s (Δt = 0,001s),
- wersja 5 złożona z pozytywnych (prostych) modyfikacji: 2,028s (Δt = 0,063s),
- wersja 6 złożona z rotacji *RR*8, XOR i 6 S-bloków: 2,061s (Δt = 0,030s),
- wersja 7 złożona z operacji mnożenia, XOR i bez *RR*: 2,019s (Δt = 0,072s).

Z przeprowadzonych testów wynika, że możliwe jest zredukowanie oryginalnego algorytmu generowania kluczy rundowych szyfru PP-1 64/64 do postaci z pozytywnymi modyfikacjami, bez pogarszania jakości generowanych kluczy rundowych, przyjmując jako kryterium oceny jakości testy statystyczne NIST SP800-22. Wszystkie przeprowadzone w tym wariancie testy, dla próbek o długości 1408 bitów, zakończyły się sukcesem. Wykonane testy pokazały również znaczenie zastosowania stałej *B* o dobrej jakości, która umożliwia wprowadzenie na wejście algorytmu generowania kluczy rundowych dobrego ciągu bitów nawet w przypadku wprowadzenia klucza głównego o niskiej jakości.

Podsumowanie wyników testów NIST SP800-22, dla analizowanych wersji algorytmu generowania kluczy rundowych w szyfrze PP-1 64/64, przedstawiono w tabeli (tab. 4.3).

Tab. 4.3. Podsumowanie testów NIST 800-22 dla różnych wersji algorytmu generowania kluczy rundowych w szyfrze PP-1 64/64

wersja algorytmu	rodzaj testu								
	frequency	block-frequency (M = 4)	cumulative-sums (forward)	cumulative-sums (backward)	runs	fft	apen (m = 4)	serial1 (m = 4)	serial2 (m = 4)
0 – oryginalna	+	+	+	+	+	+	+	+	+
1 – bez rotacji *RR*	+	+	+	+	+	+	+	+	+
2 – z operacjami XOR	+	+	+	+	+	+	+	+	+
3.1 – bez S-bloków	+	+	+	+	+	+	–	+	–
3.2 – z 4 S-blokami	+	+	+	+	–	+	+	–	+
3.3 – z 6 S-blokami	+	+	+	+	+	+	+	+	+
4.1 – ze stałą B_1=0x0101010101010101	–	+	–	–	+	+	–	–	–
4.2 – ze stałą B_1=0x0F0F0F0F0F0F0F0F	+	–	+	+	+	+	–	+	+
5 – złożona z pozytywnych modyfikacji (bez *RR*, z XOR, z 6 S-blokami)	+	+	+	+	+	+	+	+	+
6 – złożona z rotacją *RR*8 (z *RR*8, z XOR, z 6 S-blokami)	+	+	+	+	+	+	+	+	–
7 – złożona z mnożeniem (bez *RR*, z XOR, z mnożeniem)	+	+	+	+	–	+	–	–	+

Wniosek 4.2.

W metodzie A, przy losowych wartościach kluczy głównych (formuła (4.9)), w wariancie oryginalnym algorytmu generowania kluczy rundowych szyfru PP-1 64/64 oraz w jego wersjach (modyfikacjach): wersja 1 bez rotacji *RR*, wersja 2 z operacjami XOR, wersja 3.3 z sześcioma S-blokami i wersja 5 złożona z pozytywnych modyfikacji (bez *RR*, XOR, 6 S-bloków), spełnione jest kryterium jakości (losowości), a w pozostałych wersjach: 3.1, 3.2, 4.1, 4.2, 6, 7 – nie jest spełnione to kryterium.

■

4.3.3. Metoda B (nadpróbek)

Metoda B (nadpróbek) polega na przebadaniu testami NIST SP800-22 wydłużonych próbek, tj. konkatenacji liczby kluczy rundowych większej od liczby, określonej w definicji szyfru. Próbki wydłużono do rozmiaru przekraczającego 10^6 bitów, w celu wykonania wszystkich testów pakietu. Wydłużenie takie jest możliwe jeśli konstrukcja algorytmu generowania kluczy rundowych pozwala na zwiększenie liczby jego iteracji, bez modyfikacji elementów składowych algorytmu i w rezulacie na wygenerowanie większej liczby kluczy rundowych z pojedynczego klucza głównego.

Metoda B może być zastosowana w odniesieniu do algorytmów generowania kluczy rundowych, rozpatrywanych w rozprawie szyfrów: LOKI97 (wniosek 3.5), LCB-IoT (wniosek 3.13), AES (wniosek 3.15), Serpent (wniosek 3.17), SAFER+ (wniosek 3.19), PP-1 (wniosek 3.23), PP-2 (wniosek 3.25), IDEA (wniosek 3.28), RC6 (wniosek 3.30), Speck (wniosek 3.32), a nie może być zastosowana w przypadku szyfrów: DES (wniosek 3.1), 3DES (wniosek 3.3), SM4 (wniosek 3.7), KASUMI (wniosek 3.9), FeW (wniosek 3.11), PRESENT (wniosek 3.21).

Pakiet testów statystycznych NIST SP800-22, przeznaczonych jest do badania generatorów liczb losowych i pseudolosowych w zastosowaniach kryptograficznych. Algorytm generowania kluczy rundowych może być traktowany jako generator liczb pseudolosowych. Wejściem takiego generatora, inaczej ziarnem (ang. *seed*), jest klucz główny szyfru, a wyjściem, inaczej nadpróbką – ciąg kluczy rundowych. W idealnym przypadku, źródłem ziarna jest generator liczb losowych, ale najczęściej – generator liczb pseudolosowych. Do oceny generatora liczb pseudolosowych, za pomocą pakietu testów NIST SP800-22, zalecana jest duża liczba ziaren, $z \geq 1000$ i duża długość próbki, $n \geq 10^6$ bitów.

W metodzie B zastosowano wariant kolejnych kluczy głównych, określony formułą (4.10), a więc wykonano lokalne badanie przestrzeni tych kluczy.

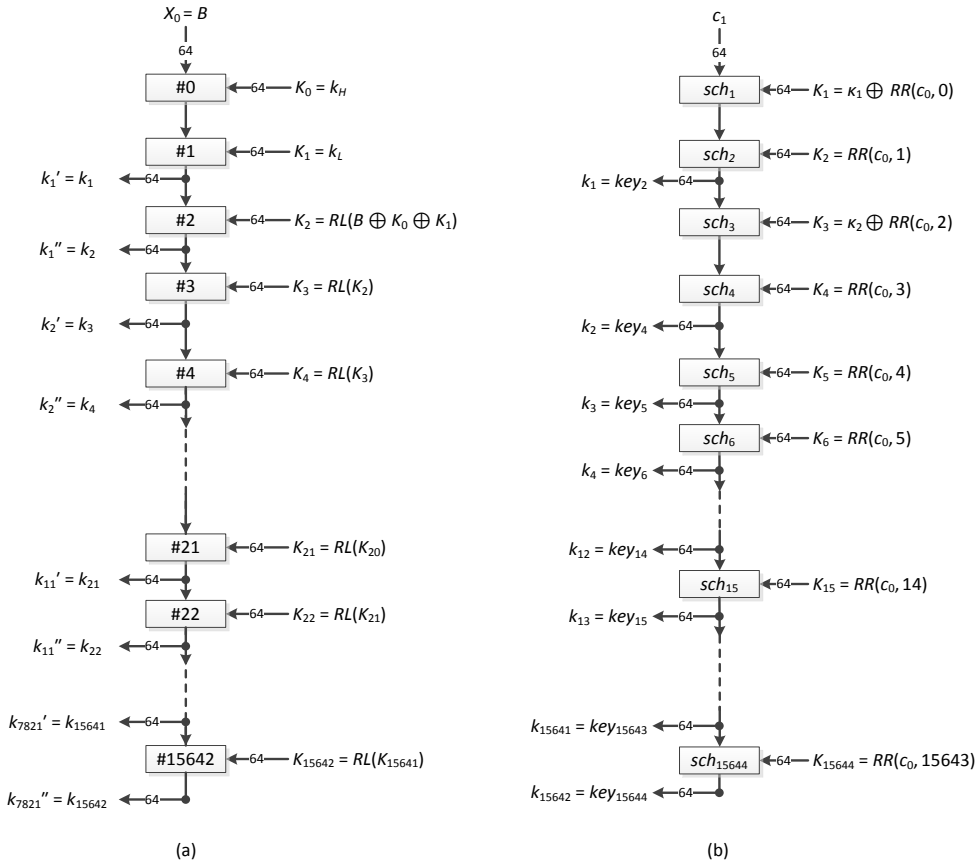
B1. Algorytm generowania kluczy rundowych szyfru PP-2 64/128, jako generator liczb pseudolosowych

Przykładem zastosowania metody B są badania, przeprowadzone przez autora, nad rozszerzonym (ze zmienionym warunkiem stopu) algorytmem generowania kluczy rundowych szyfru PP-2 64/128 [6].

Załóżmy, że algorytm generowania kluczy rundowych szyfru PP-2 64/128 (rys. 4.12 (b)), tj. z blokiem danych 64-bitowym oraz z kluczem 128-bitowym, wytwarza nie 13 lecz 15642 klucze rundowe o długości 64 bity. Wówczas, z pojedynczego klucza głównego, otrzymamy sekwencję (nadpróbkę) o długości $1001088 \geq 10^6$ bitów.

W tabeli (tab. 4.4) przedstawiono przykładowy zestaw kluczy rundowych, wygenerowanych ze 128-bitowego klucza głównego, złożonego z samych zer oraz klucza głównego zawierającego tylko jedną jedynkę. W obu zestawach klucze rundowe k_1 są sobie równe, ponieważ 64 bardziej znaczące bity klucza głównego są identyczne. Wyłącznie od wartości tych bitów oraz wartości stałych c_0 oraz c_1 zależy wartość klucza rundowego k_1, po

rundach rozbiegowych. Mniej znaczące 64 bity klucza głównego wprowadzane są do algorytmu dopiero po wytworzeniu klucza k_1 (rys. 4.12 (b)).



Rys. 4.12. Rozszerzony algorytm generowania kluczy rundowych szyfru: (a) PP-1 64/128, (b) PP-2 64/128, obliczający 15642 klucze rundowe

W tabeli (tab. 4.4) pokazano też, że liczby zer i jedynek, w sekwencjach utworzonych w wyniku konkatenacji kluczy rundowych uzyskanych z obu wartości klucza głównego, są zbliżone i bliskie wartości 1001088 / 2 = 500544.

Tab. 4.4. Przykładowe klucze rundowe z rozszerzonego algorytmu generowania kluczy rundowych szyfru PP-2 64/128

klucz główny	0x0000000000000000 0000000000000000	0x0000000000000000 0000000000000001
k_1	0x553DC29DD68BC880	0x553DC29DD68BC880
k_2	0xAFCF9688025D3E89	0xA74CA28992597E8D
…	…	…
k_{12}	0xF9A4A91B320E86C0	0x0278379D5C697624
k_{13}	0xAA279AD8EA6116D2	0x5D615BC3447D3DB9
…	…	…
k_{15641}	0x2B640E6B9B68411C	0xD2FD0FB00B0C16AA
k_{15642}	0x58D5FF55F364D989	0xF7F99FCAF84FCA57
liczba 0 w sekwencji	501344	500812
liczba 1 w sekwencji	499744	500276

Dla algorytmu generowania kluczy rundowych szyfru PP-2 64/128 otrzymano pozytywne wyniki wszystkich testów statystyczny NIST SP800-22, przedstawione w tabeli (tab. 4.5).

Tab. 4.5. Zestawienie wyników *p* testów NIST SP800-22 dla rozszerzonych algorytmów generowania kluczy rundowych PP-1 64/128 oraz PP-2 64/128 (PA = <0.9806; 0.9994>)

nr testu	rodzaj testu	ograniczenie dolne na *n*	algorytm generowania kluczy rundowych	
			PP-1	PP-2
1	częstości {frequency}	100	0.9890	0.9840
2	częstości w blokach {block-frequency (*M* = 128)}	100	0.9940	0.9910
3	serii {runs}	100	0.9940	0.9930
4	najdłuższych serii {longest-runs}	128	0.9840	0.9890
5	rzędów macierzy {rank}	38921	0.9920	0.9890
6	transformaty Fouriera {fft}	1000	0.9870	0.9870
7	nienachodzących wzorców {non-overlapping-templates (*m* = 9)}	16	pozytywny	pozytywny
8	nachodzących wzorców {overlapping-templates (*m* = 9)}	10^6	0.9860	0.9920
9	uniwersalny {universal (*M* = 7, *Q* = 1280)}	387840	0.9840	0.9810
10	złożoności liniowych {linear-complexity (*M* = 500)}	10^6	0.9910	0.9930
11.1	seryjny 1 {serial1 (*m* = 4)}	128	0.9880	0.9920
11.2	seryjny 2 {serial2 (*m* = 4)}	128	0.9920	0.9830
12	przybliżonej entropii {apen (*m* = 4)}	1024	0.9910	0.9920
13.1	kumulowanych sum (wprzód) {cumulative-sums (forward)}	100	0.9940	0.9840
13.2	kumulowanych sum (wstecz) {cumulative-sums (backward)}	100	0.9920	0.9840
14	losowych wycieczek {random-excursions}	10^6	pozytywny	pozytywny
15	wariantu losowych wycieczek {random-excursions-variant}	10^6	pozytywny	pozytywny

Przeprowadzone testy statystyczne oraz otrzymane wyniki potwierdzają jakość algorytmu generowania kluczy rundowych szyfru PP-2 64/128, który wytwarza klucze rundowe o dobrych własnościach statystycznych.

Wniosek 4.3.

W metodzie B, przy kolejnych wartościach kluczy głównych (formuła (4.10)), rozszerzony algorytm generowania kluczy rundowych w szyfrze PP-2 64/128 jest dobrym jakościowo generatorem liczb pseudolosowych.

■

B2. Algorytm generowania kluczy rundowych szyfru PP-1 64/128, jako generator liczb pseudolosowych

Innym przykładem zastosowania metody B są badania, przeprowadzone przez autora, nad rozszerzonym do 15642 iteracji, algorytmem generowania kluczy rundowych szyfru PP-1 64/128 (rys. 4.12 (a)) [6]. Dla z = 1000 sekwencji po 1001088 bitów, wszystkie testy dały wynik pozytywny (tab. 4.5).

Wyniki przeprowadzonych testów NIST SP800-22 potwierdzają jakość algorytmu generowania kluczy rundowych szyfru PP-1 64/128, wytwarzającego klucze rundowe o dobrych własnościach statystycznych.

Wniosek 4.4.

W metodzie B, przy kolejnych wartościach kluczy głównych (formuła (4.10)), rozszerzony algorytm generowania kluczy rundowych szyfru PP-1 64/128 można uznać za dobry jakościowo generator liczb pseudolosowych.

■

4.3.4. Metoda C (metapróbek)

Metoda C (metapróbek) polega na utworzeniu złożonych próbek, o długości przekraczającej 10^6 bitów, poprzez konkatenację próbek standardowych otrzymanych z różnych kluczy głównych. Dzięki takiemu podejściu, możliwe jest przeprowadzenie wszystkich 15 testów NIST SP800-22, w celu oceny algorytmów generowania kluczy rundowych, dla których niemożliwe jest zwiększenie liczby iteracji (metoda B).

Przykładem zastosowania metody C są badania przeprowadzone na złożonych próbkach (metapróbkach), o następujących długościach pojedynczej metapróbki:

- DES: 1000704 bitów, konkatenacja 1303 próbek standardowych,
- IDEA: 1000896 bitów, konkatenacja 1203 próbek standardowych,
- KASUMI: 1000448 bitów, konkatenacja 977 próbek standardowych,
- PP-1 64/128: 1001088 bitów, konkatenacja 711 próbek standardowych,

- PP-2 64/128: 1000896 bitów, konkatenacji 1203 próbek standardowych.

Metapróbki wygenerowano z wykorzystaniem losowych wartości kluczy głównych (formuła (4.9)) oraz kolejnych wartości kluczy głównych (formuła (4.10)). W wariancie z kolejnymi wartościami kluczy głównych, możliwa jest dodatkowa ocena jakości algorytmu, ponieważ pojedyncza metapróbka zawiera klucze rundowe wygenerowane z wielu kluczy głównych, nieznacznie różniących się względem siebie.

Tab. 4.6. Zestawienie wyników *p* testów NIST SP800-22 dla algorytmu generowania kluczy rundowych szyfru DES i złożonych próbek (metapróbek) o długości 1000704 bitów (PA = <0.9806; 0.9994>)

nr testu	rodzaj testu	klucze główne losowe			klucze główne kolejne		
		zestaw 1	zestaw 2	zestaw 3	zestaw 1	zestaw 2	zestaw 3
1	częstości {frequency}	0.0000	0.0000	0.0000	0.0400	0.0160	0.0340
2	częstości w blokach {block-frequency (*M* = 128)}	0.0000	0.0000	0.0000	0.8040	0.5250	0.5420
3	serii {runs}	0.0000	0.0000	0.0010	0.0060	0.0000	0.0080
4	najdłuższych serii {longest-runs}	0.1230	0.1370	0.1390	0.0000	0.0000	0.0000
5	rzędów macierzy {rank}	0.9890	0.9910	0.9900	0.0000	0.0000	0.0000
6	transformaty Fouriera {fft}	0.0000	0.0000	0.0040	0.0000	0.0000	0.0000
7	nienachodzących wzorców {non-overlapping-templates (*m* = 9)}	negatywny	negatywny	negatywny	negatywny	negatywny	negatywny
8	nachodzących wzorców {overlapping-templates (*m* = 9)}	0.0000	0.0000	0.0000	0.0000	0.0000	0.0050
9	uniwersalny {universal (*M* = 7, *Q* = 1280)}	0.0000	0.0000	0.0000	0.1000	0.1280	0.1790
10	złożoności liniowych {linear-complexity (*M* = 500)}	0.9900	0.9900	0.9890	0.9900	0.9810	0.9910
11.1	seryjny 1 {serial1 (*m* = 4)}	0.0000	0.0000	0.0110	0.0000	0.0000	0.0000
11.2	seryjny 2 {serial2 (*m* = 4)}	0.1000	0.0890	0.6380	0.0000	0.0000	0.0000
12	przybliżonej entropii {apen (*m* = 4)}	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
13.1	kumulowanych sum (wprzód) {cumulative-sums (forward)}	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
13.2	kumulowanych sum (wstecz) {cumulative-sums (backward)}	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
14	losowych wycieczek {random-excursions}	negatywny	negatywny	negatywny	negatywny	negatywny	negatywny
15	wariantu losowych wycieczek {random-excursions-variant}	negatywny	negatywny	negatywny	negatywny	negatywny	negatywny

Tabela (tab. 4.6) zawiera wyniki testów przeprowadzonych na metapróbkach, utworzonych z próbek standardowych, algorytmu generowania kluczy rundowych szyfru DES. Dla każdego wariantu kluczy głównych przeprowadzono trzy eksperymenty –

przygotowano trzy zestawy kluczy głównych i wygenerowano klucze rundowe, poddane następnie testom. Wszystkie testy, za wyjątkiem testu numer 5 rzędów macierzy (przy losowych kluczach głównych) oraz testu numer 10 złożoności liniowej (przy obu sposobach generowania kluczy głównych), dały wynik negatywny. Dla szyfrów IDEA i KASUMI, uzyskano negatywne wyniki, podobnie jak w przypadku algorytmu generowania kluczy rundowych szyfru DES i dlatego tabele tych wyników w rozprawie pominięto.

Tab. 4.7. Zestawienie wyników *p* testów NIST SP800-22 dla algorytmu generowania kluczy rundowych szyfru PP-1 64/128 i złożonych próbek (metapróbek) o długości 1001088 bitów (PA = <0.9806; 0.9994>)

nr testu	rodzaj testu	klucze główne losowe			klucze główne kolejne		
		zestaw 1	zestaw 2	zestaw 3	zestaw 1	zestaw 2	zestaw 3
1	częstości {frequency}	0.9880	0.9930	0.9900	0.0530	0.1580	0.3770
2	częstości w blokach {block-frequency (*M* = 128)}	0.9930	0.9000	0.9910	0.3650	0.8860	0.9750
3	serii {runs}	0.9840	0.9910	0.9920	0.0570	0.1340	0.0590
4	najdłuższych serii {longest-runs}	0.9910	0.9870	0.9860	0.9800	0.9450	0.9660
5	rzędów macierzy {rank}	0.9920	0.9900	0.9930	0.9900	0.9830	0.9930
6	transformaty Fouriera {fft}	0.9900	0.9800	0.9890	0.0000	0.0000	0.0000
7	nienachodzących wzorców {non-overlapping-templates (*m* = 9)}	pozytywny	pozytywny	pozytywny	negatywny	negatywny	negatywny
8	nachodzących wzorców {overlapping-templates (*m* = 9)}	0.9880	0.9880	0.9940	0.9100	0.8690	0.7480
9	uniwersalny {universal (*M* = 7, *Q* = 1280)}	0.9820	0.9840	0.9880	0.9310	0.7370	0.7990
10	złożoności liniowych {linear-complexity (*M* = 500)}	0.9900	0.9880	0.9860	0.9890	0.9890	0.9920
11.1	seryjny 1 {serial1 (*m* = 4)}	0.9890	0.9880	0.9840	0.0000	0.0000	0.0000
11.2	seryjny 2 {serial2 (*m* = 4)}	0.9880	0.9850	0.9810	0.0000	0.0000	0.0340
12	przybliżonej entropii {apen (*m* = 4)}	0.9880	0.9850	0.9880	0.0000	0.0000	0.0000
13.1	kumulowanych sum (wprzód) {cumulative-sums (forward)}	0.9870	0.9930	0.9930	0.0540	0.1520	0.3850
13.2	kumulowanych sum (wstecz) {cumulative-sums (backward)}	0.9870	0.9910	0.9940	0.0530	0.1520	0.3660
14	losowych wycieczek {random-excursions}	pozytywny	pozytywny	pozytywny	negatywny	pozytywny	pozytywny
15	wariantu losowych wycieczek {random-excursions-variant}	pozytywny	pozytywny	pozytywny	pozytywny	pozytywny	pozytywny

W tabeli (tab. 4.7) przedstawiono wyniki testów przeprowadzonych na metapróbkach, utworzonych z próbek standardowych, algorytmu generowania kluczy rundowych szyfru PP-1 64/128. Przy wariancie losowych kluczy głównych wszystkie testy zakończyły się

wynikiem pozytywnym, natomiast przy wariancie kolejnych kluczy głównych, większość testów zakończyła się wynikiem negatywnym i w wielu wypadkach, odległym znacznie od przedziału akceptacji PA.

Tab. 4.8. Zestawienie wyników *p* testów NIST SP800-22 dla algorytmu generowania kluczy rundowych szyfru PP-2 64/128 i złożonych próbek (metapróbek) o długości 1000896 bitów (PA = <0.9806; 0.9994>)

nr testu	rodzaj testu	klucze główne losowe			klucze główne kolejne		
		zestaw 1	zestaw 2	zestaw 3	zestaw 1	zestaw 2	zestaw 3
1	częstości {frequency}	0.9840	0.9870	0.9910	0.0020	0.0010	0.5350
2	częstości w blokach {block-frequency (*M* = 128)}	0.9880	0.9870	0.9930	0.3590	0.2950	1.0000
3	serii {runs}	0.9900	0.9850	0.9880	0.0020	0.0000	0.0000
4	najdłuższych serii {longest-runs}	0.9950	0.9920	0.9920	0.9470	0.9330	0.9510
5	rzędów macierzy {rank}	0.9900	0.9980	0.9960	0.0000	0.0000	0.0000
6	transformaty Fouriera {fft}	0.9880	0.9860	0.9850	0.0000	0.0000	0.0000
7	nienachodzących wzorców {non-overlapping-templates (*m* = 9)}	pozytywny	pozytywny	pozytywny	negatywny	negatywny	negatywny
8	nachodzących wzorców {overlapping-templates (*m* = 9)}	0.9860	0.9860	0.9880	0.5070	0.4950	0.7640
9	uniwersalny {universal (*M* = 7, *Q* = 1280)}	0.9870	0.9930	0.9940	0.1820	0.3480	0.2970
10	złożoności liniowych {linear-complexity (*M* = 500)}	0.9930	0.9900	0.9880	0.9880	0.9900	0.9890
11.1	seryjny 1 {serial1 (*m* = 4)}	0.9890	0.9940	0.9860	0.0000	0.0000	0.0000
11.2	seryjny 2 {serial2 (*m* = 4)}	0.9930	0.9940	0.9880	0.0020	0.0380	0.0000
12	przybliżonej entropii {apen (*m* = 4)}	0.9900	0.9840	0.9830	0.0000	0.0000	0.0000
13.1	kumulowanych sum (wprzód) {cumulative-sums (forward)}	0.9870	0.9860	0.9900	0.0020	0.0010	0.4730
13.2	kumulowanych sum (wstecz) {cumulative-sums (backward)}	0.9880	0.9870	0.9940	0.0010	0.0010	0.4840
14	losowych wycieczek {random-excursions}	pozytywny	pozytywny	pozytywny	negatywny	negatywny	negatywny
15	wariantu losowych wycieczek {random-excursions-variant}	pozytywny	pozytywny	pozytywny	negatywny	negatywny	negatywny

Wyniki testów przeprowadzonych na metapróbkach utworzonych z próbek standardowych algorytmu generowania kluczy rundowych szyfru PP-2 64/128, przedstawiono w tabeli (tab. 4.8). Podobnie jak w przypadku szyfru PP-1 64/128, przy zastosowaniu kluczy głównych losowych, wszystkie testy NIST SP800-22 zakończyły się sukcesem, a przy wariancie kolejnych kluczy głównych otrzymano wyniki negatywne (o znacząco mniejszych wartościach proporcji *p*).

Należy zwrócić uwagę, że w przypadku zastosowania w badaniach losowych kluczy głównych (formuła (4.9)), wyniki testów (dla różnych zestawów kluczy głównych) są do siebie zbliżone, w przeciwieństwie do przypadku, gdy klucze główne są kolejne (formuła (4.10)) – wtedy wyniki różnią się miedzy sobą i zależą głównie od jakości pierwszego wylosowanego klucza głównego.

Wniosek 4.5.

W metodzie C, przy losowych wartościach kluczy głównych (formuła (4.9)), algorytm generowania kluczy rundowych szyfrów PP-1 64/128 i PP-2 64/128 spełnia, a szyfrów DES, IDEA, KASUMI nie spełnia kryterium jakości (losowości), natomiast przy kolejnych wartościach kluczy głównych (formuła (4.10)) żaden z wymienionych algorytmów nie spełnia tego kryterium.

■

Metoda C umożliwia zatem przeprowadzenie wszystkich testów statystycznych NIST SP800-22, bez względu na ograniczenie konstrukcyjne liczby itearcji algorytmu generowania kluczy rundowych.

4.3.5. Metoda D (podpróbek)

Metoda D (podpróbek) polega na przeprowadzeniu wybranych testów NIST SP800-22, na skróconych próbkach, w celu oceny jakości początkowych kluczy rundowych. Przy generowaniu podpróbek zastosowano wariant losowych kluczy głównych (formuła (4.9)), a klucze rundowe ograniczono do liczby:

$$nrk_2 = \lfloor nrk \,/\, 4 \rfloor. \tag{4.11}$$

Metodę D zastosowano do następujących szyfrów:

- DES – długość podpróbki: $|rkp|$ = **192b** (nrk_2 = 4, $|rk|$ = 48b),
- IDEA – długość podpróbki: $|rkp|$ = **208b** (nrk_2 = 13, $|rk|$ = 16b),
- KASUMI – długość podpróbki: $|rkp|$ = **256b** (nrk_2 = 16, $|rk|$ = 16b),
- PP-1 64/128 – długość podpróbki: $|rkp|$ = **320b** (nrk_2 = 5, $|rk|$ = 64b),
- PP-2 64/128 – długość podpróbki: $|rkp|$ = **192b** (nrk_2 = 3, $|rk|$ = 64b),

gdzie nrk_2 oznacza liczbę kluczy rundowych skróconej próbki, $|rk|$ oznacza długość klucza rundowego, a $|rkp| = nrk_2 \cdot |rk|$.

Tab. 4.9. Zestawienie wyników *p* testów NIST SP800-22 dla algorytmów generowania kluczy rundowych szyfrów DES, IDEA, KASUMI PP-1 64/128, PP-2 64/128 i skróconych próbek (podpróbek) (PA = <0.9806; 0.9994>)

nr testu	rodzaj testu	DES	IDEA	KASUMI	PP-1	PP-2
1	częstości {frequency}	0.8150	0.9480	0.8940	0.9960	0.9910
2	częstości w blokach {block-frequency (*M* = 16)}	0.9670	0.9800	0.9740	0.9930	0.9960
3	serii {runs}	0.9540	0.9490	0.8930	0.9850	0.9910
13.1	kumulowanych sum (wprzód) {cumulative-sums (forward)}	0.8360	0.9540	0.9100	0.9950	0.9920
13.2	kumulowanych sum (wstecz) {cumulative-sums (backward)}	0.8350	0.9460	0.9040	0.9960	0.9890

W tabeli (tab. 4.9) przedstawiono wyniki wybranych testów NIST SP800-22 dla skróconych próbek (podpróbek) wygenerowanych przez algorytmy generowania kluczy rundowych szyfrów: DES, IDEA, KASUMI, PP-1 64/128 oraz PP-2 64-128. Warto zauważyć, że choć otrzymane proporcje *p* dla skróconych próbek (podpróbek) algorytmów generowania kluczy rundowych szyfrów DES, IDEA, KASUMI nie mieszczą się w przedziale akceptacji PA = <0.9806; 0.9994>, to są znacząco większe aniżeli dla próbek standardowych (metoda A, tab. 4.2). Wyniki te zatem nie pozwalają na właściwą ocenę jakości początkowych kluczy rundowych.

Rozpatrzono także złożenie czterech podpróbek, o zbliżonej długości złożenia do próbki standardowej, w celu porównania uzyskanych wyników do wyników dla próbek standardowych. Przy generowaniu złożenia podpróbek, zastosowano wariant losowych kluczy głównych (formuła (4.9)).

Rozpatrywany wariant metody D zastosowano do następujących szyfrów:

- DES – długość złożenia podpróbek: **768b**,
- IDEA – długość złożenia podpróbek: **832b**,
- KASUMI – długość złożenia podpróbek: **1024b**,
- PP-1 64/128 – długość złożenia podpróbek: **1280b**,
- PP-2 64/128 – długość złożenia podpróbek: **768b**.

Tabela (tab. 4.10) zawiera wyniki wybranych testów NIST SP800-22 dla algorytmów generowania kluczy rundowych szyfrów DES, IDEA, KASUMI, PP-1 64/128 oraz PP-2 64/128 i złożenia 4 podpróbek.

Tab. 4.10. Zestawienie wyników *p* testów NIST SP800-22 dla algorytmów generowania kluczy rundowych szyfrów DES, IDEA, KASUMI PP-1 64/128, PP-2 64/128 i złożenia 4 podpróbek (PA = <0.9806; 0.9994>)

nr testu	Rodzaj testu	DES	IDEA	KASUMI	PP-1	PP-2
1	częstości {frequency}	0.8280	0.9690	0.9210	0.9910	0.9920
2	częstości w blokach {block-frequency (*M* = 16)}	0.9410	0.9920	0.9840	0.9820	0.9860
3	serii {runs}	0.9620	0.9370	0.8490	0.9880	0.9910
13.1	kumulowanych sum (wprzód) {cumulative-sums (forward)}	0.8050	0.9690	0.9050	0.9890	0.9900
13.2	kumulowanych sum (wstecz) {cumulative-sums (backward)}	0.8130	0.9610	0.9060	0.9920	0.9900

Wniosek 4.6.

W metodzie D, przy wariancie losowych kluczy głównych (formuła (4.9)), algorytmy generowania kluczy rundowych szyfrów PP-1 64/128 oraz PP-2 64/128 spełniają kryterium jakości (losowości), a szyfrów DES, IDEA, KASUMI – nie spełniają tego kryterium.

■

4.3.6. Metoda E (hybrydowa)

Metoda E (hybrydowa) polega na przeprowadzeniu wybranych lub wszystkich testów statystycznych NIST SP800-22, dla różnych wariantów algorytmu generowania kluczy rundowych, w połączeniu z analizą skupień (taksonomią wrocławską) w celu wyznaczenia wariantu optymalnego, tj. najbliższego, w sensie odległości, hipotetycznie najlepszemu, idealnemu, wariantowi algorytmu.

Taksonomia wrocławska [55], to metoda analizy skupień (ang. *cluster analysis*), stosowana do łączenia pewnych obiektów (zmiennych) w grupy jednorodne pod względem pewnych cech (wymiarów). W ramach taksonomii wrocławskiej wyszczególnia się pojęcia:

- obiekt O_i (i = 1, 2, …, v) – jednostka badania podlegająca klasyfikacji, przedmiotem klasyfikacji jest zbiór v obiektów: $O = \{O_1, O_2, ..., O_v\}$,
- cecha Y_j (j = 1, 2, …, z) – właściwość jednostek badanego zbioru, kryterium klasyfikacji obiektów jest zbiór z cech: $Y = \{Y_1, Y_2, ..., Y_z\}$,
- podobieństwo $d_{i1,i2}$ ($i1$, $i2$ = 1, 2, …, v) – odległość między parą obiektów O_{i1} i O_{i2}, podstawą łączenia obiektów w grupy jest, przedstawiana w postaci tablicy, macierz odległości (podobieństw) D_v:

$$\boldsymbol{D_v} = \begin{bmatrix} \mathbf{0} & \boldsymbol{d_{1,2}} & \dots & \boldsymbol{d_{1,v}} \\ \boldsymbol{d_{2,1}} & \ddots & & \boldsymbol{d_{2,v}} \\ \vdots & & \ddots & \vdots \\ \boldsymbol{d_{v,1}} & \boldsymbol{d_{v,2}} & \dots & \mathbf{0} \end{bmatrix}. \tag{4.12}$$

Macierz odległości ma następujące własności:

- $d_{i1,i2} = 0$, dla $i1 = i2$,
- $d_{i1,i2} = d_{i2,i1}$,

gdzie $i1,i2 = 1, 2, \ldots, v$.

Dysponując macierzą odległości możliwe jest skonstruowanie dendrytu (lasu drzew) służącego do grupowania w skupienia obiektów jak najbardziej do siebie podobnych (algorytm 4.1).

Algorytm 4.1. Algorytm konstruowania dendrytu

1. Wyznaczenie macierzy D_v odległości taksonomicznych.
2. Ustalenie minimalnych, niezerowych odległości taksonomicznych, w ramach kolumn albo wierszy macierzy D_v, w celu wyznaczenia, dla każdego obiektu, obiektów najbliższych.
3. Przyjęcie, że każdy obiekt to wierzchołek dendrytu, a następnie połączenie krawędzią par obiektów najbliższych, wskazanych przez odległość taksonomiczną ustaloną w kroku drugim.
4. Jeżeli otrzymany dendryt jest niespójny (tj. składa się z kilku drzew – jest lasem), to należy znaleźć, dla każdego z drzew, drzewo najbliższe i połączyć je krawędzią.
5. Należy powtarzać krok 4, aż do uzyskania dendrytu spójnego.

Spójny dendryt (minimalne drzewo rozpinające) można też wyznaczyć np. algorytmem Kruskala lub Prima.

E1. Optymalizacja algorytmu generowania kluczy rundowych w szyfrze IDEA

Przykładem zastosowania metody E (hybrydowej) jest optymalizacja algorytmu generowania kluczy rundowych szyfru IDEA [5]. Jedyną operacją wykonywaną w tym algorytmie (rys. 3.47) jest rotacja w lewo o 25 bitów, $\lll$ ($a = 25$), słowa 128-bitowego. Celem optymalizacji jest wyznaczenie najlepszej wartości parametru (wariantu) rotacji a, gdzie a = 1, 2, …, 127.

Tab. 4.11. Skrócona[7] tabela wyników *p* testów NIST SP800-22 dla różnych wariantów rotacji w algorytmie generowania kluczy rundowych szyfru IDEA (PA = <0.9806; 0.9994>)

wariant rotacji *a*	rodzaj testu				
	frequency	block-frequency (*M* = 64)	cumulative-sums (forward)	cumulative-sums (backward)	runs
1	0,6910	0,8940	0,7080	0,7150	0,6100
2	0,6900	0,8900	0,7090	0,7040	**0,6220**
3	0,6900	0,9000	0,7070	0,7060	0,6030
4	0,6840	0,8990	0,7090	0,7040	0,6030
5	0,6900	0,9010	0,7200	0,7100	0,5930
6	0,6960	0,9110	**0,7220**	0,7100	0,5970
7	0,7000	0,9170	0,7180	**0,7200**	0,5930
…	…	…	…	…	…
25	0,6860	0,9150	0,7040	0,6980	0,6030
26	0,6840	0,9170	0,7080	0,7040	0,6030
27	0,6940	0,9190	0,7100	0,7120	0,5990
28	0,7000	0,9130	**0,7220**	0,7140	0,6010
29	**0,7040**	0,9130	0,7140	0,7160	0,5920
30	0,7000	0,9130	0,7100	0,7120	0,5930
…	…	…	…	…	…
35	0,6850	0,9090	0,7080	0,7010	0,5960
36	0,6850	**0,9230**	0,7000	0,6970	0,6030
37	0,6740	0,9150	0,7060	0,7000	0,6040
…	…	…	…	…	…
70	0,6960	0,9110	0,7180	0,7100	0,5970
71	0,7000	0,9170	0,7160	**0,7200**	0,6050
72	0,7020	0,9150	0,7120	0,7160	0,6020
…	…	…	…	…	…
92	0,7000	0,9130	0,7200	0,7140	0,5950
93	**0,7040**	0,9130	0,7140	0,7160	0,5910
94	0,7000	0,9130	0,7100	0,7120	0,5960
…	…	…	…	…	…
99	0,6850	0,9090	0,7060	0,7010	0,5950
100	0,6850	**0,9230**	0,7000	0,6970	0,6000
101	0,6740	0,9150	0,7040	0,7000	0,5980
…	…	…	…	…	…
112	0,6880	0,9150	0,7160	0,7100	0,5970
113	0,6960	0,9150	**0,7220**	0,7140	0,6010
114	0,6940	0,9110	0,7200	0,7180	0,6070
…	…	…	…	…	…
127	0,6910	0,8890	0,7080	0,7070	0,6010
max_i **wartość maksymalna**	**0,7040**	**0,9230**	**0,7220**	**0,7200**	**0,6220**
$\bar{x_j}$ **wartość średnia**	0,6895	0,9117	0,7084	0,7054	0,6009
s_j **odchylenie standardowe**	0,0059	0,0081	0,0057	0,0060	0,0075

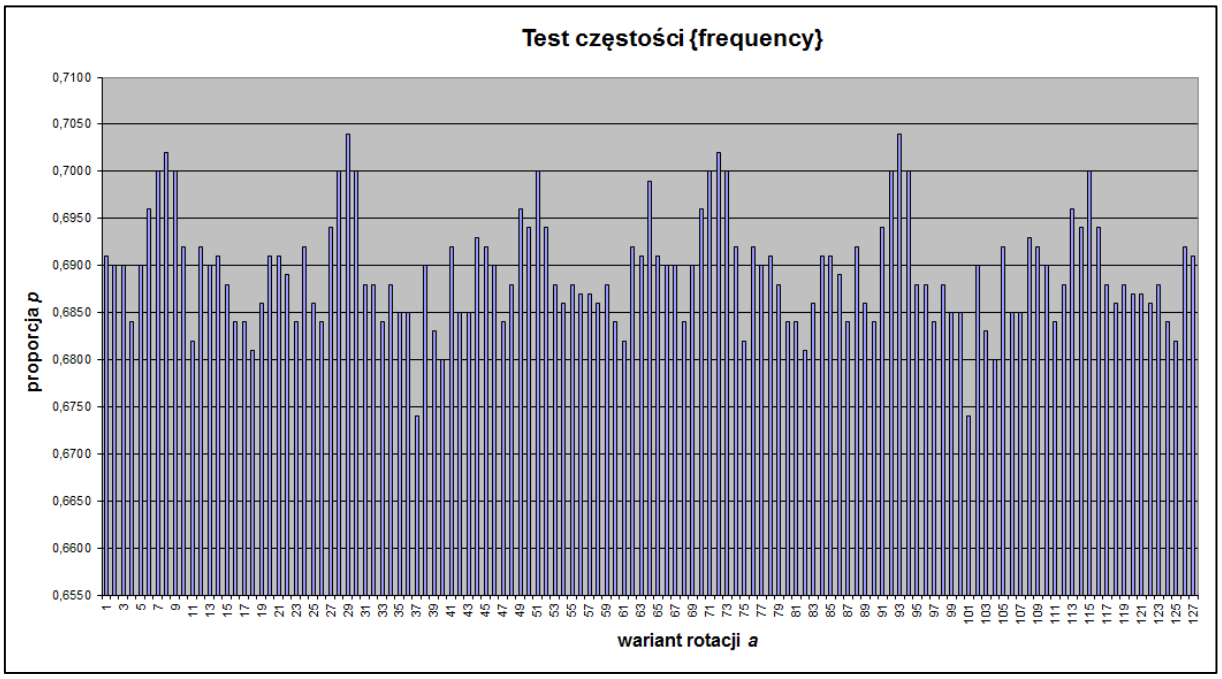
[7] Kompletna tabela wyników przeprowadzonych testów znajduje się w rozdziale Dodatki (Tab. 6.1)

W pierwszym kroku metody E, przy wariancie losowych kluczy głównych (formuła (4.9) i liczbie próbek $z = 1000$ o długości $n = 832$ bity, dla każdego wariantu rotacji, wykonano następujące testy NIST SP800-22:

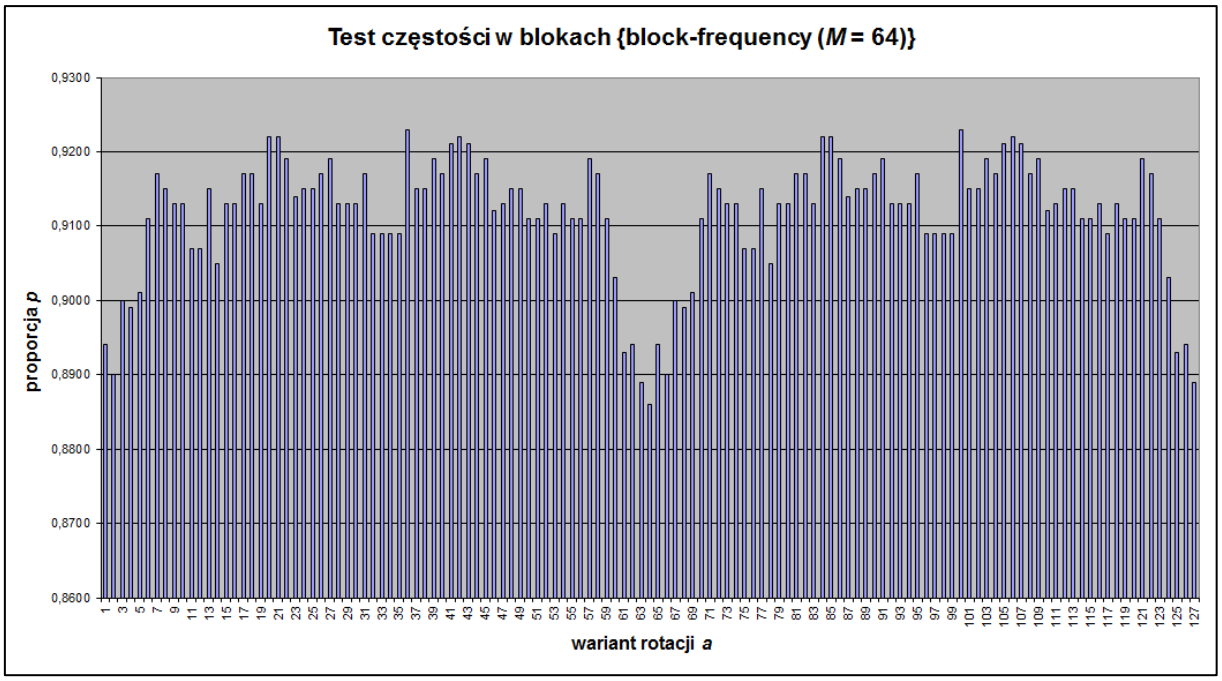
- test częstości {frequency}, $n \geq 100$,
- test częstości w blokach {block-frequency ($M = 64$)}, $n \geq 100$,
- test kumulowanych sum (wprzód) {cumulative-sums (forward)}, $n \geq 100$,
- test kumulowanych sum (wstecz) {cumulative-sums (backward)}, $n \geq 100$,
- test serii {runs}, $n \geq 100$.

Należy podkreślić, że próbki dla każdego wariantu rotacji, generowane były z tych samych kluczy głównych.

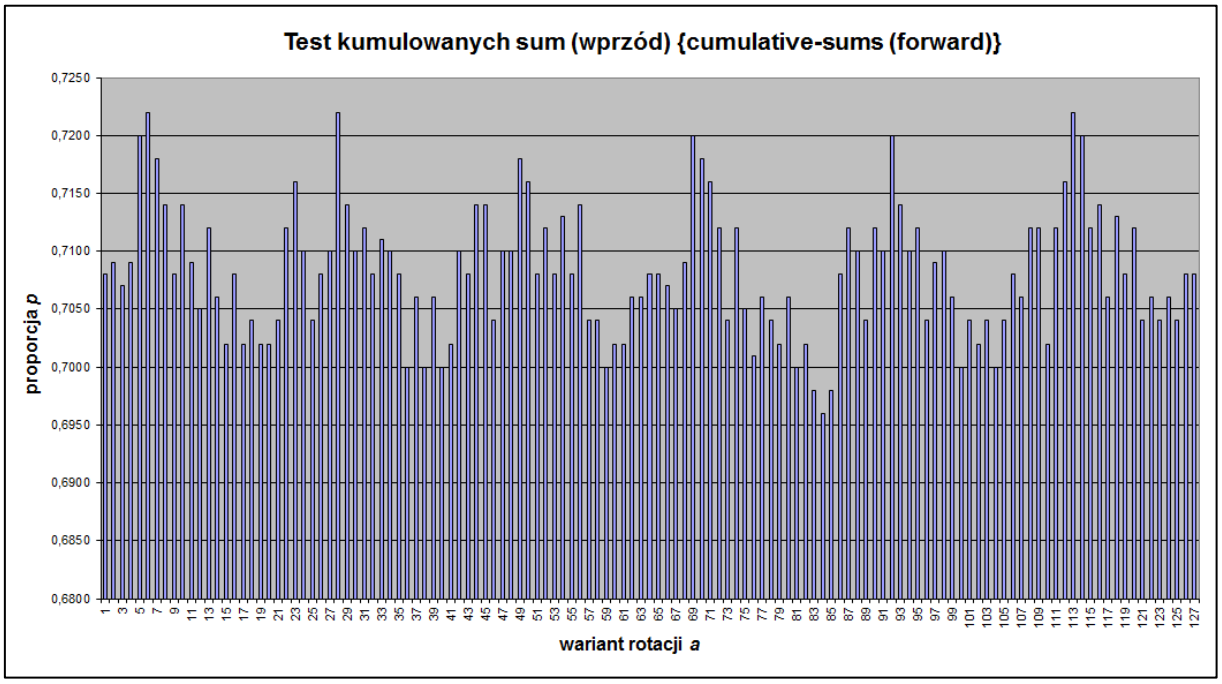
Uzyskane wyniki (proporcje) p przedstawiono w tabeli (tab. 4.11). Pogrubieniem oraz kolorem zaznaczono w tabeli najlepszy wynik otrzymany w ramach danego rodzaju testu. W trzech ostatnich wierszach tabeli zestawiono, obliczone w kolumnach, maksymalne wartości proporcji, max_j, które cechują hipotetycznie najlepszy wariant rotacji (wg. przyjętego kryterium testów NIST SP800-22), wartości średnie, $\overline{x_j}$, oraz odchylenia standardowe s_j, dla $j = 1, 2, \ldots, 5$.



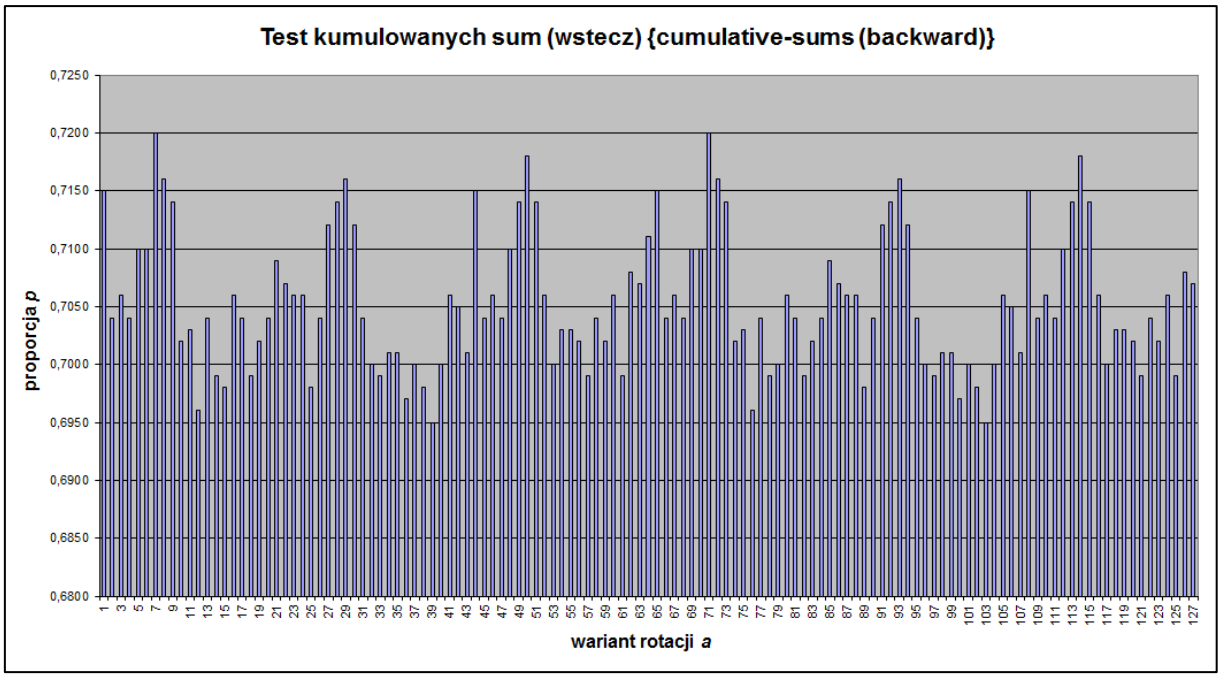
Rys. 4.13. Histogram testu częstości {frequency}



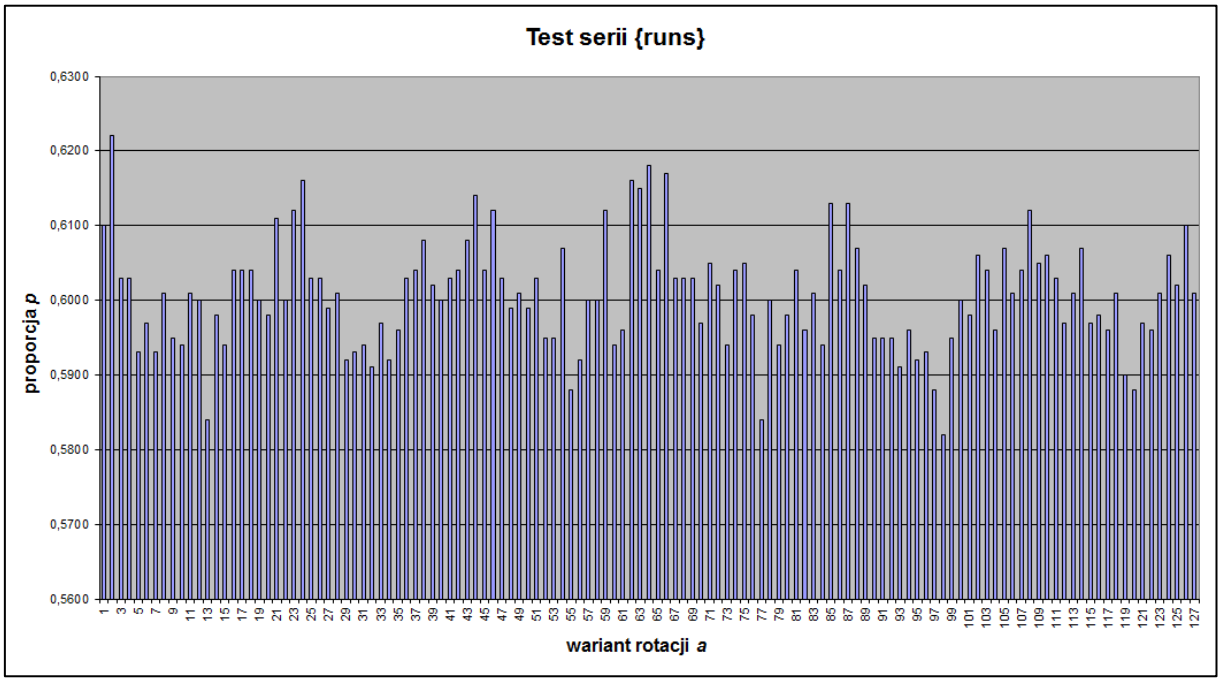
Rys. 4.14. Histogram testu częstości w blokach {block-frequency (*M* = 64)}



Rys. 4.15. Histogram testu kumulowanych sum (wprzód) {cumulative-sums (forward)}



Rys. 4.16. Histogram testu kumulowanych sum (wstecz) {cumulative-sums (backward)}



Rys. 4.17. Histogram testu serii {runs}

Na rysunkach rys. 4.13 – rys. 4.17 przedstawiono histogramy każdego rodzaju testu. Maksymalne wartości proporcji p_{max} (max_j) i odpowiadające im warianty rotacji a, dla poszczególnych testów są następujące:

- test częstości: $p_{max} = 0{,}740$ dla $a = 29, 93$;

- test częstości w blokach: $p_{max} = 0.9230$ dla $a = 36, 100$;
- test kumulowanych sum (wprzód): $p_{max} = 0.7220$ dla $a = 6, 28, 113$;
- test kumulowanych sum (wstecz): $p_{max} = 0.7200$ dla $a = 7$ i 71;
- test serii: $p_{max} = 0.6220$ dla $a = 2$.

Tab. 4.12. Skrócona[8] tabela standaryzowanych wartości cech badanych obiektów dla algorytmu generowania kluczy rundowych szyfru IDEA

obiekty reprezentujące warianty rotacji	cechy obiektów				
	Y_1	Y_2	Y_3	Y_4	Y_5
O_1	0,2567	-2,1772	-0,0689	1,6047	1,2216
O_2	0,0882	-2,6694	0,1074	-0,2397	2,8281
O_3	0,0882	-1,4390	-0,2452	0,0956	0,2845
O_4	-0,9229	-1,5620	0,1074	-0,2397	0,2845
O_5	0,0882	-1,3159	2,0466	0,7663	-1,0543
O_6	1,0993	-0,0856	2,3992	0,7663	-0,5188
O_7	1,7734	0,6527	1,6940	2,4430	-1,0543
…	…	…	…	…	…
O_{25}	-0,5859	0,4066	-0,7740	-1,2457	0,2845
O_{26}	-0,9229	0,6527	-0,0689	-0,2397	0,2845
O_{27}	0,7623	0,8988	0,2837	1,1017	-0,2510
O_{28}	1,7734	0,1605	2,3992	1,4370	0,0167
O_{29}	2,4475	0,1605	0,9889	1,7723	-1,1881
O_{30}	1,7734	0,1605	0,2837	1,1017	-1,0543
…	…	…	…	…	…
O_{36}	-0,7544	1,3909	-1,4792	-1,4134	0,2845
O_{37}	-2,6081	0,4066	-0,4214	-0,9104	0,4184
…	…	…	…	…	…
O_{70}	1,0993	-0,0856	1,6940	0,7663	-0,5188
O_{71}	1,7734	0,6527	1,3415	2,4430	0,5522
O_{72}	2,1105	0,4066	0,6363	1,7723	0,1506
…	…	…	…	…	…
O_{92}	1,7734	0,1605	2,0466	1,4370	-0,7865
O_{93}	2,4475	0,1605	0,9889	1,7723	-1,3220
O_{94}	1,7734	0,1605	0,2837	1,1017	-0,6526
…	…	…	…	…	…
O_{99}	-0,7544	-0,3316	-0,4214	-0,7427	-0,7865
O_{100}	-0,7544	1,3909	-1,4792	-1,4134	-0,1171
O_{101}	-2,6081	0,4066	-0,7740	-0,9104	-0,3849
…	…	…	…	…	…
O_{112}	-0,2488	0,4066	1,3415	0,7663	-0,5188
O_{113}	1,0993	0,4066	2,3992	1,4370	0,0167
O_{114}	0,7623	-0,0856	2,0466	2,1077	0,8200
…	…	…	…	…	…
O_{127}	0,2567	-2,7924	-0,0689	0,2633	0,0167
$O_{128(hip)}$ (hipotetycznie najlepszy obiekt)	**2,4475**	**1,3909**	**2,3992**	**2,4430**	**2,8281**

[8] Kompletna tabela wartości standaryzowanych znajduje się w rozdziale Dodatki (Tab. 6.2)

W drugim kroku metody E otrzymane w kroku pierwszym wyniki (proporcje), zebrane w tabeli (tab. 4.11), ustandaryzowano i przedstawiono w tabeli standaryzowanych wartości cech badanych obiektów dla algorytmu IDEA (tab. 4.12). Wariant rotacji i reprezentowany jest przez obiekt O_i, dla i = 1, 2, …, 127. Hipotetycznie najlepszy wariant rotacji, o maksymalnych proporcjach, reprezentowany jest przez obiekt $O_{128(hip)}$. Rodzaj testu j reprezentowany jest przez cechę Y_j, dla j = 1, 2, …, 5.

Niech v = 128 oznacza liczbę obiektów, a z = 5 oznacza liczbę cech. Wartość maksymalna max_j, przedstawiona w tabeli (tab. 4.11), obliczona jest jak następuje:

$$max_j = \max_{1 \le i \le v}(x_{i,j}), \qquad \text{dla } j = 1, 2, \ldots, z, \tag{4.13}$$

gdzie $x_{i,j}$ jest wartością proporcji dla wariantu rotacji i oraz rodzaju testu j.

Wartość średnia $\bar{x_j}$ (tab. 4.11), obliczona jest zgodnie z formułą:

$$\bar{x_j} = \frac{1}{v}\sum_{i=1}^{v} x_{i,j}\ , \qquad \text{dla } j = 1, 2, \ldots, z. \tag{4.14}$$

Odchylenie standardowe s_j (tab. 4.11), obliczone jest w następujący sposób:

$$s_j = \sqrt{\frac{1}{v-1}\sum_{i=1}^{v}\left(x_{i,j} - \bar{x_j}\right)^2}, \qquad \text{dla } j = 1, 2, \ldots, z. \tag{4.15}$$

Standaryzowane wartości cech, $y_{i,j}$, przedstawione w tabeli (tab 4.7), obliczono na podstawie wyniku (proporcji) testu $x_{i,j}$, średniej $\bar{x_j}$ oraz odchylenia standardowego s_j, następująco:

$$y_{i,j} = \frac{(x_{i,j} - \bar{x_j})}{s_j}\ , \qquad \text{dla } i = 1, 2, \ldots, v \text{ oraz } j = 1, 2, \ldots, z. \tag{4.16}$$

Tab. 4.13. Skrócona macierz (tablica) odległości D_v dla 128 wariantów algorytmu generowania kluczy rundowych szyfru IDEA

	O_1	O_2	…	O_{70}	O_{71}	O_{72}	…	O_{126}	O_{127}	$O_{128(hip)}$
O_1	0	4,2879	…	7,2759	7,2646	6,3814	…	1,3422	3,1614	10,6718
O_2	4,2879	0	…	9,5345	11,1999	10,3167	…	3,2827	3,7823	11,3941
O_3	3,5292	4,4620	…	5,7777	7,9786	6,5599	…	2,5240	**2,1337**	12,7246
…	…	…	…	…	…	…	…	…	…	…
O_{70}	7,2759	9,5345	…	0	4,5126	4,2364	…	6,6044	6,3509	8,5534
O_{71}	7,2646	11,1999	…	4,5126	0	2,3606	…	8,2698	9,0873	**4,7459**
O_{72}	6,3814	10,3167	…	4,2364	2,3606	0	…	7,3866	7,4008	6,4325
…	…	…	…	…	…	…	…	…	…	…
O_{126}	1,3422	3,2827	…	6,6044	8,2698	7,3866	…	0	2,1563	11,6770
O_{127}	3,1614	3,7823	…	6,3509	9,0873	7,4008	…	2,1563	0	13,8333
$O_{128(hip)}$	10,6718	11,3941	…	8,5534	4,7459	6,4325	…	11,6770	13,8333	0
min_{i2}	$d_{65,1}$ = **0,8033**	$d_{66,2}$ = **1,0220**	…	$d_{6,70}$ = **0,7052**	d_{771} = **1,9591**	$d_{8,72}$ = **0,4865**	…	$d_{62,126}$ = **1,1558**	$d_{3,127}$ = **2,1337**	$d_{71,128}$ = **4,7459**

W trzecim kroku proponowanej metody E (hybrydowej) wyznacza się grupy obiektów najbardziej podobnych do siebie (w tym najbardziej podobnych do hipotetycznie najlepszego obiektu). W tym celu, zgodnie z algorytmem (algorytm 4.1), wyznaczono najpierw, na podstawie tabeli (tab. 4.12), macierz odległości D_v, reprezentowaną w postaci tablicy (tab. 4.13). Zastosowaną w badaniach miarą podobieństwa jest tzw. odległość taksówkowa Manhattan:

$$d_{i1,i2} = \sum_{j=1}^{z} \left| y_{i1,j} - y_{i2,j} \right|, \qquad \text{dla } i1, i2 = 1, 2, \ldots, v, \tag{4.17}$$

gdzie:

$y_{i1,j}$ – standaryzowana wartość cechy Y_j w obiekcie O_{i1},

$y_{i2,j}$ – standaryzowana wartość cechy Y_j w obiekcie O_{i2}.

W rozważanym przypadku liczba obiektów $v = 128$, a liczba cech $z = 5$.

W drugim kroku algorytmu (algorytm 4.1) ustala się minimalne, niezerowe wartości min_{i2}, przedstawione w tabeli (tab. 4.13), obliczone następująco:

$$min_{i2} = \min_{1 \le i1 \le v} (d_{i1,i2} \neq 0), \qquad \text{dla } i2 = 1, 2, \ldots, v, \tag{4.18}$$

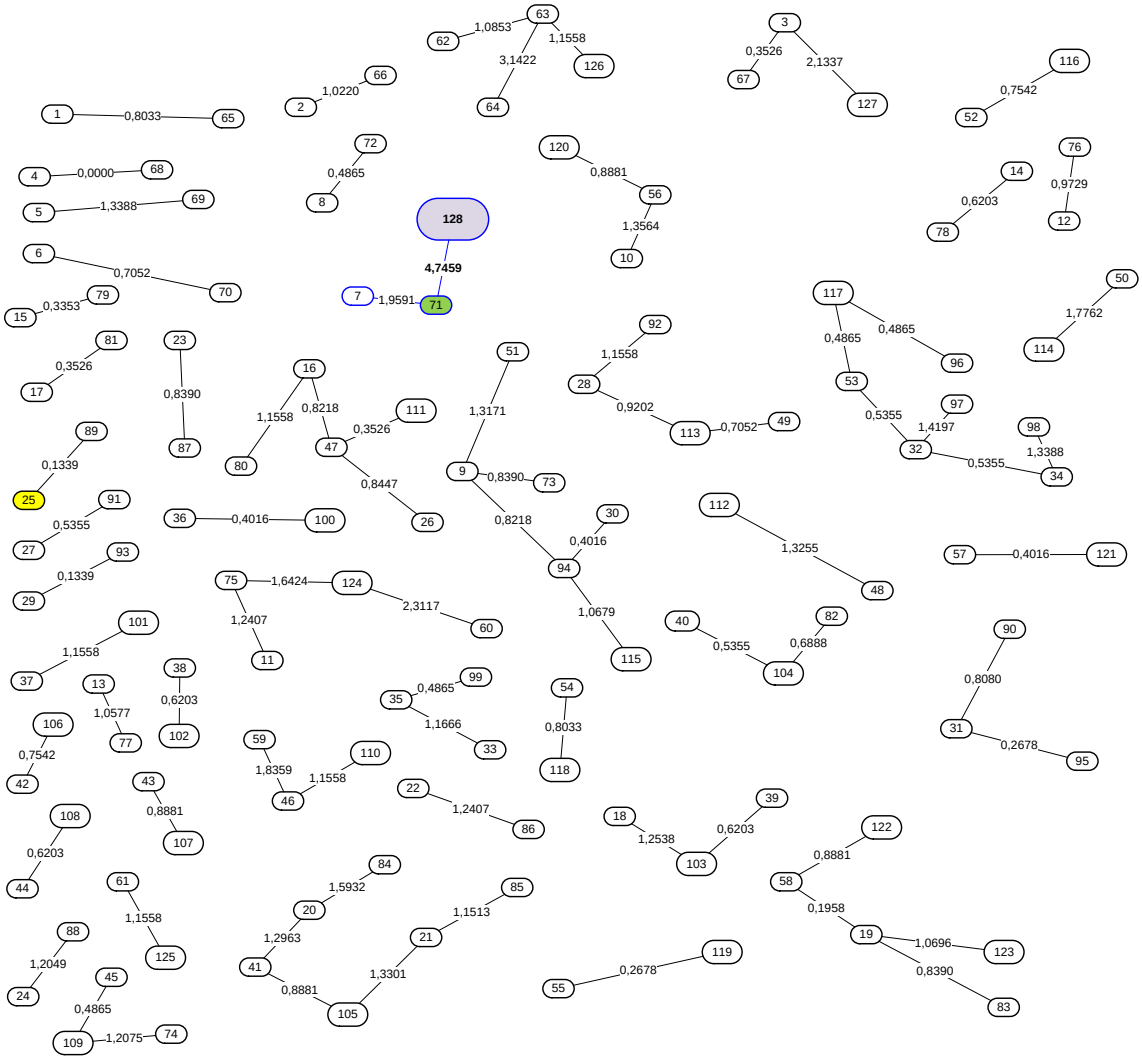
gdzie $d_{i1,i2} = D_v[i1, i2]$.

Przy obliczaniu wartości min_{i2} należy również zapamiętać odpowiadające jej indeksy $i1$. W tabeli (tab. 4.13) użyto dla wyrazistości notację $d_{i1,i2} = min_{i2}$ wskazując jeden indeks $i1$, z wielu możliwych.

W trzecim kroku algorytmu (algorytm 4.1) tworzony jest tzw. dendryt wrocławski 1-stopnia skupień jednorodnych. Obiekty stanowią wierzchołki dendrytu. Pary obiektów najbliższych łączone są krawędzią etykietowaną minimalną odległością.

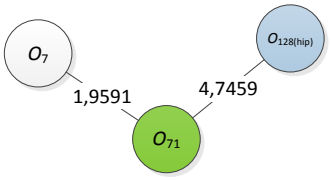
Na rysunku (rys. 4.18) przedstawiono dendryt dla 128 wariantów rotacji w algorytmie generowania kluczy rundowych szyfru IDEA, utworzony na podstawie macierzy odległości D_v (tab. 4.13). Dla uproszczenia, w wierzchołkach dendrytu zamiast obiektu O_{i2} zapisano jego indeks $i2$ ($i2 = 1, 2, \ldots, 128$). Krawędzie etykietowane są wartością min_{i2} i łączą obiekt O_{i2} z obiektami O_{i1} takimi, że $d_{i1,i2} = min_{i2}$. Uzyskany dendryt jest lasem (drzew, skupień). Wierzchołek o etykiecie 128 odpowiada hipotetycznie najlepszemu obiektowi $O_{128(hip)}$.

Kroki czwarty i piąty algorytmu (algorytm 4.1) nie są w metodzie E wykorzystywane.



Rys. 4.18. Dendryt wrocławski 1-stopnia skupień jednorodnych dla 128 wariantów algorytmu generowania kluczy rundowych szyfru IDEA

W czwartym i zarazem ostatnim kroku metody E, wybierany jest optymalny wariant algorytmu generowania kluczy rundowych. Na rysunku (rys. 4.19) przedstawiono skupienie jednorodne (drzewo), zawierające wierzchołek odpowiadający hipotetycznie najlepszemu obiektowi $O_{128(hip)}$. Skupienie to wybrane zostało z dendrytu (lasu) przedstawionego na rysunku (rys. 4.18). Obiekt O_{71} jest obiektem optymalnym, ponieważ jest najbliższy obiektowi $O_{128(hip)}$.



Rys. 4.19. Skupienie zawierające hipotetycznie najlepszy obiekt

Wniosek 4.7.

W metodzie E, przy wariancie losowych kluczy głównych (formuła (4.9)), optymalnym wariantem algorytmu generowania kluczy rundowych w szyfrze IDEA, jest wariant z operacją rotacji w lewo o 71 bitów.

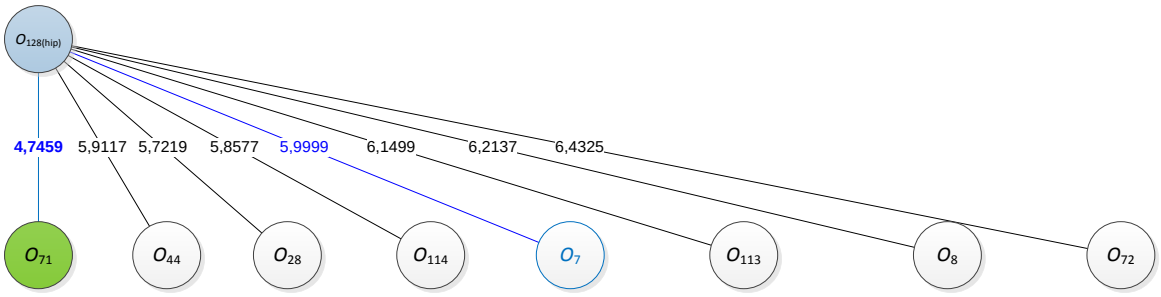
■

W tabeli (tab. 4.14) przedstawiono, pobrane z tabeli (tab. 4.11), wyniki testów NIST SP800-22 dla wariantu oryginalnego i optymalnego. Dla wszystkich testów, w wariancie optymalnym uzyskano większą wartość proporcji niż w wariancie oryginalnym.

Tab. 4.14. Porównanie wyników testów NIST SP800-22 dla rotacji oryginalnej o 25 bitów i optymalnej o 71 bitów (PA = <0.9806; 0.9994>)

wariant rotacji a	rodzaj testu				
	frequency	block-frequency (M = 64)	cumulative-sums (forward)	cumulative-sums (backward)	runs
25	0,6860	0,9150	0,7040	0,6980	0,6030
71	0,7000	0,9170	0,7160	**0,7200**	0,6050
max_j **wartość maksymalna**	**0,7040**	**0,9230**	**0,7220**	**0,7200**	**0,6220**

Rysunek (rys. 4.20) przedstawia 8 obiektów najbliższych obiektowi $O_{128(hip)}$. Rotacja o 71 bitów jest wariantem najlepszym. Kolejnymi jakościowo rotacjami są rotacje o 28, 114, 44, 7, 113, 8, 72 bity. Brak wśród nich oryginalnej rotacji o 25 bitów, stosowanej w algorytmie generowania kluczy rundowych w szyfrze IDEA.



Rys. 4.20. Osiem obiektów najbliższych obiektowi $O_{128(hip)}$

E2. Optymalizacja algorytmu generowania kluczy rundowych w szyfrze PP-1 64/64, z uwzględnieniem czasu obliczeń

Innym przykładem wykorzystania metody E (hybrydowej) jest zastosowanie jej do wyznaczenia optymalnego wariantu algorytmu generowania kluczy rundowych w szyfrze PP-1 64/64, z uwzględnieniem czasu obliczeń [4]. Spośród oryginalnego i 10 rozważanych w punkcie 4.3.2. wersji modyfikacji tego algorytmu, wybrano warianty o pozytywnych

wynikach wszystkich wykonanych testów statystycznych NIST SP800-22 (tab. 4.3). Celem optymalizacji jest wyznaczenie najlepszego z tych wariantów. Zatem w analizie skupień rozważać będziemy 6 wariantów algorytmu – oryginalny, wersja 1 bez rotacji, wersja 2 z operacjami XOR, wersja 3.3 z sześcioma S-blokami, wersja 5 złożona z pozytywnych modyfikacji oraz hipotetycznie najlepszy, z uwzględnieniem 10 cech – dziewięć cech to rodzaje poszczególnych testów statystycznych, natomiast dziesiątą cechą jest czas generowania kluczy rundowych (czas obliczeń).

Tab. 4.15. Wyniki testów NIST SP800-22 dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64 spełniających kryterium losowości, z uwzględnieniem czasu obliczeń (PA = <0.9806; 0.9994>)

wariant algorytmu	rodzaj testu									
	frequency	block-frequency (M = 4)	cumulative-sums (forward)	cumulative-sums (backward)	runs	fft	apen (m = 4)	serial1 (m = 4)	serial2 (m = 4)	czas obliczeń [s]
oryginalny	0,9890	0,9970	0,9910	0,9900	0,9860	**0,9890**	0,9940	0,9910	0,9890	2,091
bez rotacji	0,9890	0,9880	0,9910	0,9900	0,9840	0,9850	**0,9960**	**0,9960**	**0,9950**	2,057
tylko z operacjami XOR	0,9890	0,9940	0,9930	0,9880	0,9880	0,9880	0,9920	0,9950	0,9800	2,075
z sześcioma S-blokami	0,9960	**0,9990**	0,9960	**0,9980**	**0,9950**	0,9860	0,9910	0,9890	0,9880	2,074
z pozytywnymi modyfikacjami	**0,9970**	0,9960	**0,9990**	0,9970	0,9880	0,9840	0,9900	0,9900	0,9920	**2,028**
max_j / min_j wartość najlepsza	**0,9970**	**0,9990**	**0,9990**	**0,9980**	**0,9950**	**0,9890**	**0,9960**	**0,9960**	**0,9950**	**2,028**
$\bar{x}_j$ wartość średnia	0,9928	0,9955	0,9948	0,9935	0,9893	0,9868	0,9932	0,9928	0,9898	2,0588
s_j odchylenie standardowe	0,0042	0,0041	0,0037	0,0046	0,0046	0,0021	0,0026	0,0032	0,0056	0,0262

W pierwszym kroku metody E, przy wariancie losowych kluczy głównych (formuła (4.9) i liczbie próbek z = 1000 o długości n = 1408 bitów, dla każdego z pięciu wariantów algorytmu, wykonano następujące testy NIST SP800-22:

- test częstości {frequency}, $n \geq 100$,
- test częstości w blokach {block-frequency (M = 4)}, $n \geq 100$,
- test kumulowanych sum (wprzód) {cumulative-sums (forward)}, $n \geq 100$,
- test kumulowanych sum (wstecz) {cumulative-sums (backward)}, $n \geq 100$,
- test serii {runs}, $n \geq 100$,

- test transformaty Fouriera {fft}, $n \geq 1000$,
- test przybliżonej entropi {apen ($m = 4$)}, $n \geq 1024$,
- test seryjny 1 {serial1 ($m = 4$)}, $n \geq 128$,
- test seryjny 2 {serial2 ($m = 4$)}, $n \geq 128$.

Należy zaznaczyć, że próbki dla każdego wariantu algorytmu, generowane były z tych samych kluczy głównych. W rozważanym przypadku, wyniki testów i czas obliczeń pobrano z punktu 4.3.2 i przedstawiono w tabeli (tab. 4.15). Pogrubieniem oraz kolorem zaznaczono w tabeli wartość najlepszą, tj. maksymalny wynik otrzymany w ramach danego rodzaju testu i minimalny czas obliczeń. W trzech ostatnich wierszach tabeli przedstawiono, obliczone w kolumnach, wartości najlepsze, tj. maksymalne max_j ($j = 1, 2, \ldots, 9$) i minimalną min_j ($j = 10$), cechujące hipotetycznie najlepszy wariant, wartości średnie $\overline{x_j}$ oraz odchylenia standardowe s_j ($j = 1, 2, \ldots, 10$).

Tab. 4.16. Standaryzowane wartości cech badanych obiektów dla algorytmu generowania kluczy rundowych szyfru PP-1 64/64, z uwzględnieniem czasu obliczeń

obiekty reprezentujące wariant algorytmu	cechy obiektów									
	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}
O_1	-0,9094	0,3627	-1,0331	-0,7548	-0,7194	1,0139	0,3252	-0,5750	-0,1479	1,2280
O_2	-0,9094	-1,8137	-1,0331	-0,7548	-1,1511	-0,8579	1,1057	0,9931	0,9167	-0,0700
O_3	-0,9094	-0,3627	-0,4941	-1,1862	-0,2878	0,5459	-0,4553	0,6795	-1,7447	0,6172
O_4	0,7513	0,8464	0,3144	0,9705	1,2231	-0,3900	-0,8455	-1,2022	-0,3253	0,5790
O_5	0,9885	0,1209	1,1230	0,7548	-0,2878	-1,3259	-1,2357	-0,8886	0,3844	-1,1771
$O_{6(hip)}$ (hipotetycznie najlepszy obiekt)	**0,9885**	**0,8464**	**1,1230**	**0,9705**	**1,2231**	**1,0139**	**1,1057**	**0,9931**	**0,9167**	**-1,1771**

W drugim kroku metody E tworzona jest, na podstawie tabeli (tab. 4.15), tabela standaryzowanych wartości cech badanych obiektów dla algorytmu generowania kluczy rundowych szyfru PP-1 64/64 (tab. 4.16). Wariant algorytmu i reprezentowany jest przez obiekt O_i, dla $i = 1, 2, \ldots, 5$. Hipotetycznie najlepszy wariant, o maksymalnych wynikach testów i minimalnym czasie obliczeń, reprezentowany jest przez obiekt $O_{6(hip)}$. Rodzaj testu j reprezentowany jest przez cechę Y_j, dla $j = 1, 2, \ldots, 9$. Czas generowania kluczy rundowych reprezentowany jest przez cechę Y_{10}.

Niech $v = 6$ oznacza liczbę obiektów, a $z = 10$ oznacza liczbę cech. Wartość najlepsza, w tabeli (tab. 4.15), dla $j = 1, 2, \ldots, z–1$, to wartość maksymalna max_j, obliczona zgodnie z formułą (4.13).

Wartość najlepsza dla $j = z$, obliczona jest jak następuje:

$$min_j = \min_{1 \le i \le v}(x_{i,j}), \tag{4.19}$$

gdzie $x_{i,j}$ jest wartością czasu obliczeń ($j = 10$), dla wariantu algorytmu i.

Wartość średnia $\overline{x_j}$ (tab. 4.15), obliczona jest zgodnie z formułą (4.14), a odchylenie standardowe s_j (tab. 4.15), obliczone jest zgodnie z formułą (4.15). Standaryzowane wartości cech, $y_{i,j}$, w tabeli (tab. 4.16), obliczono na podstawie wyniku testu lub wartości czasu obliczeń $x_{i,j}$, średniej $\overline{x_j}$ oraz odchylenia standardowego s_j, zgodnie z formułą (4.16), dla $i = 1, 2, \ldots, v = 6$ oraz $j = 1, 2, \ldots, z = 10$.

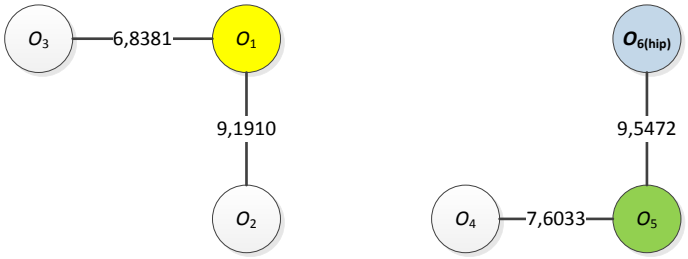
Tab. 4.17. Macierz (tablica) odległości D_v dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64, z uwzględnieniem czasu obliczeń

	O_1	O_2	O_3	O_4	O_5	$O_{6(hip)}$
O_1	0	**9,1910**	**6,8381**	11,1880	13,3889	14,0238
O_2	9,1910	0	9,9116	16,2733	14,6922	13,7926
O_3	**6,8381**	9,9116	0	12,0113	14,0834	15,1898
O_4	11,1880	16,2733	12,0113	0	**7,6033**	9,5943
O_5	13,3889	14,6922	14,0834	**7,6033**	0	**9,5472**
$O_{6(hip)}$	14,0238	13,7926	15,1898	9,5943	9,5472	0
min_{i2}	$d_{3,1}$ = **6,8381**	$d_{1,2}$ = **9,1910**	$d_{1,3}$ = **6,8381**	$d_{5,4}$ = **7,6033**	$d_{4,5}$ = **7,6033**	$d_{5,6(hip)}$ = **9,5472**

W trzecim kroku metody E wyznacza się, zgodnie z algorytmem konstruowania dendrytu (algorytm 4.1), grupy obiektów najbardziej do siebie podobnych. W pierwszym kroku algorytmu obliczana jest, na podstawie tabeli (tab. 4.16), macierz odległości (podobieństw) D_v, reprezentowana w postaci tablicy (tab. 4.17). Miarą podobieństwa jest tzw. odległość taksówkowa Manhattan, określona zgodnie z formułą (4.17), dla liczby obiektów $z = 6$ i liczby cech $v = 10$.

W drugim kroku algorytmu (algorytm 4.1) oblicza się minimalne, niezerowe wartości min_{i2} (tab. 4.17), zgodnie z formułą (4.19). Przy obliczaniu wartości min_{i2} należy zapamiętać odpowiadające jej indeksy $i1$. W tabeli (tab. 4.17) użyto notacji $d_{i1,i2} = min_{i2}$, wskazując jeden indeks $i1$, z wielu możliwych.

W trzecim kroku algorytmu (algorytm 4.1), na podstawie tabeli (tab. 4.17), tworzony jest dendryt wrocławski 1-stopnia skupień jednorodnych, przedstawiony na rysunku (rys. 4.21). Wierzchołki dendrytu odpowiadają obiektom O_{i2} ($i2 = 1, 2, \ldots, v$). Krawędzie dendrytu etykietowane są wartością min_{i2} i łączą obiekt O_{i2} z obiektami O_{i1}, takimi że $d_{i1,i2} = min_{i2}$. Uzyskany dendryt jest lasem, złożonym z dwóch drzew (skupień). Kroki czwarty i piąty algorytmu (algorytm 4.1) nie są w metodzie E wykorzystywane.



Rys. 4.21. Dendryt wrocławski 1-stopnia skupień jednorodnych dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64, spełniających kryterium losowości, z uwzględnieniem czasu obliczeń

W czwartym (ostatnim) kroku metody E wybierany jest optymalny wariant algorytmu generowania kluczy rundowych w szyfrze PP-1, z uwzględnieniem czasu obliczeń.

Na rysunku (rys. 4.21) istotne dla optymalnego rozwiązania jest skupienie jednorodne (drzewo), zawierające wierzchołek odpowiadający hipotetycznie najlepszemu obiektowi $O_{6(hip)}$. Obiekt O_5 jest obiektem optymalnym, ponieważ jest on najbliższy obiektowi $O_{6(hip)}$.

Wniosek 4.8.

W metodzie E, przy wariancie losowych kluczy głównych (formuła (4.9)), optymalnym wariantem algorytmu generowania kluczy rundowych w szyfrze PP-1 64/64, z uwzględnieniem czasu obliczeń, jest wariant złożony z pozytywnych modyfikacjami, tj. bez rotacji *RR*, z operacjami XOR zamiast dodawania i odejmowania modulo 256 oraz z liczbą S-bloków zredukowaną do sześciu.

■

Tab. 4.18. Porównanie wyników testów NIST SP800-22 dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64 oryginalnego i optymalnego, z uwzględnieniem czasu obliczeń (PA = <0.9806; 0.9994>)

wariant algorytmu	rodzaj testu									czas obliczeń [s]
	frequency	block-frequency ($M = 4$)	cumulative-sums (forward)	cumulative-sums (backward)	runs	fft	apen ($m = 4$)	serial1 ($m = 4$)	serial2 ($m = 4$)	
oryginalny	0,9890	0,9970	0,9910	0,9900	0,9860	**0,9890**	0,9940	0,9910	0,9890	2,091
z pozytywnymi modyfikacjami	**0,9970**	0,9960	**0,9990**	0,9970	0,9880	0,9840	0,9900	0,9900	0,9920	**2,028**
max_j / min_j wartość najlepsza	**0,9970**	**0,9990**	**0,9990**	**0,9980**	**0,9950**	**0,9890**	**0,9960**	**0,9960**	**0,9950**	**2,028**

W tabeli (tab. 4.18) przedstawiono wyniki testów NIST SP800-22 oraz czas obliczeń, dla wariantu oryginalnego i optymalnego algorytmu generowania kluczy rundowych szyfru PP-1 64/64, pobrane z tabeli (tab. 4.15). Dla pięciu testów z dziewięciu, w wariancie optymalnym uzyskano większą wartość proporcji niż w wariancie oryginalnym. Czas generowania kluczy rundowych w wariancie optymalnym jest krótszy.

E3. Optymalizacja algorytmu generowania kluczy rundowych w szyfrze PP-1 64/64, z pominięciem czasu obliczeń

Trzeci przykład wykorzystania metody E (hybrydowej) to zastosowanie jej do wyznaczenia optymalnego wariantu algorytmu generowania kluczy rundowych w szyfrze PP-1 64/64, z pominięciem czasu obliczeń. Podobnie jak w poprzednim przykładzie, spośród oryginalnego i 10 rozważanych (punkt 4.3.2.) wersji modyfikacji tego algorytmu wybrano te warianty, które spełniają kryterium jakości jakim jest pozytywny wynik wykonanych testów NIST SP800-22 (tab. 4.3). Zatem w analizie skupień, rozważanych będzie 6 następujących wariantów algorytmu:

- oryginalny,
- wersja 1 bez rotacji,
- wersja 2 z operacjami XOR,
- wersja 3.3 z sześcioma S-blokami,
- wersja 5 złożona z pozytywnych modyfikacji,
- hipotetycznie najlepszy.

W pierwszym kroku metody E, analogicznie jak w poprzednich przykładach zastosowania opisywanej metody, w tabeli (tab. 4.19) przedstawiono, pobrane z punktu 4.3.2, wyniki testów NIST SP800-22, ale z pominięciem czasu generowania kluczy rundowych (czasu obliczeń). Pogrubieniem oraz kolorem zaznaczono w tabeli wartość najlepszą, tj. maksymalny wynik otrzymany w ramach danego rodzaju testu. W trzech ostatnich wierszach tabeli przedstawiono, obliczone w kolumnach, wartości maksymalne max_j, cechujące hipotetycznie najlepszy wariant, wartości średnie $\overline{x_j}$ oraz odchylenia standardowe s_j (j = 1, 2, …, 9).

Tab. 4.19. Wyniki testów NIST SP800-22 dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64, spełniających kryterium losowości z pominięciem czasu obliczeń (PA = <0.9806; 0.9994>)

wariant algorytmu	rodzaj testu								
	frequency	block-frequency (M = 4)	cumulative-sums (forward)	cumulative-sums (backward)	runs	fft	apen (m = 4)	serial1 (m = 4)	serial2 (m = 4)
oryginalny	0,9890	0,9970	0,9910	0,9900	0,9860	**0,9890**	0,9940	0,9910	0,9890
bez rotacji	0,9890	0,9880	0,9910	0,9900	0,9840	0,9850	**0,9960**	**0,9960**	**0,9950**
z operacjami XOR	0,9890	0,9940	0,9930	0,9880	0,9880	0,9880	0,9920	0,9950	0,9800
z sześcioma S-blokami	0,9960	**0,9990**	0,9960	**0,9980**	**0,9950**	0,9860	0,9910	0,9890	0,9880
z pozytywnymi modyfikacjami	**0,9970**	0,9960	**0,9990**	0,9970	0,9880	0,9840	0,9900	0,9900	0,9920
max_j wartość maksymalna	**0,9970**	**0,9990**	**0,9990**	**0,9980**	**0,9950**	**0,9890**	**0,9960**	**0,9960**	**0,9950**
$\bar{x}_j$ wartość średnia	0,9928	0,9955	0,9948	0,9935	0,9893	0,9868	0,9932	0,9928	0,9898
s_j odchylenie standardowe z próby	0,0042	0,0041	0,0037	0,0046	0,0046	0,0021	0,0026	0,0032	0,0056

Tab. 4.20. Standaryzowane wartości cech badanych obiektów dla algorytmu generowania kluczy rundowych szyfru PP-1 64/64, z pominięciem czasu obliczeń

obiekty reprezentujące wariant algorytmu	cechy obiektów								
	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9
O_1	-0,9094	0,3627	-1,0331	-0,7548	-0,7194	1,0139	0,3252	-0,5750	-0,1479
O_2	-0,9094	-1,8137	-1,0331	-0,7548	-1,1511	-0,8579	1,1057	0,9931	0,9167
O_3	-0,9094	-0,3627	-0,4941	-1,1862	-0,2878	0,5459	-0,4553	0,6795	-1,7447
O_4	0,7513	0,8464	0,3144	0,9705	1,2231	-0,3900	-0,8455	-1,2022	-0,3253
O_5	0,9885	0,1209	1,1230	0,7548	-0,2878	-1,3259	-1,2357	-0,8886	0,3844
$O_{6(hip)}$ (hipotetycznie najlepszy obiekt)	**0,9885**	**0,8464**	**1,1230**	**0,9705**	**1,2231**	**1,0139**	**1,1057**	**0,9931**	**0,9167**

W drugim kroku metody E, na podstawie tabeli (tab. 4.19), konstruowana jest tabela standaryzowanych wartości cech badanych obiektów dla algorytmu generowania kluczy rundowych szyfru PP-1 64/64 (tab. 4.20). Wariant algorytmu i reprezentowany jest przez obiekt O_i, dla i = 1, 2, …, 5. Hipotetycznie najlepszy wariant, o maksymalnych wynikach testów, reprezentowany jest przez obiekt $O_{6(hip)}$. Rodzaj testu j reprezentowany jest przez cechę Y_j, dla j = 1, 2, …, 9.

Niech $v = 6$ oznacza liczbę obiektów, a $z = 9$ oznacza liczbę cech. Wartość najlepsza, w tabeli (tab. 4.19), dla $j = 1, 2, \ldots, z$, to wartość maksymalna max_j, obliczona zgodnie z formułą (4.13). Wartość średnia $\overline{x_j}$ (tab. 4.19), obliczona jest zgodnie z formułą (4.14), a odchylenie standardowe s_j (tab. 4.19), obliczone jest zgodnie z formułą (4.15).

Standaryzowane wartości cech, $y_{i,j}$, w tabeli (tab. 4.20), obliczono na podstawie wyniku testu $x_{i,j}$, średniej $\overline{x_j}$ oraz odchylenia standardowego s_j, zgodnie z formułą (4.16), dla $i = 1, 2, \ldots, v = 6$ oraz $j = 1, 2, \ldots, z = 9$.

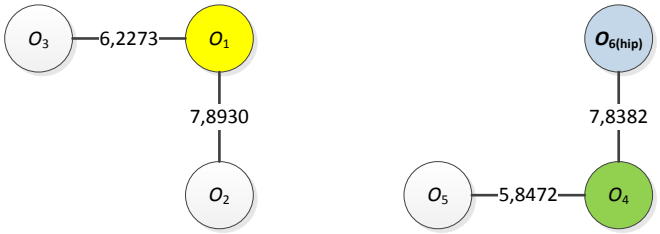
Tab. 4.21. Macierz (tablica) odległości D_v dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64, z pominięciem czasu obliczeń

	O_1	O_2	O_3	O_4	O_5	$O_{6(hip)}$
O_1	0	**7,8930**	**6,2273**	10,5390	10,9838	11,6187
O_2	7,8930	0	9,2244	15,6243	13,5851	12,6855
O_3	**6,2273**	9,2244	0	11,9732	12,2892	13,3956
O_4	10,5390	15,6243	11,9732	0	**5,8472**	**7,8382**
O_5	10,9838	13,5851	12,2892	**5,8472**	0	9,5472
$O_{6(hip)}$	11,6187	12,6855	13,3956	7,8382	9,5472	0
min_{i2}	$d_{3,1}$ = **6,2273**	$d_{1,2}$ = **7,8930**	$d_{1,3}$ = **6,2273**	$d_{5,4}$ = **5,8472**	$d_{4,5}$ = **5,8472**	$d_{4,6(hip)}$ = **7,8382**

W trzecim kroku metody E wyznacza się, zgodnie z algorytmem konstruowania dendrytu (algorytm 4.1), grupy obiektów najbardziej do siebie podobnych. W pierwszym kroku algorytmu obliczana jest, na podstawie tabeli (tab. 4.20), macierz odległości (podobieństw) D_v, reprezentowana w postaci tablicy (tab. 4.21). Miarą podobieństwa jest tzw. odległość taksówkowa Manhattan, określona zgodnie z formułą (4.17), dla liczby obiektów $z = 6$ i liczby cech $v = 9$.

W drugim kroku algorytmu (algorytm 4.1) oblicza się minimalne, niezerowe wartości min_{i2} (tab. 4.21), zgodnie z formułą (4.19). Przy obliczaniu wartości min_{i2} należy zapamiętać odpowiadające jej indeksy $i1$. W tabeli (tab. 4.21) użyto notacji $d_{i1,i2} = min_{i2}$, wskazując jeden indeks $i1$, z wielu możliwych.

W trzecim kroku algorytmu (algorytm 4.1), na podstawie tabeli (tab. 4.21), tworzony jest dendryt wrocławski 1-stopnia skupień jednorodnych, przedstawiony na rysunku (rys. 4.22). Wierzchołki dendrytu odpowiadają obiektom O_{i2} ($i2 = 1, 2, \ldots, v$). Krawędzie dendrytu etykietowane są wartością min_{i2} i łączą obiekt O_{i2} z obiektami O_{i1} takimi, że $d_{i1,i2} = min_{i2}$. Uzyskany dendryt jest lasem, złożonym z dwóch drzew (skupień). Kroki czwarty i piąty algorytmu (algorytm 4.1) nie są w metodzie E wykorzystywane.



Rys. 4.22. Dendryt wrocławski 1-stopnia skupień jednorodnych dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64, spełniających kryterium losowości, z pominięciem czasu obliczeń

W czwartym (ostatnim) kroku metody E wybierany jest optymalny wariant algorytmu generowania kluczy rundowych w szyfrze PP-1 64/64, z pominięciem czasu obliczeń tych kluczy.

Na rysunku (rys. 4.22) istotne dla optymalnego rozwiązania jest skupienie jednorodne (drzewo), zawierające wierzchołek odpowiadający hipotetycznie najlepszemu obiektowi $O_{6(hip)}$. Obiekt O_4 jest obiektem optymalnym, ponieważ jest on najbliższy obiektowi $O_{6(hip)}$.

Wniosek 4.9.

W metodzie E, przy wariancie losowych kluczy głównych (formuła (4.9)), optymalnym wariantem algorytmu generowania kluczy rundowych w szyfrze PP-1 64/64, z pominięciem czasu obliczeń, jest wariant z sześcioma S-blokami.

■

Tab. 4.22. Porównanie wyników testów NIST SP800-22 dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64, oryginalnego i optymalnego, z pominięciem czasu obliczeń (PA = <0.9806; 0.9994>)

wariant algorytmu	rodzaj testu								
	frequency	block-frequency ($M = 4$)	cumulative-sums (forward)	cumulative-sums (backward)	runs	fft	apen ($m = 4$)	serial1 ($m = 4$)	serial2 ($m = 4$)
oryginalny	0,9890	0,9970	0,9910	0,9900	0,9860	**0,9890**	0,9940	0,9910	0,9890
z sześcioma S-blokami	0,9960	**0,9990**	0,9960	**0,9980**	**0,9950**	0,9860	0,9910	0,9890	0,9880
max_j wartość maksymalna	**0,9970**	**0,9990**	**0,9990**	**0,9980**	**0,9950**	**0,9890**	**0,9960**	**0,9960**	**0,9950**

W tabeli (tab. 4.22) przedstawiono wyniki testów NIST SP800-22 dla wariantu oryginalnego i optymalnego w przypadku pominięcia czasu obliczeń, algorytmu generowania

kluczy rundowych szyfru PP-1 64/64, pobrane z tabeli (tab. 4.19). Dla czterech testów z dziewięciu, w wariancie optymalnym uzyskano większą wartość proporcji niż w wariancie oryginalnym.

Zgodnie z wnioskami – wniosek 4.8 i wniosek 4.9 – przy uwzględnieniu czasu obliczeń kluczy rundowych, najlepszym wariantem algorytmu generowania tych kluczy jest wersja 5.1 z pozytywnymi modyfikacjami (rys. 4.9), a przy pominięciu czasu obliczeń – wersja 3.3 z sześcioma S-blokami (rys. 4.8). Wersja 5.1, w porównaniu z wersją 3.3, ma dwa istotne uproszczenia: brak rotacji sterowanej danymi, *RR* oraz operacje XOR, zamiast sumy i różnicy modulo 256.

Uproszczenia te skracają czas obliczeń, ale pogarszają jakość generowania kluczy rundowych.

5. Podsumowanie

Rozprawa poświęcona jest tematyce oceny jakości algorytmów generowania kluczy rundowych w szyfrach blokowych. Rozważono teoretyczną ocenę jakości algorytmu i ocenę praktyczną, opartą na testach statystycznych NIST SP800-22, w powiązaniu z analizą skupień, w celu optymalizacji konstrukcji algorytmu. Część wyników zawartych w rozprawie opublikowano w artykułach [3][4][5][6] oraz zaprezentowano na konferencjach naukowych.

Rozdział drugi poświęcony jest podstawom kryptografii symetrycznej i asymetrycznej. Przedstawiono w nim:

- aspekty bezpieczeństwa i procesy (poufność, integralność, dostępność, niezaprzeczalność, uwierzytelnianie, kontrola dostępu),
- zapewnienie poufności z wykorzystaniem kryptografii symetrycznej (algorytm 2.1, rys. 2.1),
- szyfrowanie i deszyfrowanie z wykorzystaniem szyfrów strumieniowych (rys. 2.2),
- szyfrowanie i deszyfrowanie z wykorzystaniem szyfrów blokowych (rys. 2.3),
- tryby pracy szyfrów blokowych (ECB, CBC, CFB, OFB),
- zapewnienie poufności z wykorzystaniem kryptografii asymetrycznej (algorytm 2.2, rys. 2.4),
- uwierzytelnienie nadawcy i zapewnienie niezaprzeczalności w kryptografii asymetrycznej (algorytm 2.3, rys. 2.5),
- zastosowanie funkcji skrótu do weryfikacji integralności danych (algorytm 2.4, rys. 2.6),
- generowanie podpisu cyfrowego przez użytkownika A (algorytm 2.5, rys. 2.7),
- weryfikacja podpisu cyfrowego i integralności danych przez użytkownika B (algorytm 2.6, rys. 2.8).

W rozdziale trzecim przeanalizowano algorytmy generowania kluczy rundowych w wybranych szyfrach blokowych z podziałem ze względu na budowę szyfru:

- sieć Feistela: DES, 3DES, LOKI97, SM4, KASUMI, FeW, LCB-IoT,
- sieć SPN: Rijndael (AES), Serpent, SAFER+, PRESENT, PP-1, PP-2,
- inne sieci: IDEA, RC6, Speck.

Prezentację algorytmów generowania kluczy rundowych poprzedzono omówieniem ogólnej struktury szyfru blokowego, złożonym z następujących elementów:

- wewnętrzna struktura funkcji szyfrująca E_k i deszyfrującej D_k (rys. 3.1),

- podstawowa struktura funkcji szyfrującej *EF* i deszyfrującej *DF* w szyfrze SPN (rys. 3.2),
- warstwy: *KA* dodania klucza, *S* podstawień i *P* dyfuzji,
- podstawowa struktura funkcji szyfrującej/deszyfrującej, *EF*/*DF* i funkcji bazowej *f* w szyfrze Feistela (rys. 3.3),
- algorytm generowania kluczy rundowych dla przypadku $R = r$ (algorytm 3.1).

Opis szyfru i algorytmu generowania kluczy rundowych składa się z następujących elementów:

- informacja o szyfrze i jego podstawowych parametrach,
- struktura funkcji szyfrującej (/deszyfrującej) w formie rysunku,
- operacje funkcji szyfrującej (/deszyfrującej) w formie listy z opisem,
- struktura funkcji generowania kluczy rundowych w formie rysunku,
- operacje funkcji i algorytmu generowania kluczy rundowych w formie listy z opisem,
- algorytm generowania kluczy rundowych (wejście, wyjście, procedura),
- szczegóły operacji funkcji i algorytmu generowania kluczy rundowych (tablice operacji),
- tabela podstawowych cech szyfru z parametrem $|rkp|$ wykorzystywanym w metodzie A,
- tabela z listą operacji funkcji szyfrującej (/deszyfrującej) i algorytmu generowania kluczy rundowych,
- wniosek dotyczący operacji liniowych oraz nieliniowych algorytmu generowania kluczy rundowych i możliwości zastosowania metody B,
- analiza zależności bitu klucza rundowego od bitów klucza głównego i utworzenie schematu zastępczego algorytmu generowania kluczy rundowych dla wybranych algorytmów,
- wniosek dotyczący zależności klucza rundowego i pojedynczego bitu klucza rundowego od bitów klucza głównego,
- charakterystyka algorytmu generowania kluczy rundowych z teoretyczną oceną jego jakości.

Na podstawie zestawienia cech (tab. 3.49, tab. 3.50) dokonano analizy porównawczej algorytmu generowania kluczy rundowych rozważanych szyfrów blokowych. W szczególności przeprowadzono klasyfikację algorytmów ze względu na:

- elastyczność w doborze długości klucza,

- liczbę kluczy rundowych,
- długość klucza rundowego,
- liczbę iteracji algorytmu,
- długość przetwarzanego przez algorytm słowa,
- uzupełnienie klucza głównego do określonej długości,
- indywidualizację iteracji,
- iteracje nadmiarowe lub rozbiegowe,
- bezpośrednie użycie klucza głównego w kluczach rundowych,
- skalowalność algorytmu.

Przeanalizowano również operacje liniowe i nieliniowe stosowane w algorytmach, w odniesieniu do warstw: *KA* – dodania klucza (pomocniczego), *S* – podstawień i *P* – dyfuzji. W wyniku tej analizy uporządkowano algorytmy generowania kluczy rundowych ze względu na jakość warstwy podstawień i warstwy dyfuzji (rys. 3.55). Podzielono też algorytmy generowania kluczy rundowych na podobne (inspirowane) i odmienne od funkcji szyfrującej (/deszyfrującej).

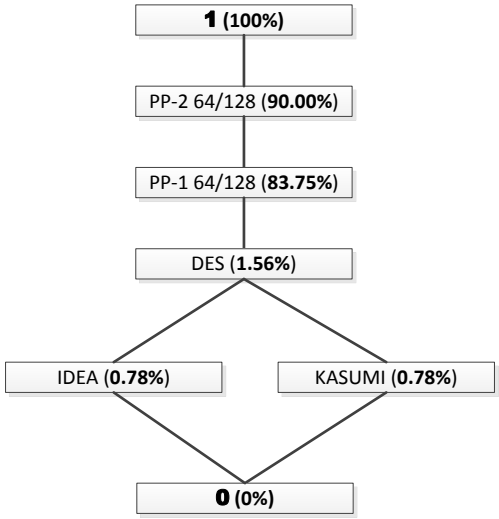
W ramach całościowej, teoretycznej oceny jakości algorytmu wykonano:

- klasyfikację algorytmów ze względu na zależność (całego) klucza rundowego od bitów klucza głównego,
- klasyfikację algorytmów ze względu na zależność (pojedynczego) bitu klucza rundowego od bitów klucza głównego,
- uporządkowano algorytmy ze względu na przyjęte teoretyczne kryterium jakości (rys. 3.56).

W rozdziale czwartym przedstawiono:

- metodologię testów statystycznych, służących do badania generatorów ciągów losowych i pseudolosowych, w tym zasady badania hipotez statystycznych H_0 i H_a (podrozdział 4.1),
- szczegółowy opis pakietu testów NIST SP800-22, z zapisem testów w postaci procedur i omówieniem pliku wyników (podrozdział 4.2),
- wyniki oceny algorytmów generowania kluczy rundowych szyfrów DES, IDEA, KASUMI, PP-1, PP-2, z wykorzystaniem pakietu NIST SP800-22 (metoda A, podpunkt A1),
- projektowanie algorytmu generowania kluczy rundowych, na przykładzie szyfru PP-1 64/64, z wykorzystaniem pakietu NIST SP800-22 (metoda A, podpunkt A2),

- wyniki oceny algorytmu generowania kluczy rundowych szyfru PP-2 64/128, traktowanego jako generator liczb pseudolosowych, z wykorzystaniem pakietu NIST SP800-22 (metoda B, podpunkt B1),
- wyniki oceny algorytmu generowania kluczy rundowych szyfru PP-1 64/128, traktowanego jako generator liczb pseudolosowych, z wykorzystaniem pakietu NIST SP800-22 (metoda B, podpunkt B2),
- wyniki oceny algorytmów generowania kluczy rundowych szyfrów DES, IDEA, KASUMI, PP-1, PP-2, z wykorzystaniem wszystkich testów pakietu NIST SP800-22 (metoda C),
- wyniki oceny początkowych kluczy algorytmów generowania kluczy rundowych szyfrów DES, IDEA, KASUMI, PP-1, PP-2, z wykorzystaniem pakietu NIST SP800-22 (metoda D),
- optymalizację algorytmu generowania kluczy rundowych w szyfrze IDEA, polegającą na wyborze najlepszej wartości parametru rotacji (metoda E, podpunkt E1),
- optymalizację algorytmu generowania kluczy rundowych szyfru PP-1 64/64, z uwzględnieniem czasu obliczeń (metoda E, podpunkt E2),
- optymalizację algorytmu generowania kluczy rundowych szyfru PP-1 64/64, bez uwzględnienia czasu obliczeń (metoda E, podpunkt E3).



Rys. 5.1. Uporządkowanie algorytmów generowania kluczy rundowych szyfrów DES, IDEA, KASUMI, PP-1, PP-2, według teoretycznego kryterium jakości, potwierdzone testami NIST SP800-22 (wniosek 4.1)

Wnioski wynikające z badań są następujące:

- w metodzie A (próbek), przy losowych wartościach kluczy głównych (formuła (4.9)), algorytm generowania kluczy rundowych w szyfrach DES, IDEA, KASUMI, nie spełnia kryterium jakości (losowości), a w szyfrach PP-1 i PP-2 spełnia (wniosek 4.1, rys. 5.1),
- w metodzie A (próbek), przy losowych wartościach kluczy głównych (formuła (4.9)), w wariancie oryginalnym algorytmu generowania kluczy rundowych szyfru PP-1 64/64 oraz w jego wersjach (modyfikacjach): wersja 1 bez rotacji *RR*, wersja 2 z operacjami XOR, wersja 3.3 z sześcioma S-blokami i wersja 5 złożona z pozytywnych modyfikacji (bez *RR*, XOR, 6 S-bloków), spełnione jest kryterium jakości (losowości), a w pozostałych wersjach: 3.1, 3.2, 4.1, 4.2, 6, 7 – nie jest spełnione to kryterium (wniosek 4.2),
- w metodzie B (nadpróbek), przy kolejnych wartościach kluczy głównych (formuła (4.10)), rozszerzony algorytm generowania kluczy rundowych w szyfrze PP-2 64/128 jest dobrym jakościowo generatorem liczb pseudolosowych (wniosek 4.3)
- w metodzie B (nadpróbek), przy kolejnych wartościach kluczy głównych (formuła (4.10)), rozszerzony algorytm generowania kluczy rundowych szyfru PP-1 64/128 można uznać za dobry jakościowo generator liczb pseudolosowych (wniosek 4.4),
- w metodzie C (metapróbek), przy losowych wartościach kluczy głównych (formuła (4.9)), algorytm generowania kluczy rundowych szyfrów PP-1 64/128 i PP-2 64/128 spełnia, a szyfrów DES, IDEA, KASUMI nie spełnia kryterium jakości (losowości), natomiast przy kolejnych wartościach kluczy głównych (formuła (4.10)) żaden z wymienionych algorytmów nie spełnia tego kryterium (wniosek 4.5),
- w metodzie D (podpróbek), przy wariancie losowych kluczy głównych (formuła (4.9)), algorytmy generowania kluczy rundowych szyfrów PP-1 64/128 oraz PP-2 64/128 spełniają kryterium jakości (losowości), a szyfrów DES, IDEA, KASUMI – nie spełniają tego kryterium (wniosek 4.6),
- w metodzie E (hybrydowej), przy wariancie losowych kluczy głównych (formuła (4.9)), optymalnym wariantem algorytmu generowania kluczy rundowych w szyfrze IDEA, jest wariant z operacją rotacji w lewo o 71 bitów (wniosek 4.7),
- w metodzie E (hybrydowej), przy wariancie losowych kluczy głównych (formuła (4.9)), optymalnym wariantem algorytmu generowania kluczy rundowych w

szyfrze PP-1 64/64, z uwzględnieniem czasu obliczeń, jest wariant złożony z pozytywnych modyfikacjami, tj. bez rotacji *RR*, z operacjami XOR zamiast dodawania i odejmowania modulo 256 oraz z liczbą S-bloków zredukowaną do sześciu (wniosek 4.8),

- w metodzie E (hybrydowej), przy wariancie losowych kluczy głównych (formuła (4.9)), optymalnym wariantem algorytmu generowania kluczy rundowych w szyfrze PP-1 64/64, z pominięciem czasu obliczeń, jest wariant z sześcioma S-blokami (wniosek 4.9).

Na szczególne zwrócenie uwagi, w rozprawie, zasługują:

- omówienie ogólnej struktury szyfru blokowego (podrozdział 3.1),
- opis szyfru AES (punkt 3.3.1) i algorytmu generowania kluczy rundowych szyfru PP-2 (punkt 3.3.6),
- metoda określenia zależności bitu klucza rundowego od bitów klucza głównego, przyjętej jako podstawa teoretycznej oceny jakości algorytmu generowania kluczy rundowych,
- wyodrębnienie metod A, B, C, D, na podstawie próbek, nadpróbek, metapróbek i podpróbek, stanowiących różne warianty konkatenacji kluczy rundowych,
- metoda E (hybrydowa), zastosowania testów statystycznych NIST SP800-22 w powiazaniu z analizą skupień, do optymalizacji algorytmu generowania kluczy rundowych szyfrów IDEA oraz PP-1 64/64.

Przedstawione wyniki badań pozwalają wyrazić przekonanie o udowodnieniu tezy rozprawy:

Proponowana metoda „hybrydowa" oceny jakości algorytmów generowania kluczy rundowych szyfrów blokowych, oparta na testach statystycznych i analizie skupień, umożliwia wyznaczenie optymalnego wariantu algorytmu, najbliższego hipotetycznie najlepszemu rozwiązaniu dla zbioru testowanych wariantów, spełniających zadane kryteria projektowe.

Wartość praktyczna opracowanej metody oceny algorytmów generowania kluczy rundowych polega na możliwości jej wykorzystania jako elementu wspomagającego proces projektowania tych algorytmów (jak i całych szyfrów), w szczególności gdy istotne jest wyznaczenie rozwiązania optymalnego.

Interesującym kierunkiem badań, w zakresie metod oceny jakości algorytmu generowania kluczy rundowych w szyfrach blokowych, jest zastosowanie innych teoretycznych i praktycznych kryteriów oceny jakości tych algorytmów. Z rozważań przeprowadzonych w rozprawie wynika, że takim kryterium teoretycznym może być dyfuzja, a kryterium praktycznym – testy statystyczne spoza pakietu NIST SP800-22.

6. Dodatki

Tab. 6.1. Kompletna tabela wyników *p* testów NIST SP800-22 dla różnych wariantów rotacji w algorytmie generowania kluczy rundowych szyfru IDEA (PA = <0.9806; 0.9994>)

wariant rotacji *a*	rodzaj testu				
	frequency	block-frequency (*M* = 64)	cumulative-sums (forward)	cumulative-sums (backward)	runs
1	0,6910	0,8940	0,7080	0,7150	0,6100
2	0,6900	0,8900	0,7090	0,7040	**0,6220**
3	0,6900	0,9000	0,7070	0,7060	0,6030
4	0,6840	0,8990	0,7090	0,7040	0,6030
5	0,6900	0,9010	0,7200	0,7100	0,5930
6	0,6960	0,9110	**0,7220**	0,7100	0,5970
7	0,7000	0,9170	0,7180	**0,7200**	0,5930
8	0,7020	0,9150	0,7140	0,7160	0,6010
9	0,7000	0,9130	0,7080	0,7140	0,5950
10	0,6920	0,9130	0,7140	0,7020	0,5940
11	0,6820	0,9070	0,7090	0,7030	0,6010
12	0,6920	0,9070	0,7050	0,6960	0,6000
13	0,6900	0,9150	0,7120	0,7040	0,5840
14	0,6910	0,9050	0,7060	0,6990	0,5980
15	0,6880	0,9130	0,7020	0,6980	0,5940
16	0,6840	0,9130	0,7080	0,7060	0,6040
17	0,6840	0,9170	0,7020	0,7040	0,6040
18	0,6810	0,9170	0,7040	0,6990	0,6040
19	0,6860	0,9130	0,7020	0,7020	0,6000
20	0,6910	0,9220	0,7020	0,7040	0,5980
21	0,6910	0,9220	0,7040	0,7090	0,6110
22	0,6890	0,9190	0,7120	0,7070	0,6000
23	0,6840	0,9140	0,7160	0,7060	0,6120
24	0,6920	0,9150	0,7100	0,7060	0,6160
25	0,6860	0,9150	0,7040	0,6980	0,6030
26	0,6840	0,9170	0,7080	0,7040	0,6030
27	0,6940	0,9190	0,7100	0,7120	0,5990
28	0,7000	0,9130	**0,7220**	0,7140	0,6010
29	**0,7040**	0,9130	0,7140	0,7160	0,5920
30	0,7000	0,9130	0,7100	0,7120	0,5930
31	0,6880	0,9170	0,7120	0,7040	0,5940
32	0,6880	0,9090	0,7080	0,7000	0,5910
33	0,6840	0,9090	0,7110	0,6990	0,5970
34	0,6880	0,9090	0,7100	0,7010	0,5920
35	0,6850	0,9090	0,7080	0,7010	0,5960
36	0,6850	**0,9230**	0,7000	0,6970	0,6030
37	0,6740	0,9150	0,7060	0,7000	0,6040
38	0,6900	0,9150	0,7000	0,6980	0,6080
39	0,6830	0,9190	0,7060	0,6950	0,6020
40	0,6800	0,9170	0,7000	0,7000	0,6000
41	0,6920	0,9210	0,7020	0,7060	0,6030
42	0,6850	0,9220	0,7100	0,7050	0,6040

43	0,6850	0,9210	0,7080	0,7010	0,6080
44	0,6930	0,9170	0,7140	0,7150	0,6140
45	0,6920	0,9190	0,7140	0,7040	0,6040
46	0,6900	0,9120	0,7040	0,7060	0,6120
47	0,6840	0,9130	0,7100	0,7040	0,6030
48	0,6880	0,9150	0,7100	0,7100	0,5990
49	0,6960	0,9150	0,7180	0,7140	0,6010
50	0,6940	0,9110	0,7160	0,7180	0,5990
51	0,7000	0,9110	0,7080	0,7140	0,6030
52	0,6940	0,9130	0,7120	0,7060	0,5950
53	0,6880	0,9090	0,7080	0,7000	0,5950
54	0,6860	0,9130	0,7130	0,7030	0,6070
55	0,6880	0,9110	0,7080	0,7030	0,5880
56	0,6870	0,9110	0,7140	0,7020	0,5920
57	0,6870	0,9190	0,7040	0,6990	0,6000
58	0,6860	0,9170	0,7040	0,7040	0,6000
59	0,6880	0,9110	0,7000	0,7020	0,6120
60	0,6840	0,9030	0,7020	0,7060	0,5940
61	0,6820	0,8930	0,7020	0,6990	0,5960
62	0,6920	0,8940	0,7060	0,7080	0,6160
63	0,6910	0,8890	0,7060	0,7070	0,6150
64	0,6990	0,8860	0,7080	0,7110	0,6180
65	0,6910	0,8940	0,7080	0,7150	0,6040
66	0,6900	0,8900	0,7070	0,7040	0,6170
67	0,6900	0,9000	0,7050	0,7060	0,6030
68	0,6840	0,8990	0,7090	0,7040	0,6030
69	0,6900	0,9010	0,7200	0,7100	0,6030
70	0,6960	0,9110	0,7180	0,7100	0,5970
71	0,7000	0,9170	0,7160	**0,7200**	0,6050
72	0,7020	0,9150	0,7120	0,7160	0,6020
73	0,7000	0,9130	0,7040	0,7140	0,5940
74	0,6920	0,9130	0,7120	0,7020	0,6040
75	0,6820	0,9070	0,7050	0,7030	0,6050
76	0,6920	0,9070	0,7010	0,6960	0,5980
77	0,6900	0,9150	0,7060	0,7040	0,5840
78	0,6910	0,9050	0,7040	0,6990	0,6000
79	0,6880	0,9130	0,7020	0,7000	0,5940
80	0,6840	0,9130	0,7060	0,7060	0,5980
81	0,6840	0,9170	0,7000	0,7040	0,6040
82	0,6810	0,9170	0,7020	0,6990	0,5960
83	0,6860	0,9130	0,6980	0,7020	0,6010
84	0,6910	0,9220	0,6960	0,7040	0,5940
85	0,6910	0,9220	0,6980	0,7090	0,6130
86	0,6890	0,9190	0,7080	0,7070	0,6040
87	0,6840	0,9140	0,7120	0,7060	0,6130
88	0,6920	0,9150	0,7100	0,7060	0,6070
89	0,6860	0,9150	0,7040	0,6980	0,6020
90	0,6840	0,9170	0,7120	0,7040	0,5950
91	0,6940	0,9190	0,7100	0,7120	0,5950
92	0,7000	0,9130	0,7200	0,7140	0,5950
93	**0,7040**	0,9130	0,7140	0,7160	0,5910
94	0,7000	0,9130	0,7100	0,7120	0,5960

95	0,6880	0,9170	0,7120	0,7040	0,5920
96	0,6880	0,9090	0,7040	0,7000	0,5930
97	0,6840	0,9090	0,7090	0,6990	0,5880
98	0,6880	0,9090	0,7100	0,7010	0,5820
99	0,6850	0,9090	0,7060	0,7010	0,5950
100	0,6850	**0,9230**	0,7000	0,6970	0,6000
101	0,6740	0,9150	0,7040	0,7000	0,5980
102	0,6900	0,9150	0,7020	0,6980	0,6060
103	0,6830	0,9190	0,7040	0,6950	0,6040
104	0,6800	0,9170	0,7000	0,7000	0,5960
105	0,6920	0,9210	0,7040	0,7060	0,6070
106	0,6850	0,9220	0,7080	0,7050	0,6010
107	0,6850	0,9210	0,7060	0,7010	0,6040
108	0,6930	0,9170	0,7120	0,7150	0,6120
109	0,6920	0,9190	0,7120	0,7040	0,6050
110	0,6900	0,9120	0,7020	0,7060	0,6060
111	0,6840	0,9130	0,7120	0,7040	0,6030
112	0,6880	0,9150	0,7160	0,7100	0,5970
113	0,6960	0,9150	**0,7220**	0,7140	0,6010
114	0,6940	0,9110	0,7200	0,7180	0,6070
115	0,7000	0,9110	0,7120	0,7140	0,5970
116	0,6940	0,9130	0,7140	0,7060	0,5980
117	0,6880	0,9090	0,7060	0,7000	0,5960
118	0,6860	0,9130	0,7130	0,7030	0,6010
119	0,6880	0,9110	0,7080	0,7030	0,5900
120	0,6870	0,9110	0,7120	0,7020	0,5880
121	0,6870	0,9190	0,7040	0,6990	0,5970
122	0,6860	0,9170	0,706	0,7040	0,5960
123	0,6880	0,9110	0,7040	0,7020	0,6010
124	0,6840	0,9030	0,7060	0,7060	0,6060
125	0,6820	0,8930	0,7040	0,6990	0,6020
126	0,6920	0,8940	0,7080	0,7080	0,6100
127	0,6910	0,8890	0,7080	0,7070	0,6010
max_j wartość maksymalna	**0,7040**	**0,9230**	**0,7220**	**0,7200**	**0,6220**
$\bar{x}_j$ wartość średnia	0,689477	0,911695	0,708391	0,705430	0,600875
s_j odchylenie standardowe	0,005934	0,008127	0,005672	0,005964	0,007470

Tab. 6.2. Kompletna tabela wartości standaryzowanych cech badanych obiektów w algorytmie IDEA

obiekty reprezentujące warianty rotacji	cechy obiektów				
	Y_1	Y_2	Y_3	Y_4	Y_5
O_1	0,2567	-2,1772	-0,0689	1,6047	1,2216
O_2	0,0882	-2,6694	0,1074	-0,2397	2,8281
O_3	0,0882	-1,4390	-0,2452	0,0956	0,2845
O_4	-0,9229	-1,5620	0,1074	-0,2397	0,2845
O_5	0,0882	-1,3159	2,0466	0,7663	-1,0543
O_6	1,0993	-0,0856	2,3992	0,7663	-0,5188
O_7	1,7734	0,6527	1,6940	2,4430	-1,0543

O_8	2,1105	0,4066	0,9889	1,7723	0,0167
O_9	1,7734	0,1605	-0,0689	1,4370	-0,7865
O_{10}	0,4253	0,1605	0,9889	-0,5751	-0,9204
O_{11}	-1,2600	-0,5777	0,1074	-0,4074	0,0167
O_{12}	0,4253	-0,5777	-0,5977	-1,5811	-0,1171
O_{13}	0,0882	0,4066	0,6363	-0,2397	-2,2592
O_{14}	0,2567	-0,8238	-0,4214	-1,0781	-0,3849
O_{15}	-0,2488	0,1605	-1,1266	-1,2457	-0,9204
O_{16}	-0,9229	0,1605	-0,0689	0,0956	0,4184
O_{17}	-0,9229	0,6527	-1,1266	-0,2397	0,4184
O_{18}	-1,4285	0,6527	-0,7740	-1,0781	0,4184
O_{19}	-0,5859	0,1605	-1,1266	-0,5751	-0,1171
O_{20}	0,2567	1,2679	-1,1266	-0,2397	-0,3849
O_{21}	0,2567	1,2679	-0,7740	0,5986	1,3555
O_{22}	-0,0803	0,8988	0,6363	0,2633	-0,1171
O_{23}	-0,9229	0,2836	1,3415	0,0956	1,4894
O_{24}	0,4253	0,4066	0,2837	0,0956	2,0249
O_{25}	-0,5859	0,4066	-0,7740	-1,2457	0,2845
O_{26}	-0,9229	0,6527	-0,0689	-0,2397	0,2845
O_{27}	0,7623	0,8988	0,2837	1,1017	-0,2510
O_{28}	1,7734	0,1605	2,3992	1,4370	0,0167
O_{29}	2,4475	0,1605	0,9889	1,7723	-1,1881
O_{30}	1,7734	0,1605	0,2837	1,1017	-1,0543
O_{31}	-0,2488	0,6527	0,6363	-0,2397	-0,9204
O_{32}	-0,2488	-0,3316	-0,0689	-0,9104	-1,3220
O_{33}	-0,9229	-0,3316	0,4600	-1,0781	-0,5188
O_{34}	-0,2488	-0,3316	0,2837	-0,7427	-1,1881
O_{35}	-0,7544	-0,3316	-0,0689	-0,7427	-0,6526
O_{36}	-0,7544	1,3909	-1,4792	-1,4134	0,2845
O_{37}	-2,6081	0,4066	-0,4214	-0,9104	0,4184
O_{38}	0,0882	0,4066	-1,4792	-1,2457	0,9539
O_{39}	-1,0914	0,8988	-0,4214	-1,7488	0,1506
O_{40}	-1,5970	0,6527	-1,4792	-0,9104	-0,1171
O_{41}	0,4253	1,1448	-1,1266	0,0956	0,2845
O_{42}	-0,7544	1,2679	0,2837	-0,0720	0,4184
O_{43}	-0,7544	1,1448	-0,0689	-0,7427	0,9539
O_{44}	0,5938	0,6527	0,9889	1,6047	1,7571
O_{45}	0,4253	0,8988	0,9889	-0,2397	0,4184
O_{46}	0,0882	0,0375	-0,7740	0,0956	1,4894
O_{47}	-0,9229	0,1605	0,2837	-0,2397	0,2845
O_{48}	-0,2488	0,4066	0,2837	0,7663	-0,2510
O_{49}	1,0993	0,4066	1,6940	1,4370	0,0167
O_{50}	0,7623	-0,0856	1,3415	2,1077	-0,2510
O_{51}	1,7734	-0,0856	-0,0689	1,4370	0,2845
O_{52}	0,7623	0,1605	0,6363	0,0956	-0,7865
O_{52}	-0,2488	-0,3316	-0,0689	-0,9104	-0,7865
O_{54}	-0,5859	0,1605	0,8126	-0,4074	0,8200
O_{55}	-0,2488	-0,0856	-0,0689	-0,4074	-1,7236
O_{56}	-0,4174	-0,0856	0,9889	-0,5751	-1,1881
O_{57}	-0,4174	0,8988	-0,7740	-1,0781	-0,1171
O_{58}	-0,5859	0,6527	-0,7740	-0,2397	-0,1171

O_{59}	-0,2488	-0,0856	-1,4792	-0,5751	1,4894
O_{60}	-0,9229	-1,0699	-1,1266	0,0956	-0,9204
O_{61}	-1,2600	-2,3003	-1,1266	-1,0781	-0,6526
O_{62}	0,4253	-2,1772	-0,4214	0,4310	2,0249
O_{63}	0,2567	-2,7924	-0,4214	0,2633	1,8910
O_{64}	1,6049	-3,1615	-0,0689	0,9340	2,2926
O_{65}	0,2567	-2,1772	-0,0689	1,6047	0,4184
O_{66}	0,0882	-2,6694	-0,2452	-0,2397	2,1587
O_{67}	0,0882	-1,4390	-0,5977	0,0956	0,2845
O_{68}	-0,9229	-1,5620	0,1074	-0,2397	0,2845
O_{69}	0,0882	-1,3159	2,0466	0,7663	0,2845
O_{70}	1,0993	-0,0856	1,6940	0,7663	-0,5188
O_{71}	1,7734	0,6527	1,3415	2,4430	0,5522
O_{72}	2,1105	0,4066	0,6363	1,7723	0,1506
O_{73}	1,7734	0,1605	-0,7740	1,4370	-0,9204
O_{74}	0,4253	0,1605	0,6363	-0,5751	0,4184
O_{75}	-1,2600	-0,5777	-0,5977	-0,4074	0,5522
O_{76}	0,4253	-0,5777	-1,3029	-1,5811	-0,3849
O_{77}	0,0882	0,4066	-0,4214	-0,2397	-2,2592
O_{78}	0,2567	-0,8238	-0,7740	-1,0781	-0,1171
O_{79}	-0,2488	0,1605	-1,1266	-0,9104	-0,9204
O_{80}	-0,9229	0,1605	-0,4214	0,0956	-0,3849
O_{81}	-0,9229	0,6527	-1,4792	-0,2397	0,4184
O_{82}	-1,4285	0,6527	-1,1266	-1,0781	-0,6526
O_{83}	-0,5859	0,1605	-1,8318	-0,5751	0,0167
O_{84}	0,2567	1,2679	-2,1843	-0,2397	-0,9204
O_{85}	0,2567	1,2679	-1,8318	0,5986	1,6232
O_{86}	-0,0803	0,8988	-0,0689	0,2633	0,4184
O_{87}	-0,9229	0,2836	0,6363	0,0956	1,6232
O_{88}	0,4253	0,4066	0,2837	0,0956	0,8200
O_{89}	-0,5859	0,4066	-0,7740	-1,2457	0,1506
O_{90}	-0,9229	0,6527	0,6363	-0,2397	-0,7865
O_{91}	0,7623	0,8988	0,2837	1,1017	-0,7865
O_{92}	1,7734	0,1605	2,0466	1,4370	-0,7865
O_{93}	2,4475	0,1605	0,9889	1,7723	-1,3220
O_{94}	1,7734	0,1605	0,2837	1,1017	-0,6526
O_{95}	-0,2488	0,6527	0,6363	-0,2397	-1,1881
O_{96}	-0,2488	-0,3316	-0,7740	-0,9104	-1,0543
O_{97}	-0,9229	-0,3316	0,1074	-1,0781	-1,7236
O_{98}	-0,2488	-0,3316	0,2837	-0,7427	-2,5269
O_{99}	-0,7544	-0,3316	-0,4214	-0,7427	-0,7865
O_{100}	-0,7544	1,3909	-1,4792	-1,4134	-0,1171
O_{101}	-2,6081	0,4066	-0,7740	-0,9104	-0,3849
O_{102}	0,0882	0,4066	-1,1266	-1,2457	0,6861
O_{103}	-1,0914	0,8988	-0,7740	-1,7488	0,4184
O_{104}	-1,5970	0,6527	-1,4792	-0,9104	-0,6526
O_{105}	0,4253	1,1448	-0,7740	0,0956	0,8200
O_{106}	-0,7544	1,2679	-0,0689	-0,0720	0,0167
O_{107}	-0,7544	1,1448	-0,4214	-0,7427	0,4184
O_{108}	0,5938	0,6527	0,6363	1,6047	1,4894
O_{109}	0,4253	0,8988	0,6363	-0,2397	0,5522

O_{110}	0,0882	0,0375	-1,1266	0,0956	0,6861
O_{111}	-0,9229	0,1605	0,6363	-0,2397	0,2845
O_{112}	-0,2488	0,4066	1,3415	0,7663	-0,5188
O_{113}	1,0993	0,4066	2,3992	1,4370	0,0167
O_{114}	0,7623	-0,0856	2,0466	2,1077	0,8200
O_{115}	1,7734	-0,0856	0,6363	1,4370	-0,5188
O_{116}	0,7623	0,1605	0,9889	0,0956	-0,3849
O_{117}	-0,2488	-0,3316	-0,4214	-0,9104	-0,6526
O_{118}	-0,5859	0,1605	0,8126	-0,4074	0,0167
O_{119}	-0,2488	-0,0856	-0,0689	-0,4074	-1,4559
O_{120}	-0,4174	-0,0856	0,6363	-0,5751	-1,7236
O_{121}	-0,4174	0,8988	-0,7740	-1,0781	-0,5188
O_{122}	-0,5859	0,6527	-0,4214	-0,2397	-0,6526
O_{123}	-0,2488	-0,0856	-0,7740	-0,5751	0,0167
O_{124}	-0,9229	-1,0699	-0,4214	0,0956	0,6861
O_{125}	-1,2600	-2,3003	-0,7740	-1,0781	0,1506
O_{126}	0,4253	-2,1772	-0,0689	0,4310	1,2216
O_{127}	0,2567	-2,7924	-0,0689	0,2633	0,0167
$O_{128(hip)}$ (hipotetycznie najlepszy obiekt)	**2,4475**	**1,3909**	**2,3992**	**2,4430**	**2,8281**

Spisy

Spis rysunków

Rys. 2.1. Zapewnienie poufności z wykorzystaniem kryptografii symetrycznej 15
Rys. 2.2. Szyfr strumieniowy – szyfrowanie i deszyfrowanie 17
Rys. 2.3. Szyfr blokowy – szyfrowanie i deszyfrowanie 17
Rys. 2.4. Zapewnienie poufności z wykorzystaniem kryptografii asymetrycznej 20
Rys. 2.5. Uwierzytelnienie nadawcy i zapewnienie niezaprzeczalności w kryptografii asymetrycznej 21
Rys. 2.6. Zastosowanie funkcji skrótu do weryfikacji integralności danych 23
Rys. 2.7. Generowanie podpisu przez użytkownika A 24
Rys. 2.8. Weryfikacja podpisu i integralności danych przez użytkownika B 25
Rys. 3.1. Wewnętrzna struktura funkcji: (a) szyfrującej E_k, (b) deszyfrującej D_k 28
Rys. 3.2. Szyfr SPN – podstawowa struktura funkcji: (a) szyfrującej, (b) deszyfrującej 30
Rys. 3.3. Szyfr Feistela – podstawowa struktura funkcji: (a) szyfrującej/deszyfrującej, (b) bazowej *f* 32
Rys. 3.4. DES – struktura funkcji: (a) szyfrującej/deszyfrującej, (b) bazowej *f* 34
Rys. 3.5. DES – struktura funkcji generowania kluczy rundowych 35
Rys. 3.6. DES – schemat zastępczy funkcji generowania klucza rundowego K_i (i = 1, 2, …, 16) 38
Rys. 3.7. 3DES –przekształcenie bloku *m* wiadomości w blok *c′* szyfrogramu 40
Rys. 3.8. 3DES – schemat zastępczy funkcji generowania klucza rundowego K_i (i = 1, 2, …, 48) 41
Rys. 3.9. LOKI97 – struktura funkcji: (a) szyfrującej/deszyfrującej, (b) generowania kluczy rundowych 43
Rys. 3.10. LOKI97 – struktura funkcji *f*(*A*, *B*) 44
Rys. 3.11. LOKI97 – zależność bitu $SK_1[63]$ klucza rundowego od bitów klucza głównego $k = K4_0||K3_0||K2_0||K1_0$ 47
Rys. 3.12. LOKI97 – (a) schemat zastępczy funkcji *f*(*A*,*B*), (b) zależność bitu $SK_2[63]$ klucza rundowego od bitów klucza głównego $k = K4_0||K3_0||K2_0||K1_0$ 48
Rys. 3.13. SM4 – struktura: (a) rundy funkcji szyfrującej/deszyfrującej, (b) iteracji funkcji generowania kluczy rundowych (i = 0, 1, …, 31) 49

Rys. 3.14. SM4 – zależność bitu $k_1[0]$ klucza rundowego od bitów klucza głównego $k = (MK_0, MK_1, MK_2, MK_3)$ 52
Rys. 3.15. KASUMI – struktura funkcji: (a) szyfrującej/deszyfrującej, (b) składowych funkcji bazowej *FO, FI* i *FL* 54
Rys. 3.16. KASUMI – funkcja generowania kluczy rundowych rundy i (i = 1, 2, …, 8) 55
Rys. 3.17. KASUMI – schemat zastępczy funkcji generowania klucza rundowego CK_i (i = 1, 2, …, 8) 58
Rys. 3.18. FeW – struktura: (a) rundy i (i = 0, 1, …, 31) funkcji szyfrującej/deszyfrującej, (b) funkcji bazowej f 59
Rys. 3.19. FeW – struktura funkcji generowania podkluczy rundowych RK_i (i = 0, 1, …, 63), w przypadku: (a) $|k|$ = 80b, (b) $|k|$ = 128b 60
Rys. 3.20. FeW 64/128 – zależność bitu $RK_8[8]$ podklucza rundowego od bitów klucza głównego k 62
Rys. 3.21. LCB-IoT – struktura: (a) rundy i (i = 1, 2, …, 31) funkcji szyfrującej/deszyfrującej, (b) funkcji bazowej f 64
Rys. 3.22. LCB-IoT – struktura funkcji generowania kluczy rundowych (i = 6, 7, …, 32) 65
Rys. 3.23. LCB-IoT – zależność bitu $K_{10}[0]$ klucza rundowego od bitów klucza głównego k 67
Rys. 3.24. AES – struktura funkcji szyfrującej 70
Rys. 3.25. AES-128 – funkcja generowania kluczy rundowych 72
Rys. 3.26. AES-128 – zależność bitu $kr_5[0]$ klucza rundowego od bitów klucza głównego k 75
Rys. 3.27. Serpent – struktura funkcji: (a) szyfrującej, (b) liniowej L 78
Rys. 3.28. Serpent – struktura funkcji generowania klucza rundowego K_i (i = 0, 1,…, 32).... 79
Rys. 3.29. Serpent – zależność bitu $K_4[0]$ klucza rundowego od bitów klucza głównego $k = w_{-8}||w_{-7}||...||w_{-1}$ 81
Rys. 3.30. SAFER+ – struktura rundy #i funkcji szyfrującej (i = 1, 2, …, r) 84
Rys. 3.31. SAFER+ – struktura funkcji generowania kluczy rundowych z kluczem 128-bitowym 85
Rys. 3.32. SAFER+ 128/128 – zależność bitu $K_i[0]$ klucza rundowego od bitów klucza głównego k 87
Rys. 3.33. PRESENT – struktura funkcji szyfrującej 89
Rys. 3.34. PRESENT – struktura funkcji generowania kluczy rundowych K_i (i = 1, 2, …, 32), w przypadku: (a) $|k|$ = 80b, (b) $|k|$ = 128b 90

Rys. 3.35. PRESENT-128 – zależność bitu $K_5[0]$ klucza rundowego od bitów klucza głównego k 92
Rys. 3.36. PP-1 – struktura: (a) rundy #i (i = 1, 2, …, r) szyfru, (b) elementu nieliniowego NL #j (j = 1, 2, …, t) (szyfrowanie/deszyfrowanie) 94
Rys. 3.37. PP-1 – struktura: (a) iteracji #i (i = 0, 1, …, 2r) funkcji generowania kluczy rundowych, (b) elementu KS #j (j = 1, 2, …, t) 95
Rys. 3.38. PP-1 – struktura funkcji generowania kluczy rundowych dla szyfru: (a) 64/128, (b) 128/128 97
Rys. 3.39. PP-1 64/128 – zależność bitu $k_3[1]$ klucza rundowego od bitów klucza głównego $k = k_H||k_L$ 99
Rys. 3.40. PP-2 – struktura: (a) rundy #i (i = 1, 2, …, r) szyfru, (b) elementu nieliniowego NL #j (j = 1, 2, …, t) (szyfrowanie) 101
Rys. 3.41. PP-2 – struktura: (a) iteracji sch_i (i = 1, 2, …, R) funkcji generowania kluczy rundowych, (b) elementu KS #j (j = 1, 2, …, t) 103
Rys. 3.42. PP-2 – struktura funkcji generowania kluczy rundowych dla szyfru: (a) 64/128, (b) 128/256 104
Rys. 3.43. PP-2 64/128 – zależność bitu $k_1[1]$ klucza rundowego od bitów klucza głównego $k = \kappa_1||\kappa_2$ 106
Rys. 3.44. PP-2 64/128 – schemat zastępczy funkcji generowania klucza rundowego k_i: (a) i = 1, (b) i = 2, 3, …, r = 13 107
Rys. 3.45. PP-2′ – struktura funkcji generowania kluczy rundowych dla szyfru: (a) 64/128, (b) 128/256 108
Rys. 3.46. IDEA – struktura funkcji szyfrującej/deszyfrującej 109
Rys. 3.47. IDEA – funkcja generowania kluczy rundowych 110
Rys. 3.48. IDEA – schemat zastępczy funkcji generowania klucza rundowego k_i: (a) i = 1, 2, …, 8, (b) i = 9 112
Rys. 3.49. RC6 – struktura funkcji: (a) szyfrującej (b) deszyfrującej 114
Rys. 3.50. RC6 – iteracja s funkcji generowania kluczy rundowych (s = 1, 2, …, v), gdzie $v = 3 \cdot \max(c, 2r + 4)$ i $c = \lceil 8b / w \rceil$ 115
Rys. 3.51. RC6-32/20/16 – zależność bitu $S[2][0]$ podklucza rundowego od bitów klucza głównego $k = L[0]||L[1]||L[2]||L[3]$ 118
Rys. 3.52. Speck – struktura rundy i (i = 0, 1, …, r − 1): (a) szyfrowanie, (b) deszyfrowanie 120

Rys. 3.53. Speck – struktura: (a) iteracji *i* funkcji generowania kluczy rundowych, (b) funkcji R_i ($i = 0, 1, \ldots, r-1$) 121
Rys. 3.54. Speck64/128 – zależność bitu $sk_4[0]$ klucza rundowego od bitów klucza głównego $k = k_3||k_2||k_1||k_0$ 123
Rys. 3.55. Uporządkowanie algorytmów generowania kluczy rundowych ze względu na warstwę: (a) podstawień, (b) dyfuzji 131
Rys. 3.56. Uporządkowanie algorytmów generowania kluczy rundowych według teoretycznego kryterium jakości 135
Rys. 4.1. Przykład raportu z testów NIST SP800-22 – zawartość pliku *finalAnalysisReport.txt* 151
Rys. 4.2. Wykres wyników *p* testów NIST SP800-22 przy zastosowaniu metody A (PA = <0.9806; 0.9994>) 156
Rys. 4.3. PP-1 – wersja (0) oryginalna algorytmu generowania kluczy rundowych, struktura: (a) iteracji #*i* ($i = 0, 1, \ldots, 2r$) funkcji generowania kluczy rundowych, (b) elementu *KS #j* ($j = 1, 2, \ldots, t$) 157
Rys. 4.4. PP-1 – wersja 1 algorytmu generowania kluczy rundowych bez rotacji *RR*, struktura: (a) iteracji #*i* ($i = 0, 1, \ldots, 2r$) funkcji generowania kluczy rundowych, (b) elementu *KS #j* ($j = 1, 2, \ldots, t$) 158
Rys. 4.5. PP-1 – wersja 2 algorytmu generowania kluczy rundowych z operacjami XOR zamiast sumy i różnicy modulo 256, struktura: (a) iteracji #*i* ($i = 0, 1, \ldots, 2r$) funkcji generowania kluczy rundowych, (b) elementu *KS #j* ($j = 1, 2, \ldots, t$) 159
Rys. 4.6. PP-1 – wersja 3.1 algorytmu generowania kluczy rundowych bez S-bloków, struktura: (a) iteracji #*i* ($i = 0, 1, \ldots, 2r$) funkcji generowania kluczy rundowych, (b) elementu *KS #j* ($j = 1, 2, \ldots, t$) 160
Rys. 4.7. PP-1 – wersja 3.2 algorytmu generowania kluczy rundowych z 4 S-blokami, struktura: (a) iteracji #*i* ($i = 0, 1, \ldots, 2r$) funkcji generowania kluczy rundowych, (b) elementu *KS #j* ($j = 1, 2, \ldots, t$) 161
Rys. 4.8. PP-1 – wersja 3.3 algorytmu generowania kluczy rundowych z 6 S-blokami, struktura: (a) iteracji #*i* ($i = 0, 1, \ldots, 2r$) funkcji generowania kluczy rundowych, (b) elementu *KS #j* ($j = 1, 2, \ldots, t$) 162
Rys. 4.9. PP1 – wersja 5 algorytmu generowania kluczy rundowych złożona z pozytywnych (prostych) modyfikacji, struktura: (a) iteracji #*i* ($i = 0, 1, \ldots, 2r$) funkcji generowania kluczy rundowych, (b) elementu *KS #j* ($j = 1, 2, \ldots, t$) 163

Rys. 4.10. PP-1 – wersja 6 algorytmu generowania kluczy rundowych złożona z operacji *RR*8, XOR i 6 S-bloków, struktura: (a) iteracji #*i* (*i* = 0, 1, …, 2*r*) funkcji generowania kluczy rundowych, (b) elementu *KS* #*j* (*j* = 1, 2, …, *t*) 164
Rys. 4.11. PP-1 – wersja 7 algorytmu generowania kluczy rundowych złożona z mnożeniem, operacją XOR i bez rotacji *RR*, struktura: (a) iteracji #*i* (*i* = 0, 1, …, 2*r*) funkcji generowania kluczy rundowych, (b) elementu *KS* #*j* (*j* = 1, 2, …, *t*) 165
Rys. 4.12. Rozszerzony algorytm generowania kluczy rundowych szyfru: (a) PP-1 64/128, (b) PP-2 64/128, obliczający 15642 klucze rundowe 169
Rys. 4.13. Histogram testu częstości {frequency} 180
Rys. 4.14. Histogram testu częstości w blokach {block-frequency (*M* = 64)} 181
Rys. 4.15. Histogram testu kumulowanych sum (wprzód) {cumulative-sums (forward)} 181
Rys. 4.16. Histogram testu kumulowanych sum (wstecz) {cumulative-sums (backward)} 182
Rys. 4.17. Histogram testu serii {runs} 182
Rys. 4.18. Dendryt wrocławski 1-stopnia skupień jednorodnych dla 128 wariantów algorytmu generowania kluczy rundowych szyfru IDEA 186
Rys. 4.19. Skupienie zawierające hipotetycznie najlepszy obiekt 186
Rys. 4.20. Osiem obiektów najbliższych obiektowi $O_{128(\mathrm{hip})}$ 187
Rys. 4.21. Dendryt wrocławski 1-stopnia skupień jednorodnych dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64, spełniających kryterium losowości, z uwzględnieniem czasu obliczeń 191
Rys. 4.22. Dendryt wrocławski 1-stopnia skupień jednorodnych dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64, spełniających kryterium losowości, z pominięciem czasu obliczeń 195
Rys. 5.1. Uporządkowanie algorytmów generowania kluczy rundowych szyfrów DES, IDEA, KASUMI, PP-1, PP-2, według teoretycznego kryterium jakości, potwierdzone testami NIST SP800-22 (wniosek 4.1) 200

Spis tabel

Tab. 3.1. DES – tablica permutacji – operacja *PC-1* 36
Tab. 3.2. DES – tablica rotacji – operacja *left shift* 36
Tab. 3.3. DES – tablica permutacji – operacja *PC-2* 37
Tab. 3.4. DES – zestawienie cech szyfru 37
Tab. 3.5. DES – lista operacji 37
Tab. 3.6. 3DES – zestawienie cech szyfru 41

Tab. 3.7. LOKI97 – tablica permutacji *P* 45
Tab. 3.8. LOKI97 – zestawienie cech szyfru 46
Tab. 3.9. LOKI97 – lista operacji 46
Tab. 3.10. SM4 – wartości stałych FK_j oraz CK_i 51
Tab. 3.11. SM4 – zestawienie cech szyfru 51
Tab. 3.12. SM4 – lista operacji 51
Tab. 3.13. KASUMI – tablica stałych *Cj* (*j* = 1, 2, …, 8) 56
Tab. 3.14. KASUMI – tablica (funkcja) *TAB* wyznaczania kluczy rundowych rundy *i* (*i* = 1, 2, …, 8) 56
Tab. 3.15. KASUMI – zestawienie cech szyfru 57
Tab. 3.16. KASUMI – lista operacji 57
Tab. 3.17. FeW – zestawienie cech szyfru 61
Tab. 3.18. FeW – lista operacji 62
Tab. 3.19. LCB-IoT – tablica permutacji *P1* 66
Tab. 3.20. LCB-IoT – zestawienie cech szyfru 66
Tab. 3.21. LCB-IoT – lista operacji 66
Tab. 3.22. Rijndael – liczba rund *Nr* szyfru i parametry *Nb, Nk* 69
Tab. 3.23. AES – wartości parametrów *Nb, Nk, Nr* 70
Tab. 3.24. AES – tablica stałych $Rcon_i$ dla *i* = 1, 2, …, 10 73
Tab. 3.25. AES – zestawienie cech szyfru 74
Tab. 3.26. AES – lista operacji 74
Tab. 3.27. Serpent – zestawienie cech szyfru 80
Tab. 3.28. Serpent – lista operacji 80
Tab. 3.29. SAFER+ – zestawienie cech szyfru 86
Tab. 3.30. SAFER+ – lista operacji 86
Tab. 3.31. PRESENT – zestawienie cech szyfru 91
Tab. 3.32. PRESENT – lista operacji 91
Tab. 3.33. PP-1 – liczba rund *r* dla bloku o rozmiarze *n* bitów 94
Tab. 3.34. PP-1 – zestawienie cech szyfru w postaci liczbowej 97
Tab. 3.35. PP-1 – zestawienie cech szyfru w postaci ogólnej 98
Tab. 3.36. PP-1 – lista operacji 98
Tab. 3.37. PP-2 – liczba rund *r* dla bloku o rozmiarze *n* bitów i klucza o długości |*k*| bitów 101
Tab. 3.38. PP-2 – permutacja bitowa *P* oraz permutacja do niej odwrotna P^{-1} (*n* = 64) 102

Tab. 3.39. PP-2 – zestawienie cech szyfru 105
Tab. 3.40. PP-2 – zestawienie cech szyfru 105
Tab. 3.41. PP-2 – lista operacji 105
Tab. 3.42. IDEA – zestawienie cech szyfru 111
Tab. 3.43. IDEA – lista operacji 111
Tab. 3.44. RC6-32/20/*b* – zestawienie cech szyfru 117
Tab. 3.45. RC6 – lista operacji 117
Tab. 3.46. Speck – parametry zależne od wariantu szyfru 120
Tab. 3.47. Speck – zestawienie cech szyfru 122
Tab. 3.48. Speck – lista operacji 122
Tab. 3.49. Zestawienie cech algorytmów generowania kluczy rundowych w szyfrach o konstrukcji sieci Feistela (*n* – rozmiar bloku, *r* – liczba rund) 126
Tab. 3.50. Zestawienie cech algorytmów generowania kluczy rundowych w szyfrach o konstrukcji SPN lub innej (*n* – rozmiar bloku, *r* – liczba rund) 127
Tab. 4.1. Przypadki występowania błędów I i II rodzaju [95] 137
Tab. 4.2. Zestawienie wyników *p* testów NIST SP800-22 przy zastosowaniu metody A (PA = <0.9806; 0.9994>) 155
Tab. 4.3. Podsumowanie testów NIST 800-22 dla różnych wersji algorytmu generowania kluczy rundowych w szyfrze PP-1 64/64 167
Tab. 4.4. Przykładowe klucze rundowe z rozszerzonego algorytmu generowania kluczy rundowych szyfru PP-2 64/128 169
Tab. 4.5. Zestawienie wyników *p* testów NIST SP800-22 dla rozszerzonych algorytmów generowania kluczy rundowych PP-1 64/128 oraz PP-2 64/128 (PA = <0.9806; 0.9994>) .. 170
Tab. 4.6. Zestawienie wyników *p* testów NIST SP800-22 dla algorytmu generowania kluczy rundowych szyfru DES i złożonych próbek (metapróbek) o długości 1000704 bitów (PA = <0.9806; 0.9994>) 172
Tab. 4.7. Zestawienie wyników *p* testów NIST SP800-22 dla algorytmu generowania kluczy rundowych szyfru PP-1 64/128 i złożonych próbek (metapróbek) o długości 1001088 bitów (PA = <0.9806; 0.9994>) 173
Tab. 4.8. Zestawienie wyników *p* testów NIST SP800-22 dla algorytmu generowania kluczy rundowych szyfru PP-2 64/128 i złożonych próbek (metapróbek) o długości 1000896 bitów (PA = <0.9806; 0.9994>) 174

Tab. 4.9. Zestawienie wyników *p* testów NIST SP800-22 dla algorytmów generowania kluczy rundowych szyfrów DES, IDEA, KASUMI PP-1 64/128, PP-2 64/128 i skróconych próbek (podpróbek) (PA = <0.9806; 0.9994>).....176
Tab. 4.10. Zestawienie wyników *p* testów NIST SP800-22 dla algorytmów generowania kluczy rundowych szyfrów DES, IDEA, KASUMI PP-1 64/128, PP-2 64/128 i złożenia 4 podpróbek (PA = <0.9806; 0.9994>) 177
Tab. 4.11. Skrócona tabela wyników *p* testów NIST SP800-22 dla różnych wariantów rotacji w algorytmie generowania kluczy rundowych szyfru IDEA (PA = <0.9806; 0.9994>).....179
Tab. 4.12. Skrócona tabela standaryzowanych wartości cech badanych obiektów dla algorytmu generowania kluczy rundowych szyfru IDEA 183
Tab. 4.13. Skrócona macierz (tablica) odległości D_v dla 128 wariantów algorytmu generowania kluczy rundowych szyfru IDEA.....184
Tab. 4.14. Porównanie wyników testów NIST SP800-22 dla rotacji oryginalnej o 25 bitów i optymalnej o 71 bitów (PA = <0.9806; 0.9994>) 187
Tab. 4.15. Wyniki testów NIST SP800-22 dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64 spełniających kryterium losowości, z uwzględnieniem czasu obliczeń (PA = <0.9806; 0.9994>).....188
Tab. 4.16. Standaryzowane wartości cech badanych obiektów dla algorytmu generowania kluczy rundowych szyfru PP-1 64/64, z uwzględnieniem czasu obliczeń.....189
Tab. 4.17. Macierz (tablica) odległości D_v dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64, z uwzględnieniem czasu obliczeń.....190
Tab. 4.18. Porównanie wyników testów NIST SP800-22 dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64 oryginalnego i optymalnego, z uwzględnieniem czasu obliczeń (PA = <0.9806; 0.9994>).....191
Tab. 4.19. Wyniki testów NIST SP800-22 dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64, spełniających kryterium losowości z pominięciem czasu obliczeń (PA = <0.9806; 0.9994>).....193
Tab. 4.20. Standaryzowane wartości cech badanych obiektów dla algorytmu generowania kluczy rundowych szyfru PP-1 64/64, z pominięciem czasu obliczeń.....193
Tab. 4.21. Macierz (tablica) odległości D_v dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64, z pominięciem czasu obliczeń 194
Tab. 4.22. Porównanie wyników testów NIST SP800-22 dla wariantów algorytmu generowania kluczy rundowych szyfru PP-1 64/64, oryginalnego i optymalnego, z pominięciem czasu obliczeń (PA = <0.9806; 0.9994>) 195

Tab. 6.1. Kompletna tabela wyników *p* testów NIST SP800-22 dla różnych wariantów rotacji w algorytmie generowania kluczy rundowych szyfru IDEA (PA = <0.9806; 0.9994>).....204
Tab. 6.2. Kompletna tabela wartości standaryzowanych cech badanych obiektów w algorytmie IDEA 206

Spis algorytmów

Algorytm 2.1. Zapewnienie poufności z wykorzystaniem kryptografii symetrycznej 16
Algorytm 2.2. Zapewnienie poufności z wykorzystaniem kryptografii asymetrycznej 20
Algorytm 2.3. Realizacja uwierzytelnienia nadawcy i zapewnienia niezaprzeczalności w kryptografii asymetrycznej 21
Algorytm 2.4. Weryfikacja integralności danych 23
Algorytm 2.5. Generowanie podpisu przez użytkownika A 24
Algorytm 2.6. Weryfikacja podpisu i integralności danych przez użytkownika B 25
Algorytm 3.1. Algorytm generowania kluczy rundowych ($R = r$) 33
Algorytm 3.2. Algorytm generowania kluczy rundowych szyfru DES 36
Algorytm 3.3. Algorytm generowania kluczy rundowych szyfru LOKI97 45
Algorytm 3.4. Algorytm generowania kluczy rundowych szyfru SM4 50
Algorytm 3.5. Algorytm generowania kluczy rundowych szyfru KASUMI 56
Algorytm 3.6. Algorytm generowania kluczy rundowych szyfru FeW 61
Algorytm 3.7. Algorytm generowania kluczy rundowych szyfru LCB-IoT 66
Algorytm 3.8. Algorytm generowania kluczy rundowych szyfru AES 73
Algorytm 3.9. Algorytm generowania kluczy rundowych szyfru Serpent 79
Algorytm 3.10. Algorytm generowania kluczy rundowych szyfru SAFER+ z kluczem 128-bitowym (SAFER+ 128/128) 85
Algorytm 3.11. Algorytm generowania kluczy rundowych szyfru PRESENT 90
Algorytm 3.12. Algorytm generowania kluczy rundowych szyfru PP-1 96
Algorytm 3.13. Algorytm generowania kluczy rundowych szyfru PP-2 103
Algorytm 3.14. Algorytm generowania kluczy rundowych szyfru IDEA 111
Algorytm 3.15. Algorytm generowania kluczy rundowych szyfru RC6-*w*/*r*/*b* 116
Algorytm 3.16. Algorytm generowania kluczy rundowych szyfru Speck 122
Algorytm 4.1. Algorytm konstruowania dendrytu 178

Literatura

[1] Anderson R. J., Biham, E., Knudsen L. R., „*Serpent: A Proposal for the Advanced Encryption Standard"*, http://www.cl.cam.ac.uk/~rja14/serpent.html, **1998**.

[2] Albrecht M.R., Farshim P., Paterson K.G., Watson G.J., „*On Cipher-Dependent Related-Key Attacks in the Ideal-Cipher Model*", In: Joux A. (eds) Fast Software Encryption FSE 2011, Lecture Notes in Computer Science, vol. 6733, Springer, Berlin, Heidelberg, **2011**.

[3] **Apolinarski M.**, „*Ocena jakości algorytmów generowania kluczy rundowych w wybranych szyfrach blokowych*", Studia z Automatyki i Informatyki – tom 37, Poznań, **2012**.

[4] **Apolinarski M.**, „*Statistical Properties Analysis of Key Schedule Modification in Block Cipher PP-1*", A. Wiliński et al. (eds.), Soft Computing in Computer and Information Science, Advances in Intelligent Systems and Computing, vol. 342, pp. 257-268, Springer International Publishing Switzerland, **2015**.

[5] **Apolinarski M.**, „*IDEA key schedule evaluation using cluster analysis*", Przegląd elektrotechniczny, R. 92, nr 12, pp. 224-227, Warszawa, **2016**.

[6] **Apolinarski M.,** „*Randomness Evaluation of PP-1 and PP-2 Block Ciphers Round Keys Generators*", In: Pejaś J., El Fray I., Hyla T., Kacprzyk J. (eds) Advances in Soft and Hard Computing, ACS 2018, Advances in Intelligent Systems and Computing, vol. 889, pp. 272-281, DOI: https://doi.org/10.1007/978-3-030-03314-9_24, Springer, Cham, **2019**.

[7] Bar-On, A., Dunkelman, O., Keller, N., Weizman, A., „*DLCT: A new tool for differential-linear cryptanalysis*", In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Proceedings, Part I. LNCS, vol. 11476, pp. 313–342, Springer, **2019**.

[8] Barker, E., Roginsky, „*A. Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths*", National Institute of Standards and Technology Special Publication 800, 131A, **2015**.

[9] Barker E., Mouha N., „*Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*", Special Publication (NIST SP 800-67 rev. 2), National Institute of Standards and Technology, DOI: https://doi.org/10.6028/NIST.SP.800-67r2, Gaithersburg, MD, **2017**.

[10] Barker W., Barker E., „*Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*”, Special Publication (NIST SP 800-67 ver. 1), DOI: https://doi.org/10.6028/NIST.SP.800-67ver1, National Institute of Standards and Technology, Gaithersburg, MD, **2004**.

[11] Bertoni G., Daemen J., Peeters M., Van Assche G., „*Keccak*”, In: Johansson T., Nguyen P.Q. (eds) Advances in Cryptology – EUROCRYPT 2013, Lecture Notes in Computer Science, vol. 7881, pp. 313-314, DOI: https://doi.org/10.1007/978-3-642-38348-9_19, Springer, Berlin, Heidelberg, **2013**.

[12] Beaulieu R., Shors D., Smith J., Treatman-Clark S., Weeks B., Wingers L., „*The SIMON and SPECK families of block ciphers*”, NSA, USA, **2013**.

[13] Beaulieu R., Shors D., Smith J., Treatman-Clark S., Weeks B., Wingers L., „*The SIMON and SPECK lightweight block ciphers*”, In Proceedings of the 52nd Annual Design Automation Conference (DAC '15), Association for Computing Machinery, Article 175, pp. 1-6, DOI: https://doi.org/10.1145/2744769.2747946, USA, **2015**.

[14] Bhargavan K., Leurent G., „*On the Practical (In-)Security of 64-bit Block Ciphers: Collision Attacks on HTTP over TLS and OpenVPN*”, ACM CCS 2016 - 23rd ACM Conference on Computer and Communications Security, pp. 456-467, Vienna, Austria, **2016**.

[15] Biham E., Shamir A., „*Differential cryptanalysis of DES-like cryptosystems*”, Journal of Cryptology, vol. 4, pp. 3-72, DOI: https://doi.org/10.1007/BF00630563, **1991**.

[16] Biham E., Shamir A., „*Differential Cryptanalysis of the Data Encryption Standard*”, DOI: https://doi.org/10.1007/978-1-4613-9314-6, Springer, New York, **1993**.

[17] Biham E., „*New Types of Cryptoanalytic Attacks Using Related Keys*”, Journal of Cryptology, vol. 7, pp. 229-246, DOI: https://doi.org/10.1007/BF00203965, **1994**.

[18] Biham E., Biryukov A., Shamir A., „*Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials*”, In: Stern J. (eds) Advances in Cryptology — EUROCRYPT ’99, Lecture Notes in Computer Science, vol. 1592, pp. 12–23, DOI: https://doi.org/10.1007/3-540-48910-X_2, Springer, Berlin, Heidelberg, **1999**.

[19] Biham E., Biryukov A., Shamir A., „*Miss in the middle attacks on IDEA and Khufu*” *Fast Software Encryption, FSE'99*, Lecture Notes in Computer Science, vol. 1636, pp. 124–138, ed. L. R. Knudsen, Springer-Verlag, Berlin, **1999**.

[20] Biham E., Dunkelman O., Keller N., „*The Rectangle Attack — Rectangling the Serpent*”, In: Pfitzmann B. (eds) Advances in Cryptology — EUROCRYPT 2001, Lecture Notes in Computer Science, vol. 2045, pp. 340-357, DOI: https://doi.org/10.1007/3-540-44987-6_21, Springer, Berlin, Heidelberg, **2001**.

[21] Biham E., Dunkelman, O., Keller N., „*Linear cryptanalysis of reduced round serpent*”, In: Matsui, M. (ed.) FSE 2001, Lecture Notes in Computer Science, vol. 2355, pp. 16-27, Springer, Heidelberg, **2002**.

[22] Biham E., Dunkelman O., Keller N., „*A Related-Key Rectangle Attack on the Full KASUMI*.”, In: Roy B. (eds) Advances in Cryptology – ASIACRYPT 2005, Lecture Notes in Computer Science, vol. 3788, pp. 443-461, DOI: https://doi.org/10.1007/11593447_24, Springer, Berlin, Heidelberg, **2005**.

[23] Biham E., Dunkelman O., Keller N., „*Related-Key Impossible Differential Attacks on 8-Round AES-192*”, In: Pointcheval D. (eds) Topics in Cryptology – CT-RSA 2006. CT-RSA 2006, Lecture Notes in Computer Science, vol 3860, pp. 21-33, DOI: https://doi.org/10.1007/11605805_2, Springer, Berlin, Heidelberg, **2006**.

[24] Biryukov A., Wagner D., „*Slide attacks*”, In Proceedings of Fast Software Encryption – FSE’99, Lecture Notes in Computer Science, vol. 1636, pp. 245–259, Springer-Verlag, **1999**.

[25] Biryukov A., Wagner D., „*Advanced Slide Attacks*”, EUROCRYPT 2000, Lecture Notes in Computer Science, vol. 1807, pp. 589-606, Springer, **2000**.

[26] Biryukov A., „*Block ciphers and stream ciphers: The state of the art*”, IACR Cryptology ePrint Archive, vol. 2004, p. 94, **2004**.

[27] Biryukov A., „*Impossible Differential Attack*”, In: van Tilborg H.C.A. (eds) Encyclopedia of Cryptography and Security, DOI: https://doi.org/10.1007/0-387-23483-7_197, Springer, Boston, MA, **2005**

[28] Biryukov A., Khovratovich D., Nikolić I., „*Distinguisher and Related-Key Attack on the Full AES-256*”, In: Halevi S. (eds) Advances in Cryptology - CRYPTO 2009. CRYPTO 2009, Lecture Notes in Computer Science, vol. 5677, pp. 231-249, DOI: https://doi.org/10.1007/978-3-642-03356-8_14, Springer, Berlin, Heidelberg, **2009**.

[29] Biryukov A., Khovratovich D., „*Related-Key Cryptanalysis of the Full AES-192 and AES-256*”, In: Matsui M. (eds) Advances in Cryptology – ASIACRYPT 2009, Lecture

Notes in Computer Science, vol. 5912, pp. 1-18, DOI: https://doi.org/10.1007/978-3-642-10366-7_1, Springer, Berlin, Heidelberg, **2009**.

[30] Biryukov A., Dunkelman O., Keller N., Khovratovich D., Shamir A., „*Key Recovery Attacks of Practical Complexity on AES-256 Variants with up to 10 Rounds*", In: Gilbert H. (eds) Advances in Cryptology – EUROCRYPT 2010, Lecture Notes in Computer Science, vol. 6110, pp. pp 299-319, DOI: https://doi.org/10.1007/978-3-642-13190-5_15, Springer, Berlin, Heidelberg, **2010**.

[31] Biryukov, A., Nikolić, I., „*Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others*", Gilbert, H. (ed.) EUROCRYPT 2010, Lecture Notes in Computer Science vol. 6110, pp. 322–344, DOI: https://doi.org/10.1007/978-3-642-13190-5_17, Springer, Heidelberg, **2010**.

[32] Biryukov A., Perrin L., „*State of the Art in Lightweight Symmetric Cryptography*", IACR Cryptology ePrint Archive, vol. 511, **2017**.

[33] Blondeau, C., Leander, G., Nyberg, K. „*Differential-Linear Cryptanalysis Revisited*", Journal of Cryptology, vol. 30, pp. 859–888, DOI: https://doi.org/10.1007/s00145-016-9237-5, **2017**.

[34] Bogdanov A., Knudsen L.R., Leander G., Paar C., Poschmann A., Robshaw M.J.B., Seurin Y., Vikkelsoe C. „*PRESENT: An Ultra-Lightweight Block Cipher*", In: Paillier P., Verbauwhede I. (eds) Cryptographic Hardware and Embedded Systems - CHES 2007, Lecture Notes in Computer Science, vol. 4727, pp. 450-466, DOI: https://doi.org/10.1007/978-3-540-74735-2_31, Springer, Berlin, Heidelberg, **2007**.

[35] Bogdanov, A., Tischhauser, E., „*On the Wrong Key Randomisation and Key Equivalence Hypotheses in Matsui's Algorithm 2*", Moriai, S. (ed.) FSE 2013, Lecture Notes in Computer Science, vol. 8424, pp. 19–38. Springer, Heidelberg, **2014**.

[36] Brown L., Pieprzyk J., „*Introducing the new LOKI97 Block Cipher*", **1998**.

[37] Brown R. G., „*Dieharder: A random number test suite, version 3.31.1*", Duke University Physics Department, **2014**.

[38] Bucholc K., Chmiel K., Grocholewska-Czuryło A., Idzikowska E., Janicka-Lipska I., Stokłosa J., „*Scalable PP-1 Block Cipher"*, International Journal of Applied

Mathematics and Computer Science, vol. 20, No. 2, pp. 401–411, DOI: https://doi.org/10.2478/v10006-010-0030-6, **2010**.

[39] Bucholc, K., Chmiel, K., Grocholewska-Czuryło, A., Stokłosa, J., „*PP-2 Block Cipher*", 7th International Conference on Emerging Security Information Systems and Technologies (SECURWARE 2013), pp. 162–168, XPS Press, Wilmington, **2013**.

[40] Carter G., Dawson E., Nielsen L. „*Key schedules of iterative block ciphers*", In: Boyd C., Dawson E. (eds) Information Security and Privacy, ACISP 1998. Lecture Notes in Computer Science, vol. 1438, pp. 80-89, DOI: https://doi.org/10.1007/BFb0053723, Springer, Berlin, Heidelberg., **1998**.

[41] Carter, G., Dawson, E., and Nielsen, L., „*Key Schedule Classification of the AES Candidates*", In Proceedings of the end AES Conference, Rome, Italy, pp. 1-14, **1999**.

[42] Chmiel, K., Grocholewska-Czuryło, A., Stokłosa, J., „*Involutional Block Cipher for Limited Resources*", IEEE Global Telecommunications Conference (GLOBECOM) 2008, IEEE Press, **2008**.

[43] Chmiel, K., „*Metody różnicowej i liniowej kryptoanalizy szyfrów blokowych"*, Rozprawy nr 443, Wydawnictwo Politechniki Poznańskiej, Poznań, **2010**.

[44] Cho J.Y., „*Linear Cryptanalysis of Reduced-Round PRESENT*", Topics in Cryptology – CT-RSA 2010, Lecture Notes in Computer Science, vol. 5985, pp. 302-317, DOI: https://doi.org/10.1007/978-3-642-11925-5_21, Springer, Berlin, Heidelberg, **2010**.

[45] Collard, B., Standaert, F., „*A Statistical Saturation Attack against the Block Cipher PRESENT*", Fischlin, M. (ed.) CT-RSA 2009. Lecture Notes in Computer Science, vol. 5473, pp. 195–210, DOI: https://doi.org/10.1007/978-3-642-00862-7_13, Springer, Heidelberg, **2009**.

[46] Colburn M., Keliher L., „*Linear Cryptanalysis of the PP-1 and PP-2 Block Ciphers*", In: Meier W., Mukhopadhyay D. (eds) Progress in Cryptology – INDOCRYPT 2014, Lecture Notes in Computer Science, vol. 8885, pp. 107-123, DOI: https://doi.org/10.1007/978-3-319-13039-2_7, Springer, Cham, **2014**.

[47] Daemen J., Rijmen V., „*AES Proposal: Rijndael*", AES Algorithm Submission, September 3, **1999**.

[48] Daemen J., Rijmen V., „*The design of Rijndael: AES – the Advanced Encryption Standard*", Springer-Verlag, Berlin, **2002**.

[49] „*Data Encryption Standard*", Federal Information Processing Standards Publication 46-3, National Institute of Standards and Technology, Springfield, VA, October **1999**.

[50] Diffie W., Ledin G. (translators), „*SMS4 Encryption Algorithm for Wireless Networks"*, Cryptology ePrint Archive http://eprint.iacr.org/2008/329.pdf, **2008**.

[51] Dunkelman, O., Indesteege, S., Keller, N.: „*A differential-linear attack on 12-round Serpent*", In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT, Lecture Notes in Computer Science, vol. 5365, pp. 308–321, Springer, **2008**.

[52] Dworkin M., Barker E., Nechvatal J., Foti J., Bassham L., Roback E., Dray J., „*Advanced Encryption Standard (AES)*", Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, DOI: https://doi.org/10.6028/NIST.FIPS.197, Gaithersburg, MD, **2001**.

[53] Dworkin, M., „*SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*", Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, DOI: https://doi.org/10.6028/NIST.FIPS.202, Gaithersburg, MD, **2015**.

[54] Ferguson N., Kelsey J., Lucks S., Schneier B., Stay M., Wagner D., Whiting D., „*Improved Cryptanalysis of Rijndael*", FSE 2000, Lecture Notes in Computer Science, vol. 1978, pp. 213-230, DOI: https://doi.org/10.1007/3-540-44706-7_15, Springer, Berlin, Heidelberg, **2000**.

[55] Florek K., Łukaszewicz J., Perkal J., Steinhaus H., Zubrzycki S.: „*Taksonomia wrocławska*", Przegląd Antropologiczny, nr 17, **1956**.

[56] Gorski, M., Lucks, S., „*New related-key boomerang attacks on AES*", In Chowdhury, D.R. Rijmen, V., Das, A., eds.: INDOCRYPT, Lecture Notes in Computer Science, vol. 5356, pp. 266-278, Springer, **2008**.

[57] Hawkes P., „*Differential-Linear Weak Key Classes of IDEA*", EUROCRYPT '98, pp. 112-126, **1998**.

[58] Hong D., Lee JK., Kim DC., Kwon D., Ryu K.H., Lee DG., „*LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors*", In: Kim Y., Lee H., Perrig A. (eds) Information Security Applications, WISA 2013, Lecture Notes in Computer Science, vol. 8267, pp. 3-27, DOI: https://doi.org/10.1007/978-3-319-05149-9_1, Springer, Cham, **2013**.

[59] Hong S., Kim J., Lee S., Preneel B., „*Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192*", In Henri Gilbert and Helena Handschuh, editors, FSE, Lecture Notes in Computer Science, vol. 3557, pp. 368-383, Springer, **2005**.

[60] Huang, J., Lai, X., „*Revisiting Key Schedule's Diffusion in Relation with Round Function's Diffusion*", Designs, Codes and Cryptography, vol. 73-1, pp. 1–19, DOI: https://doi.org/10.1007/s10623-013-9804-9, **2014**.

[61] Huang J., Vaudenay S., Lai X., „*On the Key Schedule of Lightweight Block Ciphers*", In: Meier W., Mukhopadhyay D. (eds) Progress in Cryptology -- INDOCRYPT 2014. INDOCRYPT 2014, Lecture Notes in Computer Science, vol. 8885, pp. 124-142, DOI: https://doi.org/10.1007/978-3-319-13039-2_8, Springer, Cham, **2014**.

[62] ISO/IEC JTC 1/SC 27, „*ISO/IEC 29192-2:2012, Information security - Lightweight cryptography - Part 2: Block ciphers*" (withdrawn), https://www.iso.org/standard/56552.html, Geneva, Switzerland, **2012**.

[63] ISO/IEC JTC 1/SC 27, „*ISO/IEC 29192-2:2019, Information security - Lightweight cryptography - Part 2: Block ciphers*", https://www.iso.org/standard/78477.html, Geneva Switzerland, **2019**.

[64] Jakimoski G., Desmedt Y., „*Related-Key Differential Cryptanalysis of 192-bit Key AES Variants*", In Mitsuru Matsui and Robert J. Zuccherato, editors, Selected Areas in Cryptography, Lecture Notes in Computer Science, vol. 3006, pp. 208-221, Springer, **2003**.

[65] Jia K., Li L., Rechberger C., Chen J., Wang X., „*Improved Cryptanalysis of the Block Cipher KASUMI*", In: Knudsen L.R., Wu H. (eds) Selected Areas in Cryptography, SAC 2012, Lecture Notes in Computer Science, vol. 7707, pp. 222-233, DOI: https://doi.org/10.1007/978-3-642-35999-6_15, Springer, Berlin, Heidelberg, **2013**.

[66] Kelsey J., Schneier B., Wagner D., „*Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES*", Advances in Cryptology, CRYPTO '96 Proceedings, Springer-Verlag, pp. 237-251, **1996**.

[67] Kelsey J., Schneier B., Wagner D., „*Key Schedule Weaknesses in SAFER+*", The Second AES Candidate Conference, NIST **1999**.

[68] Kim T.H., Kim J., Hong S., Sung, J. „*Linear and Differential Cryptanalysis of Reduced SMS4 Block Cipher*", IACR Cryptol. ePrint Arch. 281, **2008**.

[69] Kim J., Hong S., Preneel B., Biham E., Dunkelman O., Keller N., „*Related-Key Boomerang and Rectangle Attacks*", IACR, 2010/019, http://eprint.iacr.org/2010/019, **2010**.

[70] Knudsen L.R., Meier W., „*Correlations in RC6* with a Reduced Number of Rounds", In: Goos G., Hartmanis J., van Leeuwen J., Schneier B. (eds) Fast Software Encryption, FSE 2000, Lecture Notes in Computer Science, vol. 1978, pp. 94-108, DOI: https://doi.org/10.1007/3-540-44706-7_7, Springer, Berlin, Heidelberg, **2001**.

[71] Knudsen L.R., Mathiassen J.E., „*On the Role of Key Schedules in Attacks on Iterated Ciphers*", In: Samarati P., Ryan P., Gollmann D., Molva R. (eds) Computer Security – ESORICS 2004, Lecture Notes in Computer Science, vol 3193, pp. 322-334, DOI: https://doi.org/10.1007/978-3-540-30108-0_20, Springer, Berlin, Heidelberg, **2004**.

[72] Kranz, T., Leander, G., & Wiemer, F., „*Linear Cryptanalysis: Key Schedules and Tweakable Block Ciphers*", IACR Transactions on Symmetric Cryptology, vol. 2017-1, pp. 474–505, DOI: https://doi.org/10.13154/tosc.v2017.i1.474-505, **2017**.

[73] Kumar M., Pal S.K., Panigrahi A., „*FeW: a lightweight block cipher*", Turkish Journal of Mathematics and Computer Science, vol. 11-2, pp. 58–73, **2014**.

[74] Lai X., Massey J.L., „*A Proposal for a New Block Encryption Standard*", In: Damgård I.B. (eds) Advances in Cryptology — EUROCRYPT '90, Lecture Notes in Computer Science, vol. 473, pp. 389-404, DOI: https://doi.org/10.1007/3-540-46877-3_35, Springer, Berlin, Heidelberg, **1991**.

[75] Langford S.K., Hellman M.E., „*Differential-Linear Cryptanalysis*", In: Desmedt Y.G. (eds) Advances in Cryptology — CRYPTO '94, Lecture Notes in Computer Science, vol. 839, pp. 17-25, DOI: https://doi.org/10.1007/3-540-48658-5_3, Springer, Berlin, Heidelberg, **1994**.

[76] Lee, Y., Jeong, K., Sung, J., Lee, C., Hong, S., Chang, K.-Y., „*Security Analysis of Scalable Block Cipher PP-1 Applicable to Distributed Sensor Networks*", Int. J. Distr. Sens. Net. 2013, 1–9, **2013**.

[77] Liu F., Ji W., Hu L., Ding J., Lv S., Pyshkin A., Weinmann R., „*Analysis of the SMS4 Block Cipher*", In: Pieprzyk J., Ghodosi H., Dawson E. (eds) Information Security and Privacy. ACISP 2007. Lecture Notes in Computer Science, vol. 4586, pp. 158-170,

DOI: https://doi.org/10.1007/978-3-540-73458-1_13, Springer, Berlin, Heidelberg, **2007**.

[78] Liu Y., Liang H., Wang W., Wang M., „*New Linear Cryptanalysis of Chinese Commercial Block Cipher Standard SM4*", Security and Communication Networks vol. 2017, Article ID 1461520, DOI: https://doi.org/10.1155/2017/1461520, **2017**.

[79] Lucks S., „*Ciphers Secure against Related-Key Attacks*", International Workshop on Fast Software Encryption FSE 2004: Fast Software Encryption, pp. 359-370, **2004**.

[80] Matsui M., „*Linear Cryptanalysis Method for DES Cipher*", In: Helleseth T. (eds) Advances in Cryptology — EUROCRYPT '93, Lecture Notes in Computer Science, vol. 765, pp. 386-397, DOI: https://doi.org/10.1007/3-540-48285-7_33, Springer, Berlin, Heidelberg, **1994**.

[81] Matsui M., „*The first experimental cryptanalysis of the Data Encryption Standard*", In Y. G. Desmedt, editor, Advances in Cryptology - Proc. Crypto'94, Lecture Notes in Computer Science vol. 839, pp. 1–11, Springer Verlag, **1994**.

[82] Marsaglia G., „*DIEHARD Statistical Tests*", **1995**.

[83] Massey J. L., Khachatrian G. H., Kuregian M. K., „*Nomination of SAFER+ as Candidate Algorithm for the Advanced Encryption Standard (AES)*", NIST AES Proposal, **1998**.

[84] May L., Henricksen M., Millan W., Carter G., Dawson E., „*Strengthening the Key Schedule of the AES*", ACISP 2002, Lecture Notes in Computer Science, vol. 2384, pp. 226-240, Springer-Verlag, **2002**.

[85] Menezes A. J., Oorschot P. C., Vanstone S. A., „*Kryptografia stosowana*", WNT, Warszawa, **2005**.

[86] Misztal M., „*Differential Cryptanalysis of PP-1 Cipher*", Annales UMCS Informatica AI XI 2, pp. 9–24, **2011**.

[87] Misztal M., Courtois N. T., „*Aggregated Differentials and Cryptanalysis of PP-1 and GOST"*, Periodica Mathematica Hungarica, vol. 65-2, pp. 177-192, DOI: https://doi.org/10.1007/s10998-012-2983-8, **2012**.

[88] Nguyen P. H., Wu H., Wang H., „*Improving the Algorithm 2 in Multidimensional Linear Cryptanalysis*", In: Parampalli U., Hawkes P. (eds) Information Security and

Privacy, ACISP 2011, Lecture Notes in Computer Science, vol. 6812, pp. 61-74, DOI: https://doi.org/10.1007/978-3-642-22497-3_5, Springer, Berlin, Heidelberg, **2011**.

[89] „*Universal Mobile Telecommunications System (UMTS); Specification of the 3GPP confidentiality and integrity algorithms; Document 2: Kasumi specification (3GPP TS 35.202 version 7.0.0 Release 7)*", ETSI TS 135 202 V7.0.0, **2007**.

[90] Ohkuma K., „*Weak Keys of Reduced-Round PRESENT for Linear Cryptanalysis*", In: Jacobson M.J., Rijmen V., Safavi-Naini R. (eds) Selected Areas in Cryptography, Lecture Notes in Computer Science, vol. 5867, pp. 249-265, DOI: https://doi.org/10.1007/978-3-642-05445-7_16, Springer, Berlin, Heidelberg, **2009**.

[91] Ozen O., Varici K., Tezcan C., Kocair C., „*Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT*", In Boyd, Gonzalez-Benito (Eds.) Proceedings of the 2009 Australasian (ACISP) Conference, pp. 90-107, July 1-3, 2009. Lecture Notes in Computer Science, vol. 5594, **2009**.

[92] Ramadan R. A., Aboshosha B. W., Yadav K., Alseadoon I. M., Kashout M. J., et al. „*LBC-IoT: Lightweight Block Cipher for IoT Constraint Devices",* Computers Materials & Continua 67, vol. 67-3, pp. 3563-3579, DOI: https://doi.org/10.32604/cmc.2021.015519, **2021**.

[93] Pehlivanoğlu, M.K., Sakalli, M.T., Duru, N., Sakallı, F.B., „*The New Approach of AES Key Schedule for Lightweight Block Ciphers*", IOSR Journal of Computer Engineering, vol. 19, pp. 21-26, DOI: https://doi.org/10.9790/0661-1903042126, **2017**.

[94] Rivest R.L., Robshaw M.J.B., Sidney R., Yin Y.L., „*The RC6 Block Cipher*", In in First Advanced Encryption Standard (AES) Conference, pp. 16, **1998**.

[95] Rukhin A. et al. „*A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*", NIST Special Publication 800-22, rev. 1a, National Institute of Standards and Technology, Gaithersburg, MD, **2010**.

[96] Schneier B., „*Kryptografia dla praktyków*", WNT, Warszawa **2002**.

[97] Shannon C. E., „*A mathematical theory of communication*", Bell System Technical Journal, vol. 27-3, pp. 379-423, DOI: https://doi.org/10.1002/j.1538-7305.1948.tb01338.x, Nokia Bell Labs, **1948**.

[98] Shannon C. E., „*Communication Theory of Secrecy Systems*", Bell System Technical Journal, vol. 28-4, pp. 656-715, DOI: https://doi.org/10.1002/j.1538-7305.1949.tb00928.x, Nokia Bell Labs, **1949**.

[99] Shi T., Wang W. Xu Q., „*Improved Impossible Differential Cryptanalysis of SMS4*", 2012 Eighth International Conference on Computational Intelligence and Security, pp. 492-496, DOI: https://doi.org/10.1109/CIS.2012.116, **2012**.

[100] Shirai T., Shibutani K., Akishita T., Moriai S., Iwata T., „*The 128-Bit Blockcipher CLEFIA (Extended Abstract)*", In: Biryukov A. (eds) Fast Software Encryption, FSE 2007, Lecture Notes in Computer Science, vol. 4593, pp. 181-195, DOI: https://doi.org/10.1007/978-3-540-74619-5_12, Springer, Berlin, Heidelberg, **2007**.

[101] Stokłosa J., Bilski T., Pankowski T., „*Bezpieczeństwo danych w systemach informatycznych*", Wydawnictwo Naukowe PWN, Warszawa, **2001**.

[102] Stokłosa J. (red), „*Ochrona danych i zabezpieczenia w systemach teleinformatycznych*", Wydawnictwo Politechniki Poznańskiej, Poznań **2003**.

[103] Soto J., „*Randomness Testing of the Advanced Encryption Standard Candidate Algorithms"*, NIST IR 6390, DOI: https://doi.org/10.6028/NIST.IR.6390, Gaithersburg, MD, **1999**.

[104] Soto J., Bassham L., „*Randomness Testing of the Advanced Encryption Standard Finalist Candidates"*, NIST IR 6483, DOI: https://doi.org/10.6028/NIST.IR.6483, Gaithersburg, MD, **2000**.

[105] Wagner D., „*The Boomerang Attack*", In: Knudsen L. (eds) Fast Software Encryption. FSE 1999, Lecture Notes in Computer Science, vol. 1636, pp. 156-170, DOI: https://doi.org/10.1007/3-540-48519-8_12, Springer, Berlin, Heidelberg, **1999**.

[106] Walker J., „*Ent - a pseudorandom number sequence test program*", https://www.fourmilab.ch/random/, **2008**.

[107] Wang M., „*Differential cryptanalysis of reduced-round PRESENT*", In: Vaudenay S. (eds) Progress in Cryptology – AFRICACRYPT 2008, Lecture Notes in Computer Science vol. 5023, pp. 40-49, DOI: https://doi.org/10.1007/978-3-540-68164-9_4, Springer, Berlin, Heidelberg, **2008**.

[108] Zhang W., Zhang L., Wu W., Feng D., „*Related-Key Differential-Linear Attacks on Reduced AES-192*", In: Srinathan K., Rangan C.P., Yung M., (eds) Progress in

Cryptology – INDOCRYPT 2007, Lecture Notes in Computer Science, vol. 4859, pp. 73-85, DOI: https://doi.org/10.1007/978-3-540-77026-8_7, Springer, Berlin, Heidelberg, **2007**.

[109] Zhang J., Wu W., Zheng Y., „*Security of SM4 Against (Related-Key) Differential Cryptanalysis*", In: Bao F., Chen L., Deng R., Wang G., (eds) Information Security Practice and Experience, ISPEC 2016, Lecture Notes in Computer Science, vol. 10060, pp. 65-78, DOI: https://doi.org/10.1007/978-3-319-49151-6_5, Springer, Cham, **2016**.