WOJSKOWA AKADEMIA TECHNICZNA
im. Jarosława Dąbrowskiego

Rozprawa doktorska

Metody zautomatyzowanego poszukiwania charakterystyk różnicowych w odniesieniu do kryptoanalizy szyfrów blokowych

mgr inż. Władysław Dudzic

Promotor
dr hab. inż. Andrzej Paszkiewicz

Promotor pomocniczy
dr inż. Krzysztof Kanciak

Warszawa 2022

Podziękowania.

Składam serdeczne podziękowania promotorowi,
Panu dr hab. inż. Andrzeju Paszkiewiczowi za chęć prowadzenia mojej pracy od pierwszych dni studiów doktorskich, cenne rady, opiekę, cierpliwość i pomoc na każdym etapie pracy badawczej.

Pragnę także serdecznie podziękować mojemu promotorowi pomocniczemu,
Panu dr inż. Krzysztofowi Kanciakowi za cenne wskazówki techniczne i merytoryczne, wyrozumiałość, zaangażowanie i pomoc.

Spis treści

Lista skrótów

AES — Advanced Encryption Standard

AIG — And Inverter Graph

ANF — Algebraic Normal Form

ARX — Add-Rotate-Xor

ASIC — Application Specific Integrated Circuit

ASP — Answer Set Programming

CDCL — Conflict Driven Clause Learning

CNF — Conjunctive Normal Form

CP — Constraint Programming

CSP — Constraint Satisfaction Problem

$\mathbb{D}_{bn}$ — Differantial Branch Number

DES — Data Encryption Standard

DPLL — Davis-Putnam-Logemann-Loveland

FPGA — Field Programmable Gate Array

IoT — Internet of Things

IP — Integer Programming

IT — Information Technology

LFSR — Linear Feedback Shift Register

LP — Linear Programming

MDS — Maximum Distance Separable

MILP — Mixed Integer Linear Programming

MIP — Mixed Integer Programming

NIST — National Institute of Standards and Technology

NLFSR — Nonlinear Feedback Shift Register

NSA — National Security Agency

RAM — Random Access Memory

SAT — Satisfability

SAW — Software Analysis Workbench

SBOX — Substitution Box

SMT — Satisfiability Modulo Theories

SPN — Substitution Permutation Network

STP — Simple Theorem Prover

UNSAT — Unsatisfiability

XOR — Exclusive OR

Streszczenie

W rozprawie doktorskiej skupiono się przede wszystkim na kryptoanalizie różnicowej i automatyzacji procesu mającego na celu określenie podatności algorytmu na ataki z tej grupy. W pracy zaproponowano zautomatyzowaną metodę poszukiwania optymalnych charakterystyk różnicowych (pod kątem minimalnej liczby aktywnych skrzynek podstawieniowych lub maksymalnej wartości prawdopodobieństwa) oraz zautomatyzowaną metodę poszukiwania charakterystyk różnicowych obciętych przydatnych z punktu widzenia adwersarza. Zaproponowane metody do reprezentacji modelu opisującego propagację odpowiednio charakterystyki różnicowej lub charakterystyki różnicowej obciętej wykorzystują grafy `AIG`. Sam model generowany jest na podstawie implementacji funkcji wyznaczającej prawdopodobieństwo zadanej charakterystyki różnicowej (na potrzeby badań wykonywanej w języku funkcyjnym `Cryptol`). Graf `AIG` reprezentujący to przekształcenie poddawany jest redukcji funkcjonalnej, a następnie na potrzeby wyznaczenia konkretnej postaci charakterystyki, konwertowany do problemu `SAT`, w którym zmienne reprezentujące charakterystykę różnicową traktowane są jako zbiór niewiadomych.

Podczas przeprowadzonych badań wykazano, że dzięki zaproponowanemu podejściu można efektywnie poszukiwać charakterystyk różnicowych. Wykazano, że wydajność wykonanego w ramach rozprawy oprogramowania jest zdecydowanie wyższa niż w przypadku ogólnodostępnego narzędzia `CryptoSMT`. Testy wykonywano na szyfrach blokowych: `AES`, `KLEIN`, `MIDORI`, `PRESENT`, `SPECK`, `SIMON`, `PYJAMASK` oraz `SATURNIN`. Ponadto scharakteryzowano własności modeli `SAT` (reprezentowanych jako zbiór klauzul w postaci `CNF`), które według oceny autora mają wpływ na czas rozwiązania problemu. W pracy wskazano również szereg nieznanych dotąd w literaturze charakterystyk różnicowych, wyznaczonych za pomocą zaproponowanej metody (m.in. optymalną pod względem maksymalnej wartości prawdopodobieństwa charakterystykę różnicową na 5 rund szyfru blokowego `AES` oraz na 10 rund szyfru blokowego `SPECK` ze 128 bitowym blokiem danych, optymalną pod względem minimalnej liczby aktywnych skrzynek podstawieniowych charakterystykę różnicową na 14 rund szyfru blokowego `AES`). W podsumowaniu przeprowadzonych badań wskazano także wybrane przez autora perspektywiczne kierunki dalszego rozwoju opracowanych metod.

Słowa kluczowe: kryptoanaliza różnicowa, charakterystyka różnicowa, charakterystyka różnicowa obcięta, szyfry blokowe, grafy `AIG`, `SAT` solvers.

Strona celowo pozostawiona pusta.

Abstract

This dissertation focuses mainly on differential cryptanalysis and automation of the process of determining the vulnerability of the algorithm to attacks based on differantial techniques. The work proposes an automated search method for optimal differential characteristics (in terms of the minimum number of active substitution boxes or the cumulative maximum probability value) and automated search method for truncated differential characteristics useful for the adversary. The proposed methods use `AIG` graphs to represent the model describing propagation of the differential characteristic or the truncated differentials. The model is generated from an implementation of the function determining the probability of a given differential characteristic, performed in the functional language `Cryptol`. The `AIG` graph representing this transformation is subjected to a functional reduction and then for the purpose of determining a specific form of the characteristic, it is converted to the `SAT` problem, in which the variables representing the differential characteristic are treated as a set of unknowns.

During the conducted research, it was shown that thanks to the proposed approach it is possible to effectively search for differential characteristics. It has been shown that the efficiency of the implemented software based on the proposed method is much higher than in the case of the `CryptoSMT` open source tool. Tests were performed on block ciphers: `AES`, `KLEIN`, `MIDORI`, `PRESENT`, `SPECK`, `SIMON`, `PYJAMASK` and `SATURNIN`. In addition, the properties of `SAT` models (represented as a set of clauses in the `CNF` form) which according to the author assessment affect the time of solving the problem were characterized. The work indicates also a few of previously unknown in the literature differential characteristics determined by the proposed method (e.g the optimal in terms of the maximum probability value differential characteristic for 5 rounds of the block cipher `AES` and 10 rounds of the block cipher `SPECK` with 128 bit block size and optimal in terms of the minimum number of active substitution boxes differential characteristic for 14 rounds of `AES`). The summary of the research indicates also selected by the author the perspective directions of development of the proposed methods.

Keywords: differential cryptanalysis, differential characteristic, truncated differential characteristic, block ciphers, `AIG` graphs, `SAT` solvers.

Strona celowo pozostawiona pusta.

Rozdział 1

Wprowadzenie

1.1 Wstęp

Szyfry blokowe są podstawowymi środkami odpowiadającymi za bezpieczeństwo komunikacji, zapewniając m.in. poufność wymienianych informacji. Obecnie trwają również intensywne prace nad konstrukcjami opartymi na szyfrach blokowych, które będą zapewniać tzw. szyfrowanie uwierzytelnione, w którym istnieje możliwość rozpoznania nieprawidłowych szyfrogramów i odmowy ich deszyfrowania. W tym miejscu warto wspomnieć także o dziedzinie nazywanej potocznie kryptografią lekką (ang. lightweight cryptography). Jej celem jest dostarczenie bezpiecznych szyfrów odpowiednich do implementacji i wykorzystania w małych urządzeniach o ograniczonych zasobach. Duży obszar wymagający takich właśnie rozwiązań to rynek urządzeń internetu rzeczy (ang. internet of things). Ze względu na ograniczenia sprzętowe lekki szyfr zwykle wykorzystuje podstawowe operacje logiczne takie jak rotacje, dodawanie i odejmowanie modularne czy też różnicę symetryczną zwaną popularnie operacją `XOR`.

Szyfry blokowe wykorzystywane są m.in.:

- do konstrukcji szyfratorów sprzętowych;
- w protokołach kryptograficznych;
- w podpisach cyfrowych;
- w bankowość elektronicznej;
- przy określaniu narodowych i przemysłowych standardów szyfrowania symetrycznego np.:
  - `AES` (standard przemysłowy FIPS 197),
  - `GOST R 34.12-2015` (Rosja),
  - `GB/T 32907-2016` (Chiny),
  - `STB 34.101.31-2011` (Białoruś);

- w kryptografii lekkiej, np.:
  - PRESENT,
  - SPECK,
  - SIMON.

Szczegółowa analiza bezpieczeństwa szyfru blokowego powinna być przeprowadzona przez projektantów przed jego akredytacją. Bezwzględnie powinna być również częścią dokumentacji udostępnionej podmiotowi akredytującemu.

Wiele firm rezygnuje z zatrudniania ekspertów z dziedziny kryptografii oraz powolnego procesu opracowywania i analizowania bezpieczeństwa mechanizmów kryptograficznych przed wdrożeniem. Zamiast tego korzysta z gotowych już rozwiązań. Standardy szyfrowania czy też funkcji skrótu wyłaniane są w różnego rodzaju międzynarodowych konkursach standaryzacyjnych (np. *Advanced Encryption Standard Competition*, *eSTREAM: the ECRYPT Stream Cipher Project*, *CAESAR: Competition for Authenticated Encryption: Security Applicability and Robustness*). Podejście to pozwala zaoszczędzić czas i ograniczyć koszty. Daje również argumenty za tym, że wybrany algorytm został rzetelnie i dokładnie oceniony przez grono ekspertów w dziedzinie kryptografii. Należy jednak mieć świadomość, że nawet algorytm, który uznany został za najlepszy w długim procesie certyfikacyjnym, może nie być bezwarunkowe bezpieczny. Swoje przemyślenia w tej kwestii przedstawił ceniony w środowisku kryptograficznym Daniel J. Bernstein [7].

Wykonanie rzetelnej oceny bezpieczeństwa algorytmu kryptograficznego jest bez wątpienia procesem złożonym i czasochłonnym. W niniejszej rozprawie doktorskiej zaproponowałem metody, które pozwalają zautomatyzować część tego procesu. Główny nacisk położono na analizę bezpieczeństwa skoncentrowaną wokół kryptoanalizy różnicowej.

1.2 Cel i zakres pracy

Celem przedłożonej rozprawy doktorskiej jest opracowanie wydajnej, automatycznej i generycznej (mającej zastosowanie przy każdej konstrukcji analizowanego algorytmu) metody poszukiwania optymalnych charakterystyk różnicowych (pod kątem maksymalnej wartości prawdopodobieństwa lub minimalnej liczby aktywnych skrzynek podstawieniowych) i charakterystyk różnicowych obciętych (o prawdopodobieństwie większym niż szum) na zadaną liczbę rund szyfru blokowego.

W ramach rozprawy przybliżony został aktualny stan badań w zakresie automatycznych metod poszukiwania charakterystyk różnicowych. Pierwsze rozdziały tj. nr 2 oraz 3 zawierają również podstawowe informacje związane z tematyką konstrukcji współczesnych szyfrów blokowych, ich bezpieczeństwem i kryptoanalizą. Rozdział nr 4 przybliża charakterystykę problemów CSP wybieranych do modelowania procesów związanych z automatyzacją metod poszukiwania charakterystyk różnicowych. W rozdziałach nr 5 oraz 6 oprócz przybliżenia już znanych metod do określenia minimalnej liczby skrzynek podstawieniowych w sieci SPN zaproponowano metodę poszukiwania optymalnych charakterystyk

różnicowych (pod kątem maksymalnej wartości prawdopodobieństwa lub minimalnej liczby aktywnych skrzynek podstawieniowych) oraz metodę poszukiwania charakterystyk różnicowych obciętych (o prawdopodobieństwie większym niż szum) na zadaną liczbę rund szyfru blokowego. W celu automatyzacji procesów związanych z opisem propagacji charakterystyki różnicowej i charakterystyki różnicowej obciętej zaproponowano wykorzystanie grafów `AIG`. Na bazie przedstawionych w rozprawie metod, zaimplementowano narzędzie pozwalające na automatyczną analizę szyfru blokowego pod kątem odporności na kryptoanalizę różnicową oraz kryptoanalizę różnicową z wykorzystaniem charakterystyk różnicowych obciętych. Dodatkowo w rozdziale nr 6 opisano własności modeli `SAT` opisujących propagację charakterystyki różnicowej (dla wybranych szyfrów blokowych) i charakterystyki różnicowej obciętej generowanych poprzez grafy `AIG`. Porównano wydajność zaimplementowanego narzędzia względem ogólnodostępnego oprogramowania o nazwie *CryptoSMT* [61].

Rozdział nr 7 zawiera podsumowanie przeprowadzonych badań, spostrzeżenia i wnioski oraz propozycje dalszych kierunków, w ramach których można rozwijać zarówno opracowane metody, jak i zaimplementowane narzędzie.

1.3 Aktualny stan badań i literatura przedmiotu

Badania związane z opracowywaniem efektywnych metod poszukiwania charakterystyk różnicowych są prowadzone od stosunkowo niedawna. Dopiero w ostatnich dwudziestu latach, które upłynęły od początku XXI wieku wraz z dynamicznym rozwojem technologii oraz potrzebą utrzymania poufności i integralności ogromnych ilości danych, ich intensywność przybiera na sile. Widać wyraźnie trend, w którym środowisko kryptograficzne jest coraz bardziej sceptyczne do nowo powstałych algorytmów (szczególnie przedstawianych przez potentatów związanych z branżą `IT`). Istnieje bowiem obawa, że autorzy algorytmów mogli pozostawić subtelne zmiany, które pozwalają na skompromitowanie szyfru. Jeżeli algorytm byłby wykorzystywany komercyjnie i na masową skalę mogliby wtedy bezkarnie naruszać poufność danych bez wiedzy użytkownika. Podatności te mogą również nie być celowe, a wynikać ze względów uproszczenia konstrukcji w celu optymalizacji czasu szyfrowania lub minimalizacji zasobów wymaganych do wdrożenia algorytmu w sprzęcie. Ważne jest więc, aby dysponować odpowiednią metodyką, która w sposób jak najprostszy i bezsporny dostarczy matematycznych dowodów odporności algorytmu na znane ataki kryptograficzne.

Jedną najbardziej efektywnych metod kryptoanalizy szyfrów blokowych jest kryptoanaliza różnicowa oraz jej warianty takie jak kryptoanaliza różniczek obciętych oraz niemożliwych. Niniejsza rozprawa doktorska skupia się na badaniach związanych z opracowaniem zautomatyzowanych metod poszukiwania optymalnych charakterystyk różnicowych (pod kątem maksymalnej wartości prawdopodobieństwa lub minimalnej liczby aktywnych skrzynek podstawieniowych). Pozwolą one na sprawne i efektywne sprawdzenie odporności szyfru blokowego na wspomniane wcześniej ataki różnicowe. Wydaje się, że problem poszukiwania

optymalnych charakterystyk różnicowych jest w ogólności problemem klasy NP[1]. Obecnie nie są znane metody, które rozwiązywałyby ten problem w czasie wielomianowym. Każda zaproponowana do tej metoda pory działa w czasie wykładniczym.

Aktualnie metody poszukiwania optymalnych charakterystyk różnicowych dzieli się na trzy główne grupy:

- metody bazujące na modyfikacjach metody podziału i ograniczeń (ang. *branch and bound*) [51] [15];
- metody polegające na sprowadzeniu problemu poszukiwania charakterystyk do jednego z problemów `CP` [55] [66] [65] [64] [33], najczęściej problemu `MIP` lub `MILP`;
- metody polegające na sprowadzeniu problemu poszukiwania charakterystyk do problemu `SMT\SAT` [54] [4] [44].

Niestety proces poszukiwania charakterystyk różnicowych jest ściśle powiązany ze szczegółową analizą algorytmu i trudno go zautomatyzować. Konstrukcja modelu matematycznego, który w jak najdokładniejszy sposób opisuje propagację charakterystyki różnicowej wraz ze wzrostem liczby rund szyfru blokowego, jest więc zadaniem żmudnym i czasochłonnym. Optymalizacja tego procesu jest przedmiotem wielu badań i analiz [62].

Jednym z uznanych narzędzi do analizy szyfrów blokowych pod kątem ich odporności na kryptoanalizę różnicową i liniową jest oprogramowanie *CryptoSMT* [61]. Metody poszukiwania charakterystyk różnicowych i liniowych implementowane we wskazanym oprogramowaniu bazują na sprowadzeniu problemu poszukiwania charakterystyki do modelu `SMT\SAT`, a następnie wyznaczeniu rozwiązania modelu za pomocą takich narzędzi jak *STP*, *Boolector*, *CryptoMiniSat*. Wydajność wspomnianego oprogramowania nie jest jednak zadowalająca, a co więcej nie ma w nim możliwości poszukiwania charakterystyk różnicowych do bardziej zaawansowanych technik kryptoanalizy różnicowej takich jak kryptoanaliza z wykorzystaniem różniczek obciętych (ang. *truncuted differential cryptanalysis*) oraz niemożliwych (ang. *impossible differential cryptanalysis*).

Głównym elementem, który odróżnia zaproponowane rozwiązanie od prac uznawanych za wiodące w tej dziedzinie jest wykorzystanie do opisu propagacji charakterystyki różnicowej (lub charakterystyki różnicowej obciętej) bezpośrednio grafów `AIG`. Graf `AIG` generowany jest na bazie implementacji funkcji opisującej sposób propagacji charakterystyki różnicowej lub charakterystyki różnicowej obciętej. Obecnie wszystkie ogólnodostępne narzędzia oparte są na metodach, w których równania opisujące propagację charakterystyki różnicowej są predefiniowane dla każdego algorytmu przy wykorzystaniu trików zaproponowanych przez ich autorów. W tabeli nr 1.1 przedstawiono zestawienie głównych prac wiodących w omawianej tematyce.

[1]Problem decyzyjny, dla którego weryfikacja rozwiązania można zostać wykonana w czasie wielomianowym. Alternatywna definicja mówi, że problem klasy NP, może być rozwiązany w wielomianowym czasie na niedeterministycznej maszynie Turinga

Tablica 1.1: Zestawienie głównych prac wiodących w tematyce poszukiwania charakterystyk różnicowych.

	Tool for cryptanalysis of symmetric primitives — `CryptoSMT` [61]	Accelerating the Search of Differential Characteristics with the SAT Method [62]	Rozwiązanie zaproponowane w niniejszej rozprawie doktorskiej
Rok publikacji	2016	2021	2021
Rodzaj modelu	`SAT`	`SAT`	`SAT`
Sposób generacji równań	Predefiniowane w opisie szyfru	Predefiniowane w opisie szyfru	Bezpośrednio na bazie grafu `AIG`
Charakterystyki różnicowe obcięte	✗	✗	✓
Testy szyfrów z 8-bitowym SBOX-em	✗	✗	✓
Bazowe narzędzia do rozwiązania modelu	STP, Boolector, CryptoMiniSat	CaDiCaL	CaDiCaL

Strona celowo pozostawiona pusta.

Rozdział 2

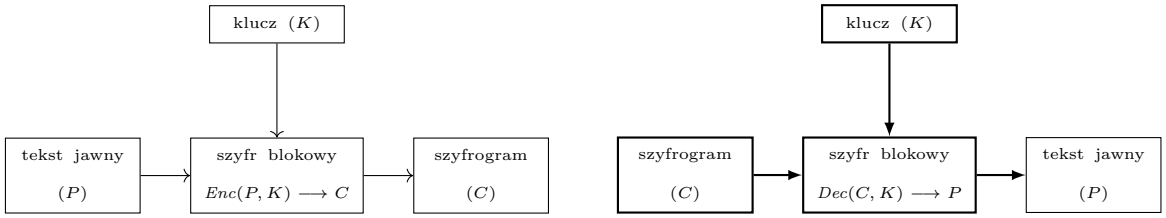
Szyfry blokowe

2.1 Definicja

Szyfrowanie blokowe to jeden z rodzajów szyfrowania symetrycznego, które dokonuje przekształcenia bloku danych o stałej długości. Rozmiar bloku wynosi zwykle 64, 128 lub 256 bitów i jest przekształcany iteracyjnie z wykorzystaniem tajnego klucza. Rozmiar klucza wyrażany jest w liczbie bitów i jest zwykle wielokrotnością liczby 64, czyli kolejno: 64, 128, 192, 256, 512 lub niekiedy nawet 1024 czy też 2048 bitów. Algorytm szyfrowania i deszyfrowania opisany jest odpowiednio przez funkcję szyfrowania *(Enc)* oraz deszyfrowania *(Dec)*:

$$Enc\ (P \in \mathbb{F}_2^n, K \in \mathbb{F}_2^m) \longrightarrow C \in \mathbb{F}_2^n$$
$$Dec\ (C \in \mathbb{F}_2^n, K \in \mathbb{F}_2^m) \longrightarrow P \in \mathbb{F}_2^n$$

gdzie wiadomość P o długości n przekształcana jest w szyfrogram C o długości n przy wykorzystaniu klucza K o długości m (rys. 2.1). Funkcja opisująca przekształcenie realizowane przez szyfr blokowy jest bijekcją, to znaczy, że jest odwracalna w obie strony i przekształca każdy możliwy wektor wejściowy w dokładnie jeden wektor wyjściowy zależny od klucza. Uściślając szyfr blokowy to sposób generowania rodziny permutacji, która indeksowana jest tajnym kluczem K [41].



(a) Szyfrowanie. (b) Deszyfrowanie.

Rysunek 2.1: Szyfr blokowy – schemat koncepcyjny.

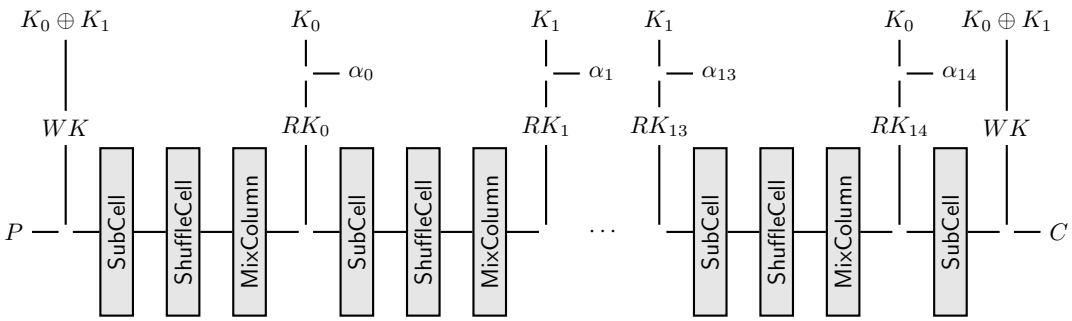
2.2 Metody konstrukcji

2.2.1 Sieć SPN

Sieć `SPN` (ang. Substitution Permutation Network), której ideę przedstawiono na rysunku nr 2.3 jest jedną z podstawowych konstrukcji wykorzystywanych przy projektowaniu szyfrów blokowych. Składa się z trzech tzw. warstw realizujących następujące przekształcenia:

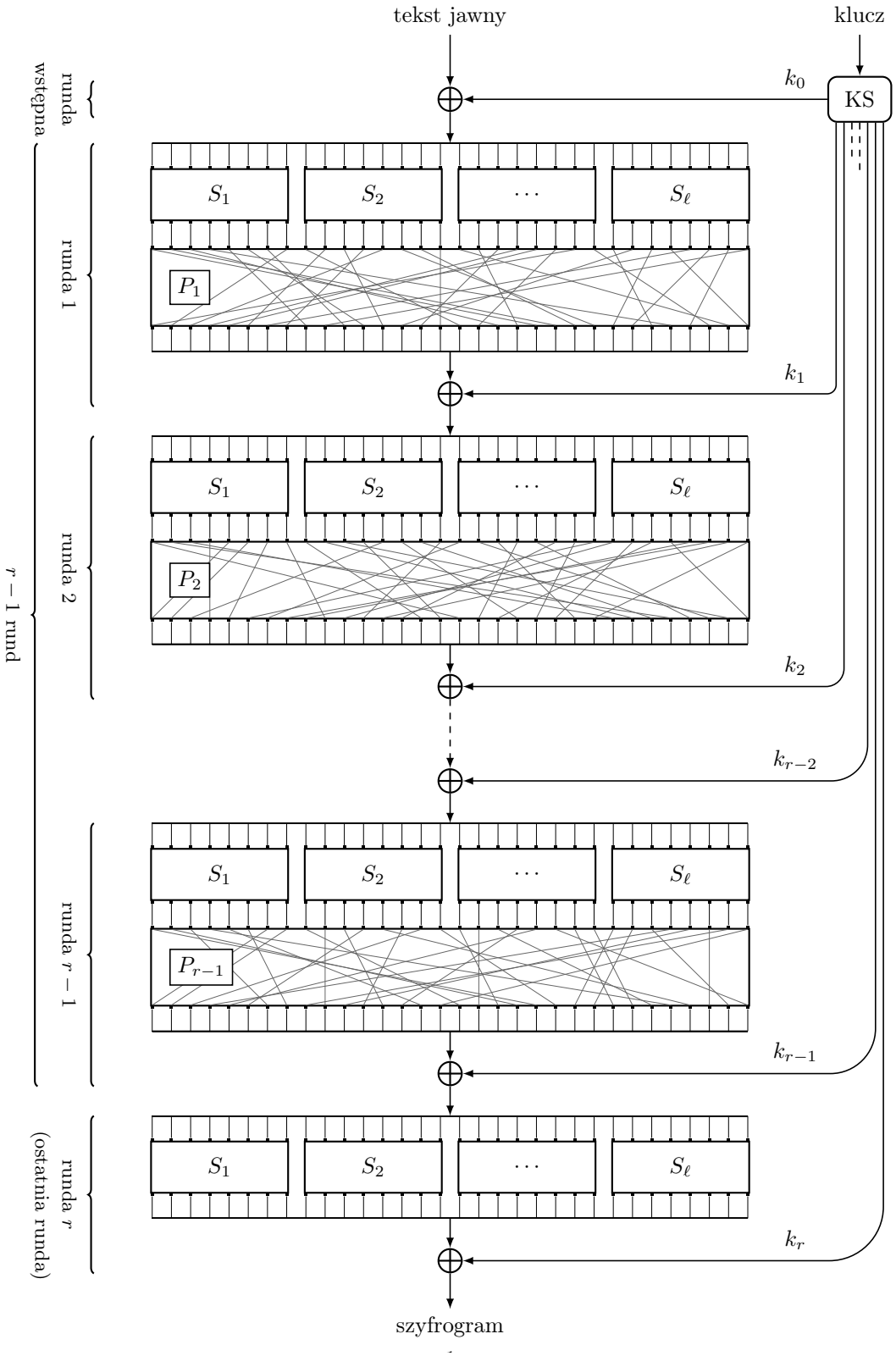
- podstawienie;
- permutacje;
- dodanie klucza rundowego.

Wszystkie operacje składają się razem na funkcję rundy. Algorytmy, których konstrukcja oparta jest na sieci `SPN` to m.in. `AES` (Rijndael, rys. 2.8), `Present`, `Kalyna`, `Midori` (rys. 2.2), `Kuznyechik`.



Rysunek 2.2: Schemat koncepcyjny szyfru blokowego `MIDORI` [5] opartego na sieci `SPN`.

W przeciwieństwie do metod konstrukcji opartych o sieć `Feistela` [rozdz. 2.2.2] czy też strukturę `Lai-Massey'a` [rozdz. 2.2.3] wszystkie operacje w sieci `SPN` muszą być odwracalne. W innym przypadku operacja deszyfrowania byłaby niemożliwa.



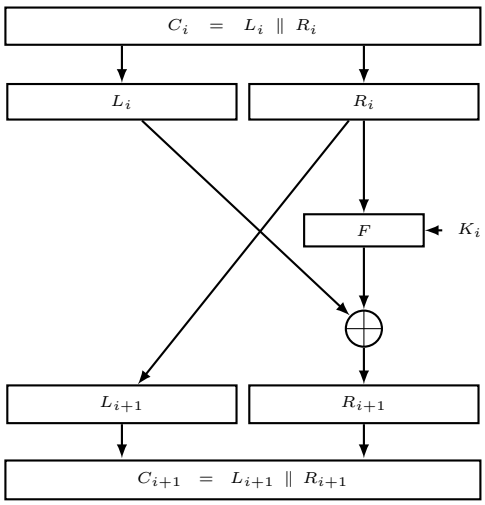
Rysunek 2.3: Schemat koncepcyjny szyfru blokowego opartego na sieci `SPN`.

2.2.2 Sieć Feistela

Główna idea algorytmu opartego na sieci `Feistela` polega na podzieleniu tekstu jawnego na dwie części i wykonaniu funkcji rundy na jednej z nich. W kolejnym kroku części szyfrogramu są zamieniane miejscami. Dodatkowo obliczana jest różnica symetryczna (`XOR`) wyniku funkcji rundy z drugą częścią tekstu jawnego. Model matematyczny sieci `Feistela` prezentuje się następująco:

$$L_{i+1} = R_i$$
$$R_{i+1} = L_i \oplus F(L_{i+1}, K_i)$$

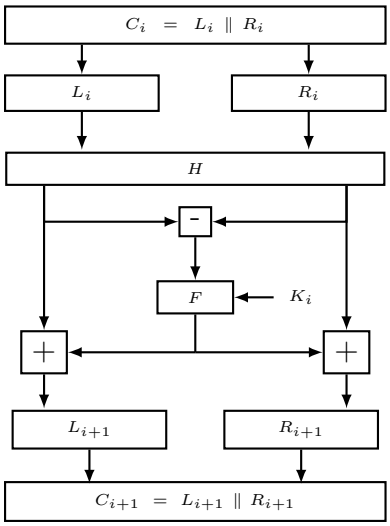
Funkcja rundy wykorzystana w sieci `Feistela` w przeciwieństwie do funkcji rundy w sieci `SPN` nie musi być odwracalna. Jedna runda wykonywana jest zwykle kilkanaście razy. Algorytmy, których konstrukcja oparta jest na sieci `Feistela` to m.in. `DES`, `Threefish`, `Skipjack`, `CAST`, `MISTY1`.



Rysunek 2.4: Schemat koncepcyjny rundy szyfru blokowego opartego na sieci `Feistela`.

2.2.3 Struktura Lai-Massey'a

Struktura `Lai-Massey'a` została przedstawiona w algorytmie `IDEA` oraz `IDEA NXT`. Jest ona pewnego rodzaju modyfikacją sieci `Feistela`, polegającą na dodaniu dodatkowej warstwy pomiędzy rundami. Ponadto operacje `XOR` na częściach szyfrogramu zastąpiono operacją dodawania lub odejmowania modularnego.

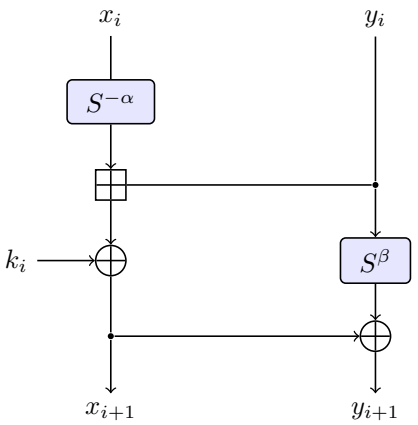


Rysunek 2.5: Schemat koncepcyjny rundy szyfru blokowego opartego na strukturze `Lai-Massey'a`.

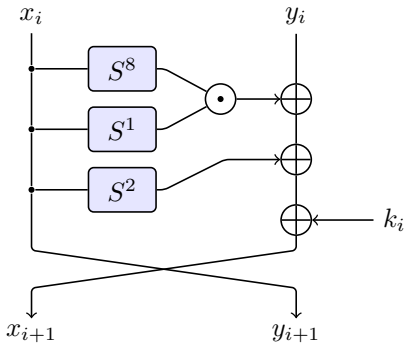
2.3 ARX

Konstrukcje wielu tzw. lekkich szyfrów blokowych[1] opierają się na sprawdzonych strukturach (sieć `Feistela`, struktura `Lai-Massey'a`) wykorzystując przy tym zbiór operacji nazywanych potocznie `ARX`. Skrót ten wywodzi się od nazwy trzech operacji matematycznych: dodawanie (ang. addition), rotacja (ang. rotate), różnica symetryczna (`XOR`). Wszystkie te operacje są wydajne sprzętowo, przez co implementacje szyfrów z rodziny `ARX` uzyskują dużą przepustowość. Szyfry `ARX` są tym samym efektywne pod względem implementacji sprzętowych (`FPGA`, `ASIC`), gdyż ich realizacja w układzie wymaga małej liczby podstawowych bramek logicznych. Najbardziej znanym reprezentantem rodziny szyfrów `ARX` są algorytmy takie jak `FEAL`, `BLAKE`, `SPECK` (rys. 2.6) czy też `SIMON` (rys. 2.7).

[1]Szyfry blokowe, których konstrukcja zoptymalizowana jest pod kątem zajętości układu cyfrowego, na którym może zostać zrealizowana sprzętowa implementacja oraz szybkości, z jaką może zostać zrealizowane szyfrowanie/deszyfrowanie.



Rysunek 2.6: Runda szyfru blokowego SPECK [38].



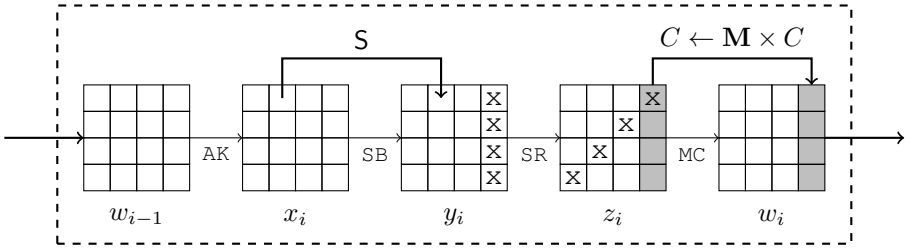
Rysunek 2.7: Runda szyfru blokowego SIMON [38].

2.4 Strategia szerokiej ścieżki

Rozwój kryptoanalizy i odkrycie takich technik jak kryptoanaliza różnicowa [14] czy też liniowa [50] doprowadził do powstania koncepcji projektowania szyfrów blokowych nazywanej *strategią szerokiej ścieżki* (ang. *wide trial strategy*) [26]. Na bazie tej strategii powstał m.in. szyfr blokowy *Rijndael*, który został wybrany standardem szyfrowania w konkursie standaryzacyjnym *Advanced Encryption Standard Competition* organizowanym przez NIST w latach 1997 - 2000.

W uproszczeniu strategia szerokiej ścieżki polega na zaprojektowaniu rundy szyfru blokowego w oparciu o starannie dobrane przekształcenie liniowe oraz uzupełniające je nieliniowe przekształcenie o stosunkowo małym rozmiarze, posiadającym jednak dobre własności uniemożliwiające aproksymację liniową i różnicową. Definicja warstwy nieliniowej szyfru najczęściej opiera się na 4 lub 8-bitowych skrzynkach podstawieniowych (ang *substitution boxes*). Strategia szerokiej ścieżki kładzie nacisk na warstwę liniową szyfru. Musi być ona zaprojektowana tak, aby zapewnić wysoki poziom dyfuzji już po kilku rundach

algorytmu. Korzystając z takich własności jak *differential branch number* warstwy liniowej [rozdz. 2.6.3], minimalnej liczby aktywnych skrzynek [rozdz. 2.6.5] oraz profilu różnicowego wykorzystanej skrzynki [rozdz. 2.6.4] projektant może w łatwy sposób oszacować maksymalną wartość prawdopodobieństwa charakterystyki różnicowej oraz liniowej. Nie daje mu to jednak możliwości oszacowania wartości prawdopodobieństwa optymalnej różniczki[2] czy też charakterystyki różnicowej obciętej.



Rysunek 2.8: Schemat koncepcyjny rundy szyfru blokowego `AES` [38].

Ostatnie badania pokazują jednak, że oparcie konstrukcji szyfru na strategii szerokiej ścieżki nie musi być gwarantem sukcesu, a złośliwy projektant może zaszyć w takiej konstrukcji podatności, które umożliwią późniejszą kompromitację szyfru [57]. Jedną z głównych wad strategii szerokiej ścieżki jest fakt, że nie rozważa ona charakterystyk różnicowych wyższego rzędu. Tym samym wykorzystując jej założenia, nie można oszacować odporności na takie techniki kryptoanalizy różnicowej jak kryptoanaliza z wykorzystaniem charakterystyk obciętych (ang. *truncuted differential cryptanalysis*) oraz niemożliwych (ang. *impossible differential cryptanalysis*).

2.5 Bezpieczeństwo

Szyfr blokowy uznaje się za bezpieczny, jeżeli nie da się skonstruować ataku o złożoności mniejszej niż $\mathcal{O}(2^m)$, gdzie $\mathcal{O}$ jest symbolem Landaua (złożoność ataku siłowego) pozwalającego odzyskać tajny klucz K o długości m bitów. W trakcie szyfrowania i deszyfrowania nie może zostać ujawniona żadna informacja o kluczu czy też przetwarzanym tekście jawnym.

2.6 Wybrane własności szyfrów blokowych

2.6.1 Nieodróżnialność ciągu szyfrogramów od ciągu prawdziwie losowego

Idea wielu technik kryptoanalizy opiera się przede wszystkim na wychwytywaniu wszelkich anomalii statystycznych w szyfrogramie, a następnie wykorzystaniu ich w celu odzy-

[2]Różnice pomiędzy różniczką a charakterystyką różnicową opisano w rozdz. 3.2.

Tablica 2.1: Szacowany poziom bezpieczeństwa algorytmu symetrycznego bazujący na długości wykorzystanego klucza tajnego K według [41].

Długość klucza K [w bitach]	**Liczba operacji w przypadku ataku siłowego**	**Status bezpieczeństwa**
40	2^{40}	brak bezpieczeństwa
64	2^{64}	słaby poziom bezpieczeństwa
80	2^{80}	względne bezpieczeństwo
128	2^{128}	dobry poziom bezpieczeństwa
256	2^{256}	bardzo wysoki poziom bezpieczeństwa

skania tajnego klucza. Wadą kompromitującą m.in. szyfry monoalfabetyczne czy też polialfabetyczne jest bezspornie fakt, że uzyskany za ich pomocą szyfrogram z łatwością można odróżnić od danych losowych. Wyszukując anomalie wynikające z własności statystycznych języka można w prosty sposób złamać algorytmy tego typu [37].

W przypadku współczesnych szyfrów jedną z pierwszych własności, jaką sprawdza się podczas analizy bezpieczeństwa, jest losowość szyfrogramów uzyskiwanych na różne sposoby m.in. poprzez dobór specyficznych tekstów jawnych, wykorzystanie różnych trybów szyfrowania czy też specyficznych postaci klucza. Losowość uzyskanych w ten sposób danych powinna wynikać wprost z własności konstrukcyjnych szyfru i wybranych prymitywów krypotograficznych (np. macierzy `MDS`, skrzynek podstawieniowych, permutacji). Uzyskany strumień danych nie powinien być odróżnialny od ciągu prawdziwie losowego. Techniki kryptoanalizy takie jak kryptoanaliza różnicowa i liniowa starają się wykorzystać nielosowość szyfrogramów uzyskiwanych przy pomocy specyficznie dobranych par tekstów jawnych. Wspomniane wcześniej techniki kryptoanalizy są metodami ataku z wybranym tekstem jawnym a adwersarz stara się przewidzieć postać odpowiednio charakterystyki różnicowej lub liniowej na wyjściu algorytmu przy ustalonej charakterystyce wejściowej. W przypadku szyfrów blokowych i klasycznych technik kryptoanalizy różnicowej i liniowej, jeżeli prawdopodobieństwo wskazanej przez atakującego wyjściowej charakterystyki jest większe niż $\frac{1}{2^n}$ (gdzie n to długość przetwarzanego bloku wyrażona w liczbie bitów). Teoretycznie możliwa jest konstrukcja ataku o złożoności obliczeniowej mniejszej niż atak siłowy.

2.6.2 Konfuzja i dyfuzja

W kryptologii konfuzja (ang. confusion) i dyfuzja (ang. diffusion) to dwie własności, które muszą zostać zapewnione przez bezpieczny szyfr na odpowiednio wysokim poziomie. Zdefiniował je Claude Shannone w 1949 roku [60] i do dziś stanowią one podstawę konstrukcji szyfrów blokowych.

Mianem dyfuzji (potocznie nazywanej *rozpraszaniem*) określa się proces wymieszania bitów przetwarzanych danych, który ma na celu rozmycie wszelkich związków pomiędzy bitami tekstu jawnego równomiernie na całym przetwarzanym bloku. W teorii ma to unie-

możliwić odkrycie związków statystycznych pomiędzy bitami na podstawie obserwacji samego szyfrogramu [45]. We współczesnych szyfrach blokowych własność dyfuzji zapewniana jest przez przekształcenia liniowe takie jak permutacje bitowe, macierze `MDS`, rotacje i przesunięcia. W blokowym algorytmie szyfrującym z odpowiednią dyfuzją zmiana dowolnego bitu w bloku wejściowym spowoduje zmianę każdego z bitów wyjściowych z prawdopodobieństwem $\frac{1}{2}$ (czasami własność ta nazywana jest ścisłym kryterium lawinowym). Własności dyfuzyjne przekształcenia można zmierzyć również za pomocą parametru nazywanego *differential branch number* [rozdz. 2.6.3].

Własność konfuzji (potocznie nazywanej *mieszaniem*) polega na związaniu szyfrowanych danych z kluczem tak, aby uczynić ten związek bardzo skomplikowanym. Konfuzja zazwyczaj wprowadzana jest za pomocą operacji nieliniowych np. skrzynek podstawieniowych czy też dodawania arytmetycznego wektorów o współrzędnych binarnych, traktowanych jako reprezentacje liczb naturalnych w układzie pozycyjnym dwójkowym. W prostym szyfrze podstawieniowym np. szyfrze Cezara konfuzja wprowadzana jest poprzez zastąpienie każdej litery tekstu jawnego inną literą, określoną przez wykorzystany klucz. We współczesnych szyfrach mechanizm ten jest o wiele bardziej skomplikowany, a w praktyce zależy on zarówno od bitów klucza, jak i od innych bitów tekstu jawnego.

Funkcja rundy we współczesnych szyfrach blokowych wprowadza zarówno konfuzję, jak i dyfuzję. Jak już wcześniej wspomniano poziom dyfuzji, można zmierzyć np. za pomocą ścisłego kryterium lawinowego, a poziom konfuzji np. za pomocą aproksymacji różnicowej (ang. differential approximations) lub aproksymacji liniowej (ang. linear approximations).

2.6.3 Differential Branch Number

Jednym z kluczowych parametrów wykorzystywanym w kryptoanalizie różnicowej szyfrów blokowych jest parametr *Differential Branch Number*. Mierzy on siłę (moc) warstwy dyfuzyjnej (realizowanej np. za pomocą mnożenia wektora przez macierz `MDS`) i odnosi się do liczby aktywnych elementów (jednostek) na wejściu i wyjściu przekształcenia.

Najprostsza i intuicyjna nieformalna definicja parametru *Differential Branch Number* może być kojarzona z minimalną liczbą aktywnych wartości tj. różnych od zera (np. bitów, bajtów itp.) na wyjściu i wejściu badanej operacji [58]. Oczywiście, aby przedstawiona definicja miała zastosowanie praktyczne, wykluczamy przypadek trywialny tzn. taki, w którym różnice te nie występują. Uściślając zatem niech $\mathbb{D}_{bn}$ określa wartość parametru *Differential Branch Number* dla funkcji F, wtedy:

$$\mathbb{D}_{bn} = min(hw(x \oplus x \oplus \Delta_x) + hw(F(x) \oplus F(x \oplus \Delta_x)),$$

$$\mathbb{D}_{bn} = min(hw(\Delta_x) + hw(\Delta_y))$$

gdzie:

- hw — funkcja określająca wagę (w przypadku gdy parametr *Differential Branch Number* określany jest z dokładnością co do każdego bitu różnicy będzie to funkcja wyznaczająca wagę Hamminga),

Δ_x — różnica wejściowa,

Δ_y — różnica wyjściowa.

Parametr ten wyznacza się zazwyczaj dla liniowej części algorytmu odpowiedzialnej za dyfuzję (np. operacja *MixColumns* w szyfrze blokowym `AES`). Można go wykorzystać np. do konstrukcji modelu, który opisuje propagację aktywnych skrzynek podstawieniowych w sieci `SPN` (rozdz. 5.2.1).

2.6.4 Profil różnicowy

Profil różnicowy opisuje własności różnicowe zadanego przekształcenia $\mathbb{P} : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2^n$, poprzez jednoznaczne określenie prawdopodobieństwa wystąpienia zadanej charakterystyki różnicowej wyjściowej przy zadanej charakterystyce różnicowej wejściowej. Może być określony za pomocą tablicy dwuwymiarowej o rozmiarze 2^m na 2^n lub za pomocą wzoru.

Przedstawienie profilu różnicowego za pomocą dwuwymiarowej tablicy teoretycznie możliwe jest dla każdego przekształcenia, jednak w praktyce nie jest to wykonalne ze względu ograniczonych zasobów pamięciowych i obliczeniowych. Profil różnicowy najczęściej tablicowany jest dla małych przekształceń takich jak skrzynka podstawieniowa (ang. *substitution box*), a dla takich przekształceń jak dodawanie arytmetyczne w $\mathbb{F}_2^n$ opisywany jest za pomocą wzorów [48]. W literaturze można też odnaleźć wzory określające własności różnicowe operacji arytmetycznych `AND` oraz `OR` [28] [16] wykorzystywanych w szyfrach opartych na konstrukcji `ARX`.

W tabeli nr 2.2 przedstawiono postać 4-bitowej skrzynki podstawieniowej wykorzystywanej w algorytmie `PRESENT`, a w tabeli nr 2.3 zaprezentowano profil różnicowy tej skrzynki. Można z niego jednoznacznie odczytać prawdopodobieństwo wystąpienia charakterystyki różnicowej Δ_{out} (etykiety kolumn) przy ustalonej charakterystyce różnicowej wejściowej Δ_{in} (etykiety wierszy). Prawdopodobieństwo wystąpienia charakterystyki różnicowej Δ_{out} przy charakterystyce wejściowej Δ_{in} równe jest liczbie zadanej w wierszu Δ_{in} i kolumnie Δ_{out} dzielonej przez liczbę wszystkich możliwych par dających określoną charakterystykę Δ_{in} (naturalną własnością profilu różnicowego jest fakt, że suma wartości w wierszu równa się liczbie wszystkich możliwych par określających zadaną charakterystykę wejściową, w prezentowanym przypadku będzie to liczba 16). Wartości w tak prezentowanym profilu różnicowym będą parzyste, ponieważ parze (x, y) określającej[3] przejście $\Delta_{in} \xrightarrow{p} \Delta_{out}$ (zdarzenie zachodzące z prawdopodobieństwem równym p) odpowiada para (y, x).

Tablica 2.2: Postać 4-bitowej skrzynki podstawieniowej wykorzystywanej w szyfrze blokowym `PRESENT`.

wejście	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
wyjście	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

[3] $\Delta_{in} = x \oplus y$

Tablica 2.3: Profil różnicowy skrzynki podstawieniowej wykorzystywanej w szyfrze blokowym PRESENT (postać skrzynki zaprezentowano w tabeli nr 2.2).

$\Delta_{in} \backslash \Delta_{out}$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	4	0	0	0	4	0	4	0	0	0	4	0	0
2	0	0	0	2	0	4	2	0	0	0	2	0	2	2	2	0
3	0	2	0	2	2	0	4	2	0	0	2	2	0	0	0	0
4	0	0	0	0	0	4	2	2	0	2	2	0	2	0	2	0
5	0	2	0	0	2	0	0	0	0	2	2	2	4	2	0	0
6	0	0	2	0	0	0	2	0	2	0	0	4	2	0	0	4
7	0	4	2	0	0	0	2	0	2	0	0	0	2	0	0	4
8	0	0	0	2	0	0	0	2	0	2	0	4	0	2	0	4
9	0	0	2	0	4	0	2	0	2	0	0	0	2	0	4	0
A	0	0	2	2	0	4	0	0	2	0	2	0	0	2	2	0
B	0	2	0	0	2	0	0	0	4	2	2	2	0	2	0	0
C	0	0	2	0	0	4	0	2	2	2	2	0	0	0	2	0
D	0	2	4	2	2	0	0	2	0	0	2	2	0	0	0	0
E	0	0	2	2	0	0	2	2	2	2	0	0	2	2	0	0
F	0	4	0	0	4	0	0	0	0	0	0	0	0	0	4	4

W implementowanych w ramach niniejszej rozprawy doktorskiej narzędziach do opisu propagacji charakterystyki różnicowej w algorytmach blokowych wykorzystywane są tzw. *zlogarytmowane profile różnicowe* zadawane przez macierz lub odpowiednio dostosowane wzory, w których zawarta jest pośrednia informacja o prawdopodobieństwie p, z jakim różnica wejściowa Δ_{in} przechodzi na różnicę wyjściową Δ_{out}. Wartość w tak przedstawionym profilu wyznaczana jest następująco:

$$\mathbb{DDT}_{log}(\Delta_i, \Delta_j) \begin{cases} \log_x \frac{1}{p} & \text{, jeżeli } p > 0 \\ -1 & \text{, jeżeli } p = 0. \end{cases} \tag{2.1}$$

W przypadku klasycznej kryptoanalizy różnicowej podstawa logarytmu x wynosi 2, ponieważ wartość prawdopodobieństwa zawsze będzie liczbą postaci 2^{-m}. Przyjęcie takiej podstawy logarytmu nie wpływa na dokładność prowadzonych obliczeń.

Wykorzystanie tablic (lub odpowiednio zmodyfikowanych wzorów) określających zlogarytmowaną wartość odwrotności prawdopodobieństwa znacznie przyspiesza czas rozwiązania problemu poszukiwania charakterystyki różnicowej. Zabieg ten pozwala uniknąć operacji mnożenia przy wyznaczaniu wartości prawdopodobieństwa wystąpienia charakterystyki różnicowej (sumowaniu prawdopodobieństw). Przy obliczeniach wykorzystana jest suma logarytmów. Przykładowy zlogarytmowany profil różnicowy opisujący własności różnicowe skrzynki podstawieniowej wykorzystanej w algorytmie PRESENT zaprezentowano w tabeli 2.4.

W przypadku operacji liniowych nie wyznacza się tak rozumianego profilu różnicowego, ponieważ zachowana jest dla nich następująca własność:

$$\mathbb{L}(x) \oplus \mathbb{L}(y) = \mathbb{L}(x \oplus y), \text{ gdzie } \mathbb{L} \text{ to zadana operacja liniowa,}$$

Tablica 2.4: Zlogarytmowany profil różnicowy skrzynki podstawieniowej wykorzystywanej w szyfrze blokowym PRESENT (postać skrzynki zaprezentowano w tabeli nr 2.2).

$\Delta_{in} \backslash \Delta_{out}$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	2	-1	-1	-1	2	-1	2	-1	-1	-1	2	-1	-1
2	-1	-1	-1	3	-1	2	3	-1	-1	-1	3	-1	3	3	3	-1
3	-1	3	-1	3	3	-1	2	3	-1	-1	3	3	-1	-1	-1	-1
4	-1	-1	-1	-1	-1	2	3	3	-1	3	3	-1	3	-1	3	-1
5	-1	3	-1	-1	3	-1	-1	-1	-1	3	3	3	2	3	-1	-1
6	-1	-1	3	-1	-1	-1	3	-1	3	-1	-1	2	3	-1	-1	2
7	-1	2	3	-1	-1	-1	3	-1	3	-1	-1	-1	3	-1	-1	2
8	-1	-1	-1	3	-1	-1	-1	3	-1	3	-1	2	-1	3	-1	2
9	-1	-1	3	-1	2	-1	3	-1	3	-1	-1	-1	3	-1	2	-1
A	-1	-1	3	3	-1	2	-1	-1	3	-1	3	-1	-1	3	3	-1
B	-1	3	-1	-1	3	-1	-1	-1	2	3	3	3	-1	3	-1	-1
C	-1	-1	3	-1	-1	2	-1	3	3	3	3	-1	-1	-1	3	-1
D	-1	3	2	3	3	-1	-1	3	-1	-1	3	3	-1	-1	-1	-1
E	-1	-1	3	3	-1	-1	3	3	3	3	-1	-1	3	3	-1	-1
F	-1	2	-1	-1	2	-1	-1	-1	-1	-1	-1	-1	-1	-1	2	2

wynikająca z liniowości przekształcenia. Zamiast tego określa się tzw. dyfuzyjny profil różnicowy, w którym nie wszystkie bity różnicy są jednoznacznie określone (charakterystyka różnicowa obcięta). Charakterystyka różnicowa agregowana jest do zbioru symboli określających postać kolejnych bitów. Zwyczajowo przyjmuje się, że taka charakterystyka składa się z symbolu aktywnego ($\neq 0$) oznaczonego symbolem * oraz symbolu nieaktywnego ($= 0$) oznaczonego symbolem 0. Takie podejście sprawia, że prawdopodobieństwo zachowania symbolu przy przejściu charakterystyki różnicowej obciętej przez skrzynkę podstawieniową jest równie 1 (zdarzenie pewne), oczywiście o ile symbole agregowane są na słowach o rozmiarze odpowiadającym rozmiarowi skrzynki podstawieniowej (liczba bitów). Dyfuzyjnym profilem różnicowym można posłużyć się podczas poszukiwania obciętych i niemożlwych charakterystyk różnicowych. Przykładowy dyfuzyjny profil różnicowy operacji *MDS* wykorzystywanej w algorytmie AES zorientowany na bajtowe słowa zaprezentowano w tabeli nr 2.5 oraz 2.6.

Tablica 2.5: Dyfuzyjny profil różnicowy (zorientowany bajtowo) operacji *MDS* wykorzystywanej w algorytmie AES.

$\Delta_{in} \backslash \Delta_{out}$	0000	000*	00*0	00**	0*00	0*0*	0**0	0***	*000	*00*	*0*0	*0**	**00	**0*	***0	****
0000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
000*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255
00*0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255
00**	0	0	0	0	0	0	0	255	0	0	0	255	0	255	255	64770
0*00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255
0*0*	0	0	0	0	0	0	0	255	0	0	0	255	0	255	255	64770
0**0	0	0	0	0	0	0	0	255	0	0	0	255	0	255	255	64770
0***	0	0	0	255	0	255	255	64770	0	255	255	64770	255	64770	64770	16322040
*000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255
00	0	0	0	0	0	0	0	255	0	0	0	255	0	255	255	64770
*0*0	0	0	0	0	0	0	0	255	0	0	0	255	0	255	255	64770
*0**	0	0	0	255	0	255	255	64770	0	255	255	64770	255	64770	64770	16322040
**00	0	0	0	0	0	0	0	255	0	0	0	255	0	255	255	64770
**0*	0	0	0	255	0	255	255	64770	0	255	255	64770	255	64770	64770	16322040
***0	0	0	0	255	0	255	255	64770	0	255	255	64770	255	68608	64770	16322040
****	0	255	255	64770	255	64770	64770	16322040	255	64770	64770	16322040	64770	16322040	16322040	4162573845

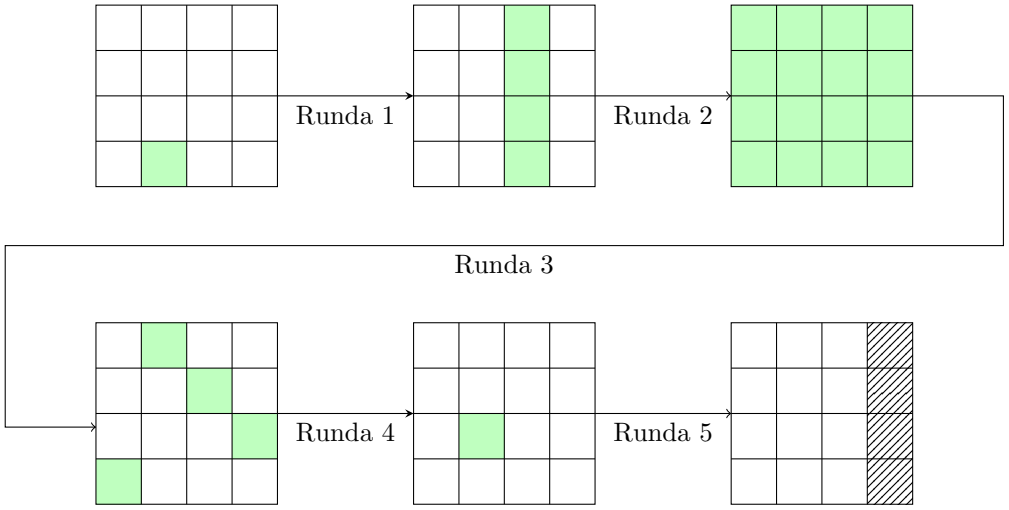
Tablica 2.6: Dyfuzyjny profil różnicowy (zorientowany bajtowo) operacji *MIxcolumns* wykorzystywanej w algorytmie AES — prawdopodobieństwo przejścia charakterystyki różnicowej wejściowej Δ_{in} na charakterystykę wyjściową Δ_{out}.

$\Delta_{in} \backslash \Delta_{out}$	0000	000*	00*0	00**	0*00	0*0*	0**0	0***	*000	*00*	*0*0	*0**	**00	**0*	***0	****
0000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0000
000*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0000
00*0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0000
00**	0	0	0	0	0	0	0	$3.9216e^{-3}$	0	0	0	$3.9216e^{-3}$	0	$3.9216e^{-3}$	$3.9216e^{-3}$	0.9961
0*00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0000
0*0*	0	0	0	0	0	0	0	$3.9216e^{-3}$	0	0	0	$3.9216e^{-3}$	0	$3.9216e^{-3}$	$3.9216e^{-3}$	0.9961
0**0	0	0	0	0	0	0	0	$3.9216e^{-3}$	0	0	0	$3.9216e^{-3}$	0	$3.9216e^{-3}$	$3.9216e^{-3}$	0.9961
0***	0	0	0	$1.538e^{-5}$	0	$1.538e^{-5}$	$1.538e^{-5}$	$3.9062e^{-3}$	0	$1.538e^{-5}$	$1.538e^{-5}$	$3.9062e^{-3}$	$1.538e^{-5}$	$3.9062e^{-3}$	$3.9062e^{-3}$	0.9844
*000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0000
00	0	0	0	0	0	0	0	$3.9216e^{-3}$	0	0	0	$3.9216e^{-3}$	0	$3.9216e^{-3}$	$3.9216e^{-3}$	0.9961
*0*0	0	0	0	0	0	0	0	$3.9216e^{-3}$	0	0	0	$3.9216e^{-3}$	0	$3.9216e^{-3}$	$3.9216e^{-3}$	0.9961
*0**	0	0	0	$1.538e^{-5}$	0	$1.538e^{-5}$	$1.538e^{-5}$	$3.9062e^{-3}$	0	$1.538e^{-5}$	$1.538e^{-5}$	$3.9062e^{-3}$	$1.538e^{-5}$	$3.9062e^{-3}$	$3.9062e^{-3}$	0.9844
**00	0	0	0	0	0	0	0	$3.9216e^{-3}$	0	0	0	$3.9216e^{-3}$	0	$3.9216e^{-3}$	$3.9216e^{-3}$	0.9961
**0*	0	0	0	$1.538e^{-5}$	0	$1.538e^{-5}$	$1.538e^{-5}$	$3.9062e^{-3}$	0	$1.538e^{-5}$	$1.538e^{-5}$	$3.9062e^{-3}$	$1.538e^{-5}$	$3.9062e^{-3}$	$3.9062e^{-3}$	0.9844
***0	0	0	0	$1.538e^{-5}$	0	$1.538e^{-5}$	$1.538e^{-5}$	$3.9062e^{-3}$	0	$1.538e^{-5}$	$1.538e^{-5}$	$3.9062e^{-3}$	$1.538e^{-5}$	$4.1376e^{-3}$	$3.9062e^{-3}$	0.9844
****	0	$6e^{-8}$	$6e^{-8}$	$1.532e^{-5}$	$6e^{-8}$	$1.532e^{-5}$	$1.532e^{-5}$	$3.8602e^{-3}$	$6e^{-8}$	$1.532e^{-5}$	$1.532e^{-5}$	$3.8602e^{-3}$	$1.532e^{-5}$	$3.8602e^{-3}$	$3.8602e^{-3}$	0.9845

2.6.5 Minimalna liczba aktywnych skrzynek podstawieniowych w sieci SPN

Szyfry blokowe bardzo często wykorzystują operacje podstawienia realizowaną za pomocą skrzynki podstawieniowej (ang. *substitution box*) nazywanej potocznie S-box. Operacja podstawienia polega na pobraniu m bitów wejściowych i zwróceniu n bitów wyjściowych. Zazwyczaj sposób podstawienia definiowany jest poprzez określoną w specyfikacji algorytmu tablicę (np. w szyfrach AES oraz DES) zawierającą 2^m elementów, niekiedy generowana jest ona dynamicznie na podstawie klucza (np. w szyfrach Blowfish oraz Twofish).

W atakach różnicowych uznaje się, że skrzynka jest aktywna w przypadku pojawienia się na jej wejściu niezerowej różnicy. Minimalna liczba aktywnych skrzynek podstawieniowych w sieci SPN to najmniejsza liczba skrzynek aktywowanych przez niezerową charakterystykę różnicową. Na rysunku nr 2.9 przedstawiono przykładowy schemat obrazujący jak może wyglądać stan wewnętrzny szyfru blokowego AES podczas 5 rund, gdy aktywowana będzie minimalna liczba skrzynek podstawieniowych (26 skrzynek — nie zliczamy skrzynek aktywnych po wykonaniu 5 rund, ponieważ nie wpływają one na prawdopodobieństwo 5-rundowej charakterystyki różnicowej). Minimalną liczbę aktywnych skrzynek można analogicznie określić również w przypadku kryptoanalizy liniowej.



Rysunek 2.9: Minimalna liczba aktywnych skrzynek podstawieniowych przy 5 rundach algorytmu AES.

Określenie minimalnej liczby aktywnych skrzynek podstawieniowych pozwala oszacować maksymalna wartość prawdopodobieństwa charakterystyki różnicowej. Jest to możliwe, ponieważ propagacja charakterystyki jest ściśle związana z omawianą własnością oraz z profilem różnicowym skrzynki podstawieniowej [rozdz. 2.6.4]. W [55] zaproponowano automatyczną metodę wyznaczania minimalnej liczby aktywnych skrzynek podstawieniowych opartą na modelu MILP i zaaplikowano ją do szyfru blokowego AES. Metoda świetnie sprawdza się w przypadku, gdy w algorytmie przetwarzane są słowa o niewielkim rozmiarze (np.

8 lub 4-bitowe). Metoda ta została udoskonalona w [63] i wykorzystana do poszukiwania charakterystyk różnicowych dla szyfrów blokowych orientowanych bitowo.

2.6.6 Reprezentacja szyfru blokowego za pomocą układu równań nieliniowych nad ciałem binarnym

Współczesne szyfry blokowe oparte są przede wszystkim na szeregu małych funkcji przetwarzających blok danych i złożonych ze sobą w celu konstrukcji tzw. funkcji rundy. Runda algorytmu iterowana jest odpowiednią ilość razy tak, aby uzyskać dostatecznie wysoki poziom dyfuzji i konfuzji.

Każde odwzorowanie wykorzystane w szyfrach blokowych może być zaprezentowane w postaci funkcji *boolowskiej* co w konsekwencji umożliwia przedstawienie całego algorytmu jako szeregu równań nad ciałem binarnym F_2. W przypadku kryptoanalizy algebraicznej i metod bazujących na rozwiązaniu takiego układu w celu odzyskania tajnego klucza zmienne opisujące klucz lub klucze rundowe traktujemy jako niewiadome. Oczywiście, jeżeli model ataku zakłada również nieznajomość tekstu jawnego, zmienne te również traktowane są jako nieznane. W praktycznych zastosowaniach takie metody kryptoanalizy zakładają jednak znajomość tekstów jawnych (lub ich fragmentów) a niekiedy nawet możliwość ich doboru.

Tablica 2.7: Przykładowa 4-bitowa skrzynka podstawieniowa przedstawiona w formie tabeli.

wejście	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
wyjście	7	4	A	9	1	F	B	0	C	3	2	6	8	E	D	5

Kod źródłowy 2.1: Program do przedstawienia skrzynki podstawieniowej za pomocą funkcji *boolowskiej*.

```
1  from sage.crypto.sbox import SBox
2
3  if __name__ == "__main__":
4
5      S = SBox(
6          0x7, 0x4, 0xA, 0x9, 0x1, 0xF, 0xB, 0x0,
7          0xC, 0x3, 0x2, 0x6, 0x8, 0xE, 0xD, 0x5
8      )
9
10     P.<y0,y1,y2,y3,x0,x1,x2,x3> = PolynomialRing(GF(2), 8, order='lex')
11
12     equations = S.polynomials(
13         [x0,x1,x2,x3],[y0,y1,y2,y3], groebner=True
14     )
15
16     for eq in equations:
17         print(eq)
```

Wynikiem działania programu opisanego przez kod źródłowy 2.1 będzie wypisanie na ekran następującego zestawu wielomianów:

$$y_0 + x_0 \cdot x_1 \cdot x_2 + x_0 \cdot x_2 \cdot x_3 + x_0 \cdot x_3 + x_0 + x_1 \cdot x_3 + x_2$$
$$y_1 + x_0 \cdot x_1 \cdot x_2 + x_0 \cdot x_1 \cdot x_3 + x_0 \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + x_1 \cdot x_2 + x_1 \cdot x_3 + x_1 + x_2 + 1$$
$$y_2 + x_0 \cdot x_1 + x_0 \cdot x_2 \cdot x_3 + x_0 \cdot x_2 + x_0 + x_1 \cdot x_2 + x_1 + x_3 + 1$$
$$y_3 + x_0 \cdot x_2 \cdot x_3 + x_0 \cdot x_2 + x_0 + x_1 \cdot x_2 \cdot x_3 + x_1 \cdot x_2 + x_1 \cdot x_3 + x_2 + x_3 + 1$$

Zaprezentowanie algorytmu kryptograficznego jako układu równań nad ciałem binarnym umożliwia wykorzystanie szeregu technik związanych z logika binarną. Najczęściej wykorzystywana jest szeroko rozumiana minimalizacji funkcji *boolowskiej*, a niekiedy równania są sztucznie skracane poprzez wprowadzanie dodatkowych zmiennych. Poniżej znajduje się zestaw równań opisujący operacje *Mixcolumn* z algorytmu `AES`.

$$y_0 = x_1 \wedge x_8 \wedge x_9 \wedge x_{16} \wedge x_{24}$$
$$y_1 = x_2 \wedge x_9 \wedge x_{10} \wedge x_{17} \wedge x_{25}$$
$$y_2 = x_3 \wedge x_{10} \wedge x_{11} \wedge x_{18} \wedge x_{26}$$
$$y_3 = x_0 \wedge x_4 \wedge x_8 \wedge x_{11} \wedge x_{12} \wedge x_{19} \wedge x_{27}$$
$$y_4 = x_0 \wedge x_5 \wedge x_8 \wedge x_{12} \wedge x_{13} \wedge x_{20} \wedge x_{28}$$
$$y_5 = x_6 \wedge x_{13} \wedge x_{14} \wedge x_{21} \wedge x_{29}$$
$$y_6 = x_0 \wedge x_7 \wedge x_8 \wedge x_{14} \wedge x_{15} \wedge x_{22} \wedge x_{30}$$
$$y_7 = x_0 \wedge x_8 \wedge x_{15} \wedge x_{23} \wedge x_{31}$$
$$y_8 = x_0 \wedge x_9 \wedge x_{16} \wedge x_{17} \wedge x_{24}$$
$$y_9 = x_1 \wedge x_{10} \wedge x_{17} \wedge x_{18} \wedge x_{25}$$
$$y_{10} = x_2 \wedge x_{11} \wedge x_{18} \wedge x_{19} \wedge x_{26}$$
$$y_{11} = x_3 \wedge x_8 \wedge x_{12} \wedge x_{16} \wedge x_{19} \wedge x_{20} \wedge x_{27}$$
$$y_{12} = x_4 \wedge x_8 \wedge x_{13} \wedge x_{16} \wedge x_{20} \wedge x_{21} \wedge x_{28}$$
$$y_{13} = x_5 \wedge x_{14} \wedge x_{21} \wedge x_{22} \wedge x_{29}$$
$$y_{14} = x_6 \wedge x_8 \wedge x_{15} \wedge x_{16} \wedge x_{22} \wedge x_{23} \wedge x_{30}$$
$$y_{15} = x_7 \wedge x_8 \wedge x_{16} \wedge x_{23} \wedge x_{31}$$
$$y_{16} = x_0 \wedge x_8 \wedge x_{17} \wedge x_{24} \wedge x_{25}$$
$$y_{17} = x_1 \wedge x_9 \wedge x_{18} \wedge x_{25} \wedge x_{26}$$
$$y_{18} = x_2 \wedge x_{10} \wedge x_{19} \wedge x_{26} \wedge x_{27}$$
$$y_{19} = x_3 \wedge x_{11} \wedge x_{16} \wedge x_{20} \wedge x_{24} \wedge x_{27} \wedge x_{28}$$
$$y_{20} = x_4 \wedge x_{12} \wedge x_{16} \wedge x_{21} \wedge x_{24} \wedge x_{28} \wedge x_{29}$$
$$y_{21} = x_5 \wedge x_{13} \wedge x_{22} \wedge x_{29} \wedge x_{30}$$
$$y_{22} = x_6 \wedge x_{14} \wedge x_{16} \wedge x_{23} \wedge x_{24} \wedge x_{30} \wedge x_{31}$$
$$y_{23} = x_7 \wedge x_{15} \wedge x_{16} \wedge x_{24} \wedge x_{31}$$
$$y_{24} = x_0 \wedge x_1 \wedge x_8 \wedge x_{16} \wedge x_{25}$$
$$y_{25} = x_1 \wedge x_2 \wedge x_9 \wedge x_{17} \wedge x_{26}$$
$$y_{26} = x_2 \wedge x_3 \wedge x_{10} \wedge x_{18} \wedge x_{27}$$
$$y_{27} = x_0 \wedge x_3 \wedge x_4 \wedge x_{11} \wedge x_{19} \wedge x_{24} \wedge x_{28}$$
$$y_{28} = x_0 \wedge x_4 \wedge x_5 \wedge x_{12} \wedge x_{20} \wedge x_{24} \wedge x_{29}$$

$y_{29} = x_5 \wedge x_6 \wedge x_{13} \wedge x_{21} \wedge x_{30}$
$y_{30} = x_0 \wedge x_6 \wedge x_7 \wedge x_{14} \wedge x_{22} \wedge x_{24} \wedge x_{31}$
$y_{31} = x_0 \wedge x_7 \wedge x_{15} \wedge x_{23} \wedge x_{24}$

Zauważmy, że wyrażenie $x_8 \wedge x_{16}$ powtarza się i można zastąpić je nową zmienną $t_0 = x_8 \wedge x_{16}$. W praktyce zmniejszy to liczbę operacji `XOR`, które należy wykonać, aby wyznaczyć wynik operacji *Mixcolumn*. Poszukiwanie takich przekształceń, które mają pożądane własności kryptograficzne, a jednocześnie da się je zaimplementować za pomocą minimalnego nakładu obliczeniowego (mała liczba bramek `XOR`, `AND` itp.) również jest ważnym aspektem badań związanych z projektowaniem algorytmów kryptograficznych.

2.6.7 Wykorzystanie grafów `AIG` do reprezentacji szyfru blokowego

Graf `AIG` (ang. And Inverter Graph) jest to acykliczny graf skierowany, który reprezentuje strukturalną implementację logicznej funkcjonalności układu. Graf składa się z wierzchołków posiadających dwie krawędzie wejściowe (odpowiada koniunkcji logicznej, bramce logicznej `AND`), wierzchołków końcowych oznaczonymi nazwami zmiennych oraz opcjonalnie krawędzi ze znacznikiem (odpowiadającymi negacji binarnej, bramce logicznej `NOT`).

Niech funkcja f trzech zmiennych x_0, x_1, x_2 będzie określona następującym wzorem:

$$f(x_0, x_1, x_2) = (y_0, y_1, y_2)$$

$$y_0 = \text{NOT } (x_0 \text{ AND } x_1)$$

$$y_1 = ((\text{NOT } x_1) \text{ AND } (\text{NOT } x_2)) \text{ XOR } x_0$$

$$y_2 = (\text{NOT } x_0) \text{ AND } x_1$$

Reprezentację funkcji f za pomocą grafu `AIG` przedstawiono na rysunku nr 2.10. Wierzchołki I_0, I_1, I_2 oraz O_0, O_1, O_2 stanowią odpowiednio wejście i wyjście układu logicznego realizującego funkcje f. Pozostałe wierzchołki W_0, W_1, W_2, W_3, W_4, W_5, W_6, W_7, W_8 określają kolejne przekształcenia realizowane w ramach wyznaczenia wartości wyjściowych układu (realizacji funkcji f). Wartości binarne wyznaczane w kolejnych wierzchołkach prezentują się następująco:

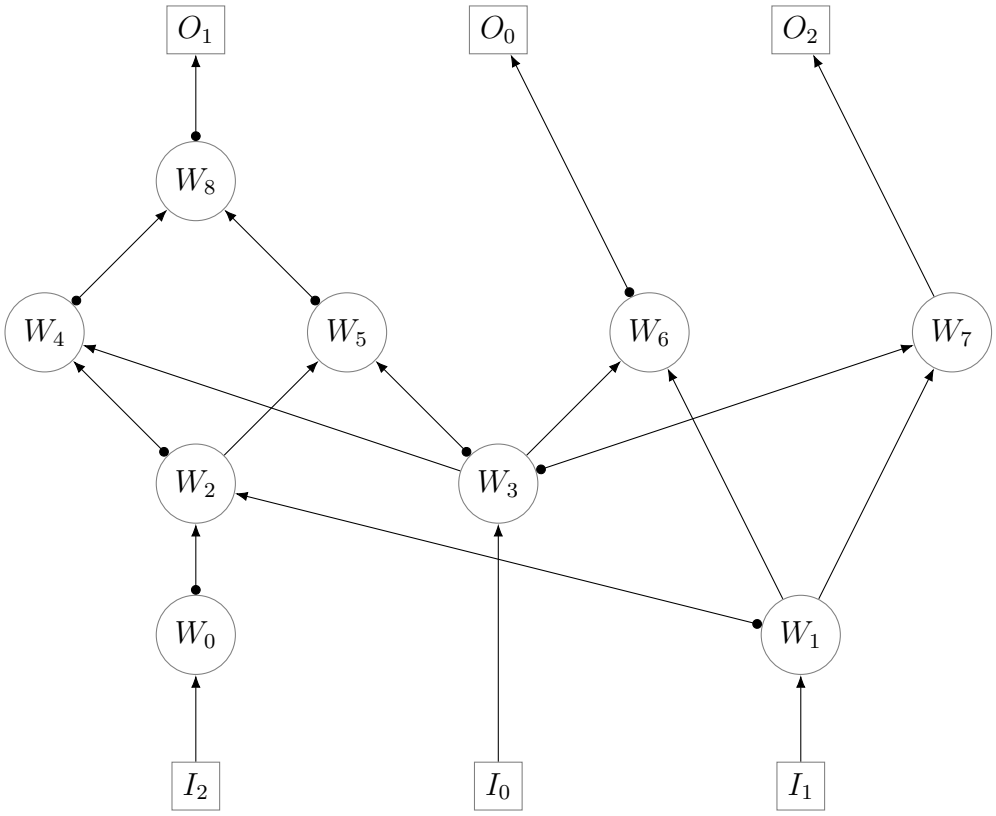
$I_0 = x_0$

$I_1 = x_1$

$I_2 = x_2$

$W_0 = I_2$

$W_1 = I_1$



Rysunek 2.10: Graf `AIG` funkcji f_1.

$W_2 = (\text{NOT } W_0) \text{ AND } (\text{NOT } W_1)$

$W_3 = I_0$

$W_4 = (\text{NOT } W_2) \text{ AND } W_3$

$W_5 = W_2 \text{ AND } (\text{NOT } W_3)$

$W_6 = W_3 \text{ AND } W_1$

$W_7 = (\text{NOT } W_3) \text{ AND } W_1$

$W_8 = (\text{NOT } W_4) \text{ AND } (\text{NOT } W_5)$

$O_0 = \text{NOT } W_6$

$O_1 = W_7$

$O_2 = \text{NOT } W_8$

$y_0 = O_0$

$y_1 = O_1$

$y_2 = O_2$

Przedstawiony graf można opisać również za pomocą następujących wskaźników:

(a) liczba bramek `AND`,

(b) głębokość grafu,

(c) liczba wierzchołków.

Reprezentacja struktury logicznej za pomocą grafu `AIG` posiada szereg zalet, z których warto wyróżnić:

- łatwy w budowie i interpretacji, zwięzły i prosty zapis (10^6 wierzchołków $\approx$ 12 MB pamięci `RAM`),
- może być skutecznie przechowywany na dysku (10^6 wierzchołków $\approx$ plik 4 MB),
- unormowany zapis (może być przenoszona pomiędzy różnymi narzędziami),
- model wspierany przez niektóre *SAT solvery* np. Cadical, Lingeling [9].

Podczas badań związanych z niniejszą rozprawą doktorską grafy `AIG` wykorzystywano do reprezentacji szyfrów blokowych (lub ich części np. funkcji rundy czy też wykorzystanych w niej prymitywów kryptograficznych) oraz reprezentacji struktury logicznej opisującej propagację charakterystyk różnicowych. Dzięki narzędziom takim jak `SAW` [21], które umożliwiają konwersję kodu źródłowego (np. z języka `Cryptol` [34] czy też `C`) do grafu `AIG` możliwa jest automatyzacja procesów związanych z reprezentacją badanych przekształceń kryptograficznych za pomocą układu równań nad ciałem binarnym. W uproszczeniu można przyjąć, że kryptoanalityk musi dysponować jedynie kodem źródłowym badanego przekształcenia.

Dodatkową zaletą wykorzystania grafów `AIG` jest możliwość wykonania tzw. redukcji funkcjonalnej (ang. Functionally Reduced And Inverters Graphs [53]), która minimalizuje graf pod względem liczby wierzchołków, a tym samym układ logiczny przez niego reprezentowany (minimalizacja liczby bramek logicznych `AND`). Operacja ta w przypadku niektórych przekształceń kryptograficznych znacznie zmniejsza rozmiar grafu, a tym samym minimalizacji ulega również układu równań nad ciałem binarnym, który wykorzystywany jest w dalszych analizach.

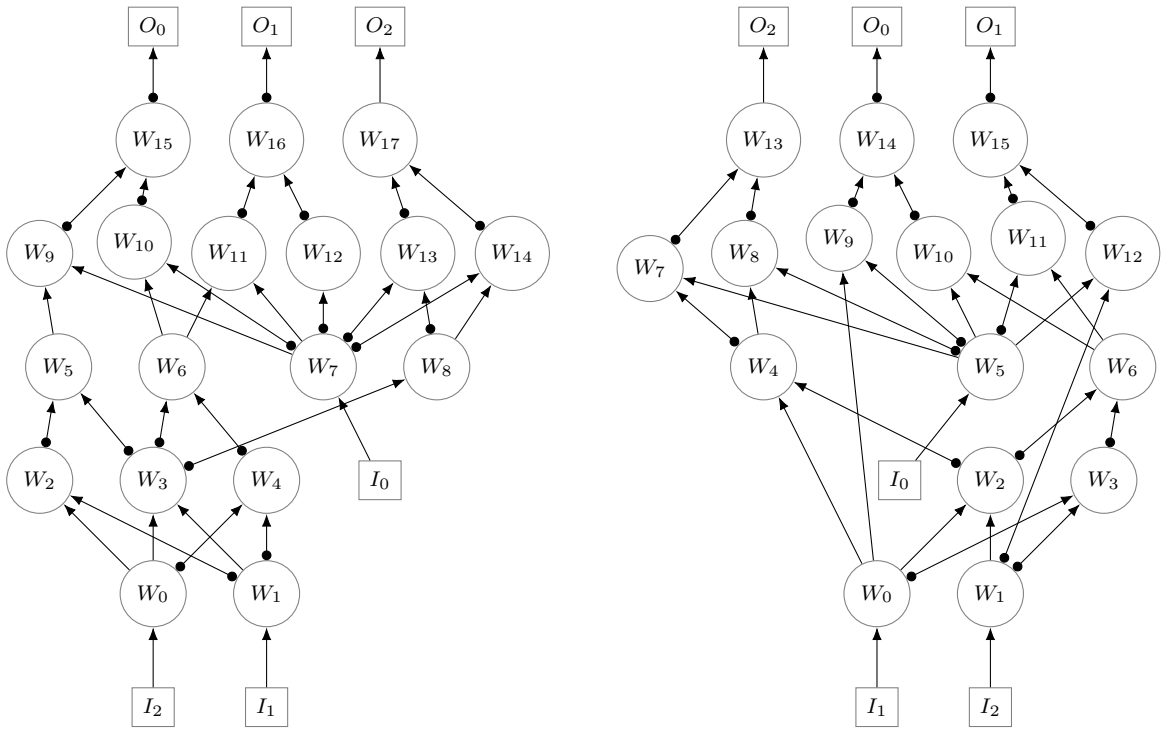
Dla lepszego zobrazowania przydatności zastosowania redukcji funkcjonalnej posłużmy się przekształceniem `S` postaci:

$$S(x) = y \text{ dla } x, y \in GF(2^3)$$

Reprezentację przekształcenia `S` w formie grafu `AIG` przedstawiono na rysunku nr 2.11. Oryginalny graf składający się z 18 wierzchołków został zredukowany do 16 wierzchołków (został zminimalizowany o $11,(11)\%$ pod względem liczby bramek `AND`).

Tablica 2.8: Przekształcenie `S` zaprezentowane w formie tablicy prawdy.

x	0	1	2	3	4	5	6	7
y	1	3	6	5	2	4	7	0



(a) Graf oryginalny.

(b) Graf po redukcji funkcjonalnej.

Rysunek 2.11: Grafy `AIG` reprezentujące strukturę logiczną układu realizującego przekształcenie `S`.

Tablica 2.9: Liczba bramek AND w grafach AIG opisujących propagację 5 rundowej charakterystyki różnicowej w wybranych szyfrach blokowych generowanych przy wykorzystaniu języka funkcyjnego Cryptol[34], narzędzia SAW[21] oraz ABC[18]

Nazwa szyfru	**Rozmiar bloku**	**Liczba bramek AND w grafie AIG**	**Liczba bramek AND po redukcji funkcjonalnej grafu AIG**	**% redukcji podstawowego grafu AIG**
AES	128	1463089	1153242	21.18
KLEIN	64	27265	17781	34.78
MIDORI	64	25985	16277	37.36
PRESENT	64	24561	15083	38.59
PYJAMASK	128	61849	36689	40.68
PYJAMASK	96	36646	17721	51.64
SPECK	32	4911	2003	59.21
SATURNIN	256	241644	78518	67.51
SPECK	64	13718	4327	68.46
SPECK	96	22550	6887	69.46
SPECK	128	31382	9530	69.63
SIMON	32	4675	1380	70.48
SIMON	64	12734	3027	76.23
SIMON	96	20798	4921	76.34
SIMON	128	28862	6847	76.28

Strona celowo pozostawiona pusta.

Rozdział 3

Metody kryptoanalizy współczesnych szyfrów blokowych

3.1 Podstawowe typy ataków kryptograficznych

Podstawowe założenie współczesnej kryptoanalizy zostały sformułowane już w 1883 roku przez holenderskiego językoznawcę i kryptologa Auguste Kerckhoffsa i mówi o tym, że przeciwnik zna zastosowany kryptosystem (szyfr), a nie zna jedynie klucza wykorzystanego do utajnienia wiadomości [3]. Dodatkowo przyjmuje się, że kryptoanalityk ma dostęp do wszystkich danych (szyfrogramów) transmitowanych przez kanał jawny. Przedstawione przez A. Kerckhoffsa założenie na przestrzeni lat nie uległo zmianie.

Ze względu na dostępność danych wymaganych do kryptoanalizy wyklarowały się pewne typy ataków kryptograficznych.

Atak, gdy znany jest wyłącznie szyfrogram — (ang. *ciphertext only*) scenariusz, w którym kryptoanalityk przechwycił grupę szyfrogramów i dysponuje ogólnymi informacjami na temat przesyłanych tekstów jawnych. Celem jest odtworzenie tekstu jawnego i\lub tajnego klucza.

Atak, gdy znany jest tekst jawny — (ang. *known plaintext*) scenariusz, w którym kryptoanalityk zna pewne pary tj. tekst jawny i odpowiadający mu szyfrogram. Celem jest odtworzenie tajnego klucza, który mógł być wykorzystany do zdeszyfrowania innych wiadomości.

Atak z wybranym tekstem jawnym — (ang. *chosen ciphertext*) scenariusz, w którym kryptoanalityk ma możliwość szyfrowania wybranych przez siebie tekstów jawnych. W ten sposób może budować bazę wiedzy na temat tekstów jawnych i odpowiadających im szyfrogramów. Celem jest odtworzenie tajnego klucza.

Atak adaptacyjny z wybranym tekstem jawnym — (ang. *adaptively chosen plaintext*) scenariusz, w którym kryptoanalityk ma możliwość wybrania tekstów jawnych na podstawie wcześniej zebranych szyfrogramów. Celem jest odtworzenie tajnego klucza.

Atak z wybranym tekstem zaszyfrowanym — (ang. *chosen ciphertext*) scenariusz, w którym kryptoanalityk ma okresowy dostęp do urządzenia deszyfrującego (może wybrać szyfrogram i odpowiadający mu tekst jawny). Celem jest odtworzenie tajnego klucza, który mógł być wykorzystany do zdeszyfrowania innych wiadomości.

Atak wykorzystujący zależność kluczy — (ang. *related keys*) sytuacja, w której kryptoanlityk może obserwować działanie szyfru na kilku różnych kluczach, których wartości są początkowo nieznane, ale w przypadku których atakujący zna pewien matematyczny związek łączący te klucze.

3.2 Kryptoanaliza różnicowa

Kryptoanaliza różnicowa to uniwersalna metoda kryptoanalizy przeznaczona przede wszystkim do szyfrów blokowych i funkcji skrótu, rzadziej wykorzystywana w przypadku szyfrów strumieniowych. W najprostszym ujęciu polega na obserwacji, w jaki sposób zachowują się informacje (szyfrowane tym samym kluczem), różniące się w szczególny, wybrany przez kryptoanalityka sposób. Zwyczajowo charakterystykę różnicową określa się poprzez operację `XOR`. Jest to metoda kryptoanalizy z wybranym tekstem jawnym, tzn. że istnieje możliwość wybierania wejścia (tekstów jawnych) i obserwacji wyjścia (szyfrogramów) w celu odzyskania klucza.

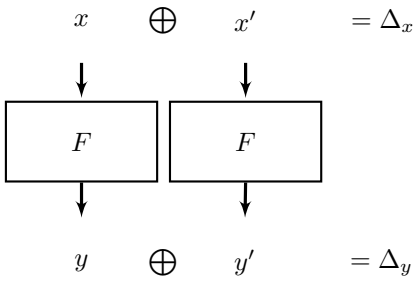
Upublicznienie metody kryptoanalizy różnicowej w 1990 roku przez *Eli'ego Bihama* i *Adi'ego Shamir* było przełomowym wydarzeniem dla kryptoanalizy. Metoda ta umożliwiła złamanie wielu ówczesnych szyfrów np. `FEAL` [13]. W pracy [14] *E. Biham* i *A. Shamir* zauważyli, że dotychczasowy standard szyfrowania `DES` jest zaskakująco odporny na kryptoanalizę różnicową, a niewielkie modyfikacje uczyniłyby go bardzo podatnym. Ostatecznie amerykańska agencja wywiadowcza `NSA` oświadczyła, że ta metoda znana jej była już w latach siedemdziesiątych. Dlatego skrzynki podstawieniowe w ówczesnym standardzie szyfrowania symetrycznego (`DES`) zaprojektowano tak, aby uodpornić algorytm na ten rodzaj ataku. W tym celu zaproponowano również nie mniej niż 16 rund.

W ataku różnicowych możemy wykorzystać tzw. różniczkę (ang. *the difference*) (Δ_x, Δ_y) określoną poprzez różnicę początkową Δ_x (różnicę tekstów jawnych) oraz końcową Δ_y (różnicę szyfrogramów). Uzyskanie różnicy końcowej Δ_y przy różnicy początkowej Δ_x możliwe jest z pewnym prawdopodobieństwem p. Jeżeli dla funkcji F dysponujemy różniczką (Δ_x, Δ_y), dla której uzyskanie różnicy końcowej Δ_y przy różnicy początkowej Δx jest możliwe z prawdopodobieństwem $p \neq 0$ to istnieje przynajmniej jedna wartość x, dla której prawdziwe jest następujące równanie:

$$F(x \oplus \Delta_x) \oplus F(x) = \Delta_y.$$

Gdy funkcja F odpowiada funkcji szyfrowania w algorytmie blokowym, a prawdopodobieństwo p jest statystycznie istotne to różnicę szyfrogramów Δ_y uzyskanych w wyniku szyfrowania dwóch tekstów jawnych (tym samym kluczem) dobranych za pomocą różnicy Δ_x można wykorzystać do odtworzenia klucza rundowego. Przedstawiona tu nieformalna

definicja różniczki jest zbieżna z nieformalną definicją przedstawioną przez *E. Bihama* oraz *A. Shamira* w [14]. W tym samym artykule autorzy przytaczają również nieformalną definicję *charakterystyki różnicowej* (ang. *differential characteristic*), nazywanej niekiedy ścieżką różnicową (ang. *differential trial* lub *differential path*).



Rysunek 3.1: Różniczka funkcji F.

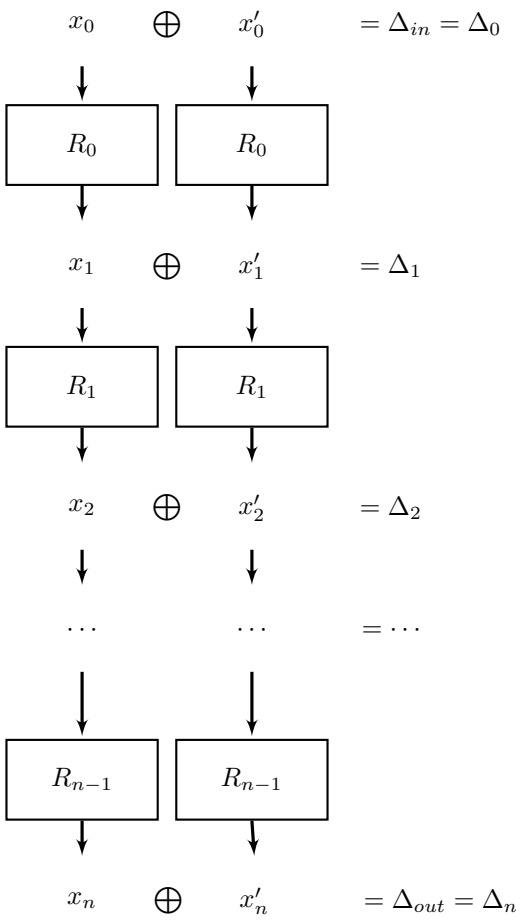
R rundowa charakterystyka różnicowa [rys. 3.2] $(\Delta_0, \cdots, \Delta_R)$ określa dokładnie postać wszystkich wartości pośrednich różniczki (wartości po każdej z rozważanych w ataku rund). Przyjmuje się, że w dobrze zaprojektowanym szyfrze blokowym prawdopodobieństwo optymalnej charakterystyki różnicowej powinno być zbliżone do prawdopodobieństwa optymalnej różniczki.

Współcześnie za najbardziej efektywne techniki kryptoanalizy różnicowej uznaje się kryptoanalizę różnicową z wykorzystaniem charakterystyk różnicowych obciętych (ang. *truncuted differential cryptanalysis*) oraz niemożliwych (ang. *impossible differential cryptanalysis*).

Kryptoanaliza różnicowa z wykorzystaniem charakterystyk różnicowych obciętych jest uogólnieniem tradycyjnej kryptoanalizy różnicowej. Technika ta została zaprezentowana przez Larsa Knudsena w 1994 roku [39]. Podczas gdy tradycyjna kryptoanaliza różnicowa skupia się na różnicach między dwoma tekstami jawnymi, kryptoanaliza z wykorzystaniem różniczek obciętych uwzględnia różnice, które są tylko częściowo określone (atak przewiduje tylko niektóre bity charakterystyki różnicowej zamiast całego bloku). Metoda została zastosowana do kryptoanalizy takich szyfrów blokowych jak `Camellia`, `IDEA`, `SAFER`, `IDEA NXT`, `CRYPTON`, `Skipjack`, `E2`, `Twofish`, a nawet do szyfru strumieniowego `Salsa20`.

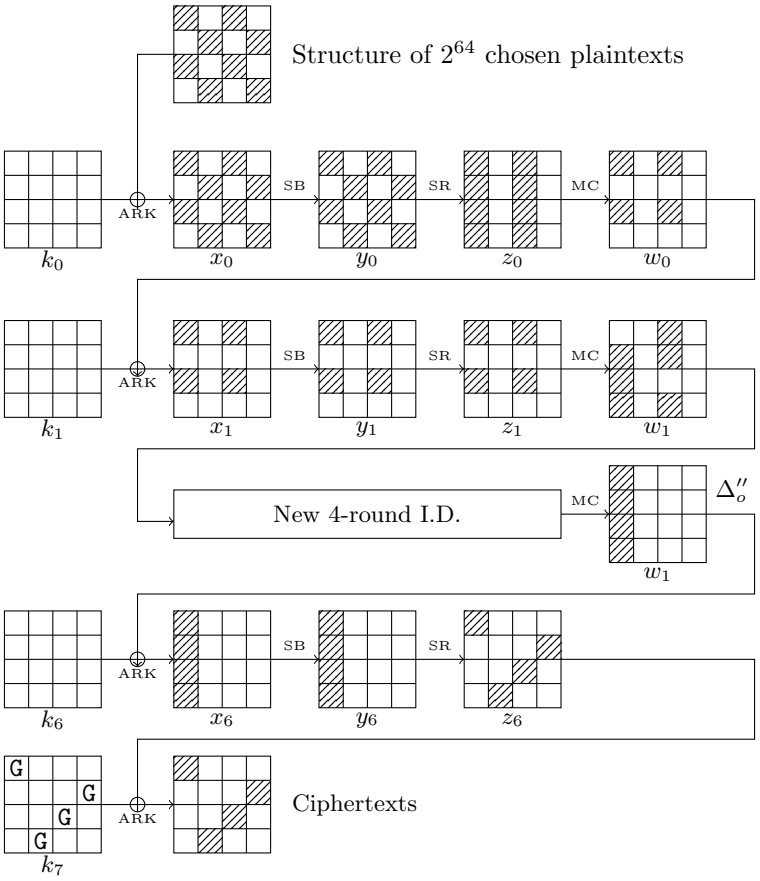
Kryptoanaliza różnicowa z wykorzystaniem różniczek niemożliwych jest techniką analizy szyfrów, która w odróżnieniu od kryptoanalizy różnicowej wykorzystującej w pełni określone i obcięte charakterystyki różnicowe nie skupia się na śledzeniu propagacji charakterystyk różnicowych, które można odróżnić od szumu. W przeciwieństwie do wspomnianych technik kryptoanaliza różnicowa z wykorzystaniem różniczek niemożliwych wykorzystuje charakterystyki różnicowe, które są niemożliwe do osiągnięcia (tzn. mające prawdopodobieństwo równe 0) w pewnym stanie pośrednim algorytmu szyfrującego. Omawiana technika kryptoanalizy została zaprezentowana po raz pierwszy w 1998 roku przez Larsa Knudsena [40] oraz na konferencji `CRYPTO'98` [59], gdzie wykorzystano ją do kryptoanalizy szyfru `IDEA` [12] oraz `Skipjack` [11]. Według wielu spekulacji nawet `NSA` nie znało wcześniej

tej techniki kryptoanalizy. Obecnie technikę tę próbuje się wykorzystać do kryptoanalizy takich szyfrów jak `Rijndael`, `MARS`, `Serpent`, `Camellia`, `ARIA`, `Twofish`, `Khufu`, `Khafre`, `E2`, `Zodiac`, `CRYPTON`, `TEA`, `XTEA`, `Hierocrypt-3` oraz `SHACAL-2` [42] [17] [43].



Rysunek 3.2: Charakterystyka różnicowa postaci $(\Delta_0, \Delta_1, \Delta_2, \cdots, \Delta_n)$

W szyfrach blokowych opartych na sieci `SPN` [rozdz. 2.2.1] własności różnicowe całego algorytmu są ściśle związane z wykorzystaną w nich skrzynką podstawieniową, a konkretnie z jej *profilem różnicowym*. Ważną rolę odgrywa też liniowa warstwa dyfuzyjna. W przypadku charakterystyk różnicowych, które są w pełni określone (klasyczna kryptoanaliza różnicowa), projektant szyfru blokowego może wykorzystać założenia strategii szerokiej ścieżki [rozdz. 2.4] aby oszacować optymalne prawdopodobieństwo charakterystyki różnicowej oraz liniowej. Niestety na chwilę obecną w przypadku kryptoanalizy różnicowej z wykorzystaniem charakterystyk różnicowych obciętych (czy też niemożliwych) nie opracowano metod konstrukcji szyfrów blokowych, które pozwalałyby w jasny sposób przedstawić dowód odporności na te techniki kryptoanalizy.



Rysunek 3.3: Koncepcja ataku z wykorzystaniem charakterystyki różnicowej niemożliwej na 7 rund szyfru blokowego AES

3.3 Kryptoanaliza liniowa

Podobnie jak kryptoanaliza różnicowa (rozdz. 3.2) kryptoanaliza liniowa to ogólna metoda kryptoanalizy. W odróżnieniu jednak od kryptoanalizy różnicowej szersze zastosowanie wykorzystuje w obrębie ataków na szyfry strumieniowe [20]. Rzadziej wykorzystywana jest w przypadku szyfrów blokowych i funkcjach skrótu. Kryptoanaliza liniowa wykorzystuje wysokie prawdopodobieństwo występowania liniowych zależności wiążących tekst jawny, szyfrogram i bity kluczy rundowych. Główną ideą jest aproksymacja nieliniowej części szyfru za pomocą funkcji liniowej. Technika została zaproponowana przez M. Matsui i wykorzystana do kryptoanalizy ówczesnego standardu szyfrowania DES [50].

Główne założenia kryptoanalizy liniowej wyglądają następująco:

- jest to metoda ataku ze znanym tekstem jawnym, w której kryptoanlityk zna pewne pary tekstu jawnego i odpowiadające im szyfrogramy lub ma możliwość szyfrowania wybranych przez siebie par i uzyskiwania w ten sposób odpowiadających im szyfrogramów;

- duża część szyfrowania wykonywana jest z wykorzystaniem operacji liniowych (np. mnożenie przez macierz, rotacje, permutacje);
- bity klucza wiązane są z tekstem jawnym poprzez liniową operację dodawania modulo 2 (`XOR`);
- dla operacji nieliniowych można wyznaczyć liniowe relacje pomiędzy bitami wejściowymi i wyjściowymi zachodzące z prawdopodobieństwem p, dzięki którym możemy wykonać aproksymację liniową przekształcenia nieliniowego (np. skrzynki podstawieniowej);
- istnieją liniowe aproksymacje (równania) bitów klucza, tekstu jawnego i szyfrogramu postaci:
$$P_i \oplus C_j \oplus K_k = 0$$
zachodzące z prawdopodobieństwem $p \neq \frac{1}{2}$.

3.4 Kryptoanaliza algebraiczna

Kryptoanaliza algebraiczna jest metodą ataku na duży podzbiór szyfrów [1] [24]. Składa się z dwóch głównych kroków. Pierwszy polega na przekształceniu szyfru w układ równań, zwykle nad ciałem binarnym [rysunek nr 3.4]. Idea ta nie ogranicza się jednak jedynie do tego konkretnego ciała. W drugim kroku układ równań jest rozwiązywany w celu odtworzenia tajnego klucza używanego do szyfrowania dostarczonej pary tj. tekst jawny – szyfrogram. Istnieje kilka podejść do rozwiązania takiego układu równań, zaczynając od algorytmu `XL` i baz Gröbnera [22] do rozwiązywania układu za pomocą oprogramowania nazywanego potocznie `SAT` solverem. Szczegółowe omówienie podstaw kryptoanalizy algebraicznej możemy odnaleźć m.in. w [6]. Technika ta jest również używana do kryptoanalizy funkcji skrótu [52].

$$
\begin{cases}
0 = x_0 \cdot z_6 + x_0 \cdot z_4 + x_1 \cdot z_5 + x_1 \cdot z_0 + x_2 \cdot z_5 + x_2 \cdot z_3 + x_2 \cdot z_0 + x_3 \cdot z_4 + x_3 \cdot z_7 \\
\quad + x_4 \cdot z_2 + x_4 \cdot z_4 + x_4 \cdot z_7 + x_5 \cdot z_1 + x_5 \cdot z_4 + x_5 \cdot z_3 + x_6 \cdot z_6 + x_6 \cdot z_3 + x_6 \cdot z_1 \\
\quad + x_7 \cdot z_0 + x_7 \cdot z_2 + x_7 \cdot z_3 + x_5 + x_7 + z_1 + z_3 + z_5 + z_7 \\
0 = x_0 \cdot z_0 + x_0 \cdot z_5 + x_0 \cdot z_3 + x_1 \cdot z_4 + x_1 \cdot z_7 + x_2 \cdot z_4 + x_2 \cdot z_2 + x_2 \cdot z_7 + x_3 \cdot z_3 \\
\quad + x_3 \cdot z_4 + x_3 \cdot z_1 + x_4 \cdot z_1 + x_4 \cdot z_3 + x_4 \cdot z_6 + x_5 \cdot z_2 + x_5 \cdot z_0 + x_5 \cdot z_3 + x_6 \cdot z_5 \\
\quad + x_6 \cdot z_0 + x_6 \cdot z_2 + x_6 \cdot z_4 + x_6 \cdot z_6 + x_6 \cdot z_1 + x_7 \cdot z_1 + x_7 \cdot z_2 + x_7 \cdot z_7 \\
\quad + x_3 + x_5 + x_7 + z_7 + z_6 + z_5 + z_4 + z_3 \\
0 = x_0 \cdot z_6 + x_0 \cdot z_5 + x_0 \cdot z_3 + x_0 \cdot z_2 + x_1 \cdot z_4 + x_1 \cdot z_3 + x_1 \cdot z_1 + x_1 \cdot z_0 + x_2 \cdot z_0 \\
\quad + x_2 \cdot z_6 + x_2 \cdot z_5 + x_2 \cdot z_7 + x_2 \cdot z_2 + x_2 \cdot z_1 + x_3 \cdot z_0 + x_3 \cdot z_5 + x_3 \cdot z_7 + x_3 \cdot z_3 \\
\quad + x_4 \cdot z_1 + x_4 \cdot z_6 + x_4 \cdot z_3 + x_4 \cdot z_5 + x_5 \cdot z_7 + x_5 \cdot z_3 + x_5 \cdot z_1 + x_5 \cdot z_4 + x_6 \cdot z_5 \\
\quad + x_6 \cdot z_2 + x_6 \cdot z_1 + x_6 \cdot z_7 + x_7 \cdot z_6 + x_7 \cdot z_1 + x_7 \cdot z_4 + x_7 \cdot z_7 \\
\quad + x_5 + x_3 + x_1 + x_6 + x_7 + z_6 + z_5 + z_3 + z_2 \\
0 = x_0 \cdot z_6 + x_0 \cdot z_1 + x_0 \cdot z_3 + x_1 \cdot z_3 + x_1 \cdot z_0 + x_1 \cdot z_2 + x_2 \cdot z_1 + x_2 \cdot z_2 + x_2 \cdot z_5 \\
\quad + x_2 \cdot z_0 + x_2 \cdot z_4 + x_2 \cdot z_6 + x_3 \cdot z_2 + x_3 \cdot z_1 + x_4 \cdot z_7 + x_4 \cdot z_6 + x_4 \cdot z_0 + x_4 \cdot z_5 \\
\quad + x_4 \cdot z_3 + x_5 \cdot z_6 + x_5 \cdot z_0 + x_5 \cdot z_1 + x_6 \cdot z_6 + x_6 \cdot z_5 + x_6 \cdot z_2 + x_6 \cdot z_4 + x_6 \cdot z_7 \\
\quad + x_7 \cdot z_7 + x_7 \cdot z_5 + x_7 \cdot z_0 + x_5 + x_3 + x_1 + x_4 + z_4 + z_1 + z_3 + z_0 \\
0 = x_0 \cdot z_0 + x_0 \cdot z_5 + x_0 \cdot z_2 + x_1 \cdot z_5 + x_1 \cdot z_2 + x_1 \cdot z_0 + x_1 \cdot z_7 + x_1 \cdot z_1 + x_2 \cdot z_7 \\
\quad + x_2 \cdot z_6 + x_3 \cdot z_0 + x_3 \cdot z_7 + x_3 \cdot z_1 + x_3 \cdot z_6 + x_3 \cdot z_4 + x_4 \cdot z_6 + x_4 \cdot z_5 + x_5 \cdot z_7 \\
\quad + x_5 \cdot z_4 + x_5 \cdot z_3 + x_5 \cdot z_1 + x_5 \cdot z_5 + x_5 \cdot z_0 + x_6 \cdot z_5 + x_6 \cdot z_1 + x_6 \cdot z_6 + x_7 \cdot z_4 \\
\quad + x_7 \cdot z_7 + x_7 \cdot z_3 + x_7 \cdot z_6 + x_7 \cdot z_2 + x_7 \cdot z_0 + x_2 + x_3 + x_6 + x_1 + x_5 + z_2 + z_0 \\
\quad + z_6 \\
0 = x_0 \cdot z_7 + x_0 \cdot z_4 + x_0 \cdot z_1 + x_1 \cdot z_6 + x_1 \cdot z_7 + x_1 \cdot z_5 + x_1 \cdot z_4 + x_1 \cdot z_1 + x_2 \cdot z_6 \\
\quad + x_2 \cdot z_3 + x_2 \cdot z_0 + x_3 \cdot z_5 + x_3 \cdot z_3 + x_3 \cdot z_1 + x_3 \cdot z_0 + x_4 \cdot z_7 + x_4 \cdot z_4 + x_4 \cdot z_2 \\
\quad + x_4 \cdot z_6 + x_4 \cdot z_1 + x_4 \cdot z_5 + x_5 \cdot z_6 + x_5 \cdot z_7 + x_5 \cdot z_0 + x_5 \cdot z_2 + x_5 \cdot z_1 + x_6 \cdot z_5 \\
\quad + x_6 \cdot z_3 + x_6 \cdot z_0 + x_6 \cdot z_4 + x_6 \cdot z_6 + x_6 \cdot z_1 + x_7 \cdot z_4 + x_7 \cdot z_7 + x_7 \cdot z_5 + x_7 \cdot z_0 \\
\quad + x_3 + x_2 + x_7 + x_1 + z_7 + z_6 + z_5 + z_4 + z_3 + z_1 + z_0 \\
0 = x_0 \cdot z_6 + x_0 \cdot z_3 + x_0 \cdot z_0 + x_1 \cdot z_1 + x_1 \cdot z_5 + x_1 \cdot z_3 + x_1 \cdot z_0 + x_2 \cdot z_4 + x_2 \cdot z_7 \\
\quad + x_2 \cdot z_6 + x_2 \cdot z_2 + x_2 \cdot z_1 + x_2 \cdot z_5 + x_3 \cdot z_1 + x_3 \cdot z_6 + x_3 \cdot z_0 + x_3 \cdot z_2 + x_3 \cdot z_7 \\
\quad + x_4 \cdot z_3 + x_4 \cdot z_3 + x_4 \cdot z_6 + x_4 \cdot z_1 + x_4 \cdot z_5 + x_4 \cdot z_0 + x_5 \cdot z_4 + x_5 \cdot z_5 + x_5 \cdot z_0 \\
\quad + x_5 \cdot z_7 + x_6 \cdot z_5 + x_6 \cdot z_3 + x_6 \cdot z_0 + x_6 \cdot z_2 + x_6 \cdot z_4 + x_6 \cdot z_7 + x_7 \cdot z_7 + x_7 \cdot z_3 \\
\quad + x_7 \cdot x_1 + x_2 + x_3 + x_7 + x_1 + z_7 + z_6 + z_5 + z_4 + z_3 + z_2 + z_1 + 1 \\
0 = x_0 \cdot z_5 + x_0 \cdot z_2 + x_0 \cdot z_7 + x_1 \cdot z_1 + x_1 \cdot z_5 + x_1 \cdot z_2 + x_1 \cdot z_7 + x_1 \cdot z_6 + x_2 \cdot z_4 \\
\quad + x_2 \cdot z_6 + x_2 \cdot z_1 + x_3 \cdot z_5 + x_3 \cdot z_0 + x_4 \cdot z_3 + x_4 \cdot z_5 + x_4 \cdot z_0 + x_5 \cdot z_4 + x_5 \cdot z_7 \\
\quad + x_6 \cdot z_2 + x_6 \cdot z_4 + x_6 \cdot z_7 + x_7 \cdot z_4 + x_7 \cdot z_3 + x_7 \cdot z_1 + x_0 + x_7 + x_1 + z_7 \\
\quad + z_6 + z_2 + z_1 + z_0 + 1
\end{cases}
$$

Rysunek 3.4: Reprezentacja przykładowego układu równań nad ciałem binarnym.

3.5 Ataki typu 0R oraz nR

Charakterystyki różnicowe (liniowe) nie koniecznie muszą być wyznaczane dla pełnej wersji szyfru (wszystkich rund), aby mogły być wykorzystane do skutecznego ataku. Ataki posługujące się charakterystykami krótszymi niż n rund określa się atakami typu nR [14]. Niech R będzie liczbą atakowanych rund, wtedy:

Atak *0R* — typ ataku różnicowego (liniowego), w którym wykorzystywana jest charakterystyka różnicowa (liniowa) $(\Delta_0, \Delta_1, \cdots, \Delta_R)$ opisująca przejście przez wszystkie rundy atakowanego szyfru. W ataku różnicowym zakładamy, że różnica szyfrogramów jest równa Δ_R z prawdopodobieństwem α (przy założeniu, że różnica tekstów jawnych wynosiła Δ_0). Dzięki takiej konstrukcji ataku łatwo identyfikować dobre pary tekstów jawnych, ponieważ poprawną parę wskazuje jednoznacznie różnica szyfrogramów. Taka konstrukcja ataku sprawia, że wymagania pamięciowe stają się praktycznie nieistotne.

Ataki *nR* — typ ataku różnicowego (liniowego), w którym wykorzystuje się charakterystyki różnicowe (liniowe) $(\Delta_0, \Delta_1, \cdots, \Delta_{R-n})$ do ataku na R rund szyfru. W praktyce najczęściej wykonuje się ataki $1R$, $2R$ lub $3R$. Metody te pozwalają na wykorzystanie charakterystyk różnicowych o wyższym prawdopodobieństwie niż ataki $0R$ jednak niesie to za sobą większą złożoność pamięciową ataku oraz gorszą możliwość identyfikacji poprawnych par, które mogą posłużyć do odzyskania części bitów kluczy rundowych. Faza filtracji kluczy jest tutaj bardziej kosztowna niż w atakach różnicowych typu OR (im bardziej wzrasta wartość n, tym szybciej wzrasta złożoność fazy filtracji kluczy). W tego typu atakach można wykorzystać również różniczkę $(\Delta_{in}, \Delta_{out})$ opisującą własności różnicowe $R-n$ rund o ile prawdopodobieństwo uzyskania różnicy Δ_{out} jest większe niż prawdopodobieństwo uzyskania tej różnicy przy losowo dobieranych różnicach początkowych. Odzyskamy wtedy maksymalnie n ostatnich kluczy rundowych.

Rozdział 4

Charakterystyka problemów CSP wybranych do modelowania procesów związanych z automatyzacją metod poszukiwania charakterystyk różnicowych

4.1 CSP

Problemem CSP (ang. *Constraint Satisfaction Problem*) można określić problem poszukiwania takiego wartościowania zmiennych należących do zbioru zmiennych decyzyjnych, które spełnia określony zbiór ograniczeń.

Klasyczna definicja problemu CSP wygląda następująco. Niech problem $\mathcal{P}$ będzie trójką $\mathcal{P} = \{\mathcal{X}, \mathcal{D}, \mathcal{C}\}$, gdzie:

- $\mathcal{X}$ – zbiór n zmiennych $\{X_1, X_2, \cdots, X_n\}$;
- $\mathcal{D}$ – zbiór n dziedzin $\{D_1, D_2, \cdots, D_n\}$ odpowiadającym zmiennym X_i takim, że $X_i \in D_i$;
- $\mathcal{C}$ – skończony zbiór ograniczeń $\{C_1(S_1), C_2(S_2), \cdots, C_t(S_t)\}$, gdzie każde S_i to zbiór zmiennych $\in \mathcal{X}$. Ograniczenie C_i jest kombinacją poprawnych wartości zmiennych ze zbioru S_i.

Rozwiązaniem problemu są zbiory zmiennych $S_1, \cdots, S_t$, tak aby spełnione były wszystkie ograniczenia C_i.

Metody skupiające się na rozwiązywaniu problemów z rodziny CSP są przedmiotem intensywnych badań zarówno w dziedzinie sztucznej inteligencji, jak i badań operacyjnych. Regularność ich formułowania stwarza wspólną podstawę do analizy i rozwiązywania

problemów z wielu pozornie niezwiązanych ze sobą obszarów [32]. Niestety algorytmy rozwiązujące CSP często wykazują dużą złożoność (wymagają połączenia heurystyki i kombinatorycznych metod wyszukiwania), aby można je było rozwiązać w akceptowalnym czasie. Dziedzina badań, która szczególnie koncentruje się na rozwiązywaniu tego rodzaju problemów, nazywana jest CP (ang. *Constraint Programming*).

Obszary nauki związane z rozwiązywaniem problemów takich jak SAT (ang. *Boolean Satisfiability Problem*), SMT (ang. *Satisfiability Modulo Theories*), MIP (ang. *Mixed Integer Programming*) czy też ASP (ang. *Answer Set Programming*) to pola badań skupiające się na rozwiązaniu poszczególnych postaci problemu CSP.

4.2 SAT

Problem SAT (ang. *Boolean Satisfiability Problem*) to problem CSP, w którym zmienne przyjmują wartości binarne. Rozwiązanie problemu polega na odnalezieniu takiego wartościowania zmiennych, które spełni wszystkie ustalone ograniczenia (w przypadku tego problemu możemy utożsamić to ze wszystkimi równaniami określającymi problem SAT).

Problem SAT jest pierwszym problemem, dla którego udowodniono, że należy do klasy problemów NP-zupełnych. W roku 1971 dokonał tego Stephen Cook [23]. Obecnie nie jest znany algorytm, który skutecznie rozwiązałby każdy problem SAT i uważa się, że taki algorytm nie istnieje. Jednak dowód tej hipotezy nie został jak dotąd przeprowadzony. Istnieje wiele uznanych algorytmów, które używają heurystycznych metod rozwiązywania problemu SAT. Można je pogrupować według wykorzystywanych w nich technik, np. DPLL (Davis-Putnam-Logemann-Loveland [27]) czy też CDCL (Conflict Driven Clause Learning [49]).

Współcześnie do rozwiązywania problemów SAT najczęściej wykorzystuje się algorytm CDCL, który inspiruje się algorytmem DPLL i jednocześnie wykorzystuje szereg technik m.in. [8]:

- niechronologiczny sposób działania algorytmu wyszukiwania wstecz (ang. *backtrack search*),
- uczenie się nowych konfliktów podczas wyszukiwania wstecz,
- wykorzystanie struktury konfliktów podczas uczenia się klauzul,
- wykorzystanie *leniwych* struktur danych do reprezentacji formuł,
- wykorzystanie heurystyk rozgałęzionych, które charakteryzują się niskimi kosztami obliczeniowymi i odbierającymi informacje zwrotne z wyszukiwania wstecz,
- okresowe ponowne uruchamianie wyszukiwania wstecz.

W potocznym języku i badaniach operacyjnych często wykorzystuje się określenie *SAT solver* w stosunku do oprogramowania rozwiązującego problem SAT. W kryptologii *SAT solvery* wykorzystywane są m.in.:

- w zagadnieniach związanych z kryptoanalizą szyfrów blokowych i strumieniowych [25] [29] [2] oraz funkcji skrótu [31],
- w metodach związanych z weryfikacją formalną, automatycznym generowaniem wzorców testowych i syntezą logiczna [36],
- do poszukiwania krótkich cykli w iterowanych algorytmach kryptograficznych [30] (takich jak szyfry strumieniowe) lub prymitywach (takich jak `LFSR`, `NLFSR`).

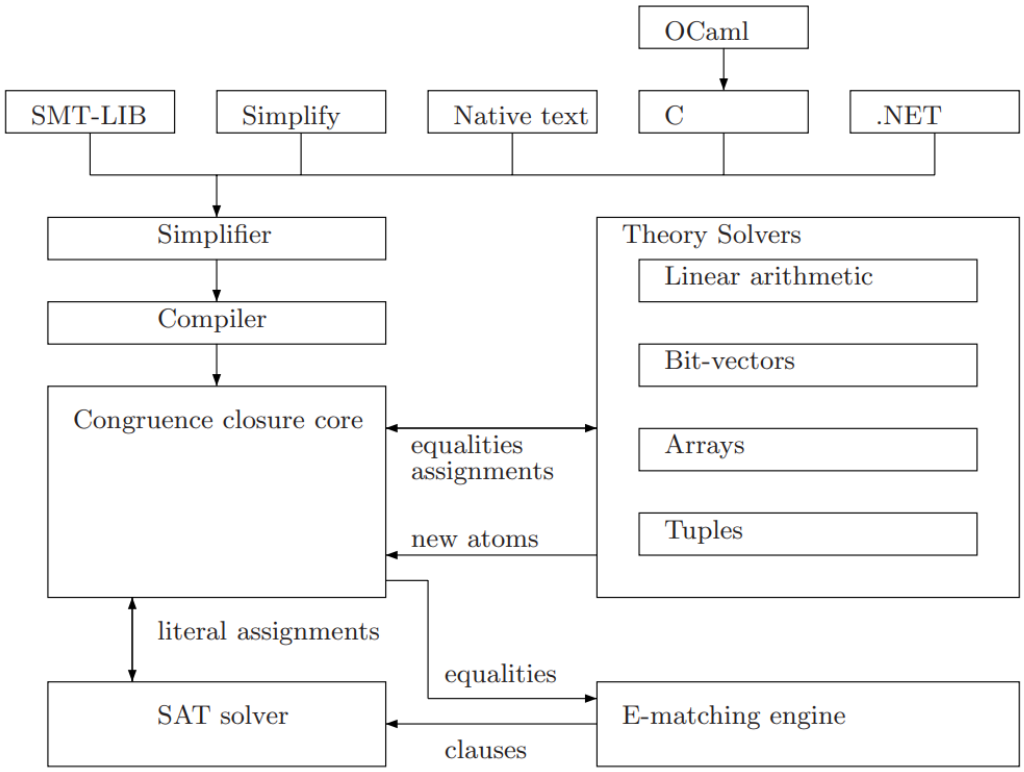
Wadą algorytmów wykorzystanych do rozwiązywania problemów `SAT` jest trudna do określenia złożoność obliczeniowa, którą dałoby się wyznaczyć na bazie własności modelu, który przedstawiany jest najczęściej jako zbiór klauzul w formacie `CNF`. Problem `SAT` w ogólności jest problemem NP-zupełnym dlatego czas działania algorytmu poszukującego rozwiązania z reguły będzie wykładniczy. Pewne obserwacje w tej kwestii w stosunku do losowo generowanych problemów `SAT` zostały opisane w [8]. Dotyczą one hipotezy, która mówi, że w przypadku losowo generowanych problemów `SAT` (ang. *Random Satisfiability*), opisanych przez zbiór klauzul w formacie `CNF` o ustalonej długości k, algorytm poszukujący rozwiązania w większości prób zadziałał w czasie wielomianowym, w przypadku gdy stosunek liczby klauzul do liczby zmiennych znajdował się w określonym przedziale. Stosunek ten określany jest niekiedy *gęstością* modelu `SAT` opisanego poprzez zbiór klauzul `CNF`. Tabela nr 4.1 przedstawia dolną i górną granicę przedziału wyznaczoną w [8] dla zadanych długości k.

Tablica 4.1: Ograniczenia dotyczące stosunku liczby klauzul w formacie `CNF` do liczby zmiennych dla niektórych małych wartości k, dla których algorytm rozwiązujący problem `SAT` zadziałał w czasie wielomianowym [8].

k	3	4	5	7	10	20
granica dolna	4.51	10.23	21.33	87.88	708.94	726.817
granica górna	3.52	7.91	18.79	84.82	704.98	726.809

4.3 SMT

Problem `SMT` (ang. *Satisfiability Modulo Theories*) to problem `CSP`, w którym zmienne przyjmują wartości z różnych dziedzin np. dziedziny liczb rzeczywistych, całkowitych czy też wartości binarne. Dodatkowo możliwe jest wykorzystanie różnych struktur danych np. listy, tablice czy też wektory bitowe. Wszystkie ograniczenia, jak i sam model problemu `SMT` wyrażony jest za pomocą klasycznej logiki pierwszego rzędu.



Rysunek 4.1: Schemat ideowy *SMT solvera* `Z3` [56].

Podobnie jak w przypadku problemów `SAT` oprogramowanie rozwiązujące problem `SMT` potocznie określa się *SMT solverem*. Na rysunku nr 4.1 przedstawiono schemat uwzględniający poszczególne komponenty ogólnodostępnego *SMT solvera* `Z3` opracowanego przez *Microsoft*. Przykładowa formuła, której rozwiązanie można określić za pomocą *SMT solvera* wygląda następująco:

$$(\sin(x^3) = \cos(\log(y) \cdot x) \vee b \vee -x^2 \geq 2.3y) \wedge (\neg b \vee y < -34.4 \exp(x) > \frac{y}{x})$$

gdzie $b \in \mathbb{B}$ (wartości binarne) oraz $x, y \in \mathbb{R}$ (liczby rzeczywiste).

Podstawowym celem badań związanych z problemem spełnialności teorii modulo jest opracowywanie algorytmów i narzędzie do weryfikacji mogących działać na wyższym poziomie abstrakcji niż logika binarna.

4.4 `MIP`

Problem `MIP` (ang. *Mixed Integer Programming*) to problem `CSP`, w którym zmienne decyzyjne mogą przyjmować wartości całkowite. Zasadniczo rozwiązanie tego problemu polega na maksymalizacji lub minimalizacji funkcji celu, poprzez dobranie odpowiedniego

wartościowania zamiennych decyzyjnych. Należy zachować przy tym wszystkie narzucone ograniczenia.

Technika ta znajduje szerokie zastosowanie w badaniach operacyjnych związanych z podjęciem optymalnych decyzji. Przykładowy matematyczny zapis takiego problemu wygląda następująco:

$$\min_{x,y} z \triangleq c^T x + d^T y$$

$$\text{przy ograniczeniach } Ax + Ey \left\{ \begin{array}{c} \leq \\ = \\ \geq \end{array} \right\} b$$

$$x_{min} \leq x \leq x_{max}$$
$$y \in \{0,1\}^{n_y}$$

gdzie n_y to rozmiar wektora y.

Jeżeli wszystkie ograniczenia wyrażone są poprzez funkcje liniowe problem ten określany jest jako `MILP` (ang. *Mixed Integer Linear Programming*). W kryptologii coraz częściej problem ten wykorzystuje się do modelowania zadań pozwalających wyznaczyć minimalną liczbę aktywnych skrzynek podstawieniowych w szyfrach opartych na sieci `SPN` [55].

Strona celowo pozostawiona pusta.

Rozdział 5

Metody zautomatyzowanego wyznaczania wybranych własności szyfrów blokowych

5.1 Differential Branch Number

Parametr *Differential Branch Number* [rozdz. 2.6.3] można wyznaczyć dowodząc matematycznych własności przekształcenia, poprzez brutalne przeszukanie lub poprzez rozwiązanie prostego zadania optymalizacji. Oczywiście najbardziej przejrzysty jest prosty matematyczny dowód, który niepodważalnie wyznaczy ten parametr. Niekiedy jego sformułowanie jest jednak kłopotliwe, a brutalne przeszukanie - zbyt czasochłonne. Warto posłużyć się wtedy modelem opisującym różnicowe własności prymitywu i na jego podstawie sformułować proste zadanie optymalizacji, którego rozwiązanie jednoznacznie wyznaczy minimalna wartość parametru *Differential Branch Number* dla badanego przekształcenia. W ramach badań związanych z niniejszą dysertacją, zaimplementowano proste narzędzie, które na bazie grafu `AIG` opisującego przekształcenia w automatyczny sposób wyznacza minimalną wartość $\mathbb{D}_{bn}$ dla zadanego przekształcenia liniowego. Tabela 5.1 przedstawia wartość parametru *Differential Branch Number* w wybranych operacjach wykorzystywanych we współczesnych szyfrach blokowych.

Graf `AIG` budowany był automatycznie na bazie implementacji badanej operacji w języku funkcyjnym `Cryptol` [34] i narzędzia `SAW` [21]. Uzyskane wyniki są zgodne z teorią a czas działania narzędzia jest akceptowalny nawet w przypadku operacji związanych z mnożeniem 32 bitowego strumienia danych przez macierze wskazane w algorytmie `PYJAMASK` [35].

Tablica 5.1: Czas wyznaczenia parametru *Differential Branch Number* dla wybranych operacji wykorzystywanych we współczesnych szyfrach blokowych.

Nazwa algorytmu	Nazwa operacji	Rozmiar wejścia [bity]	Rozmiar wyjścia [bity]	Rozmiar słowa [bity]	$\mathbb{D}_{bn}$	$\approx$ Czas [s]
AES	MixColumn	32	32	8	5	< 1
AES	MixColumn	32	32	1	6	< 1
MIDORI	MixColumn	16	16	4	4	< 1
KLEIN	MixNibbles	32	32	4	5	< 1
PRESENT	Sublayer	64	64	1	2	< 1
PYJAMASK	M_0	32	32	1	12	1230
PYJAMASK	M_1	32	32	1	12	1540
PYJAMASK	M_2	32	32	1	12	3770
PYJAMASK	M_3	32	32	1	12	1530
SATURNIN	mul	64	64	16	2	< 1
SATURNIN	mul	64	64	1	2	< 1

5.2 Minimalna liczba aktywnych skrzynek podstawieniowych w sieci *SPN*

5.2.1 Jednokryterialne zadanie optymalizacji bazujące na modelu MILP

Wyznaczenie minimalnej liczby aktywnych skrzynek podstawieniowych biorących udział w procesie szyfrowania jest ważnym elementem procesu analizy odporności szyfru na kryptoanalizę różnicową oraz jest bardzo pomocne podczas badań związanych z wyznaczeniem najlepszych charakterystyk różnicowych. Do określenia minimalnej liczby aktywnych skrzynek podstawioniowych posłużymy się metodami programowania całkowitoliczbowego oraz parametrem *Differential Branch Number*. Opisywana metoda została zaproponowana w [55].

Dla $n \in N_+$ rozważmy wyrażenie $\Delta = (\Delta_0, \Delta_1, \cdots, \Delta_{n-1})$ oraz odpowiadający mu wektor $x = (x_0, x_1, \cdots, x_{n-1})$ gdzie:

$$x_i = \begin{cases} 0 & \text{gdy } \Delta_i = 0 \\ 1 & \text{w p.p.} \end{cases}$$

Załóżmy, że badany szyfr składa się z dwóch warstw tzn. liniowej i nieliniowej. Za warstwę nieliniową odpowiadają skrzynki podstawieniowe, a za liniową funkcja:

$$L(x_0^{in}, x_1^{in}, \cdots, x_{n-1}^{in}) = (x_0^{out}, x_1^{out}, \cdots, x_{n-1}^{out})$$

Wyrażenie $\mathbb{D}_{bn} \geq 0$ określa wartość parametru *Differential Branch Number* (rozdz. 2.6.3) dla funkcji L. Aby poprawnie wyznaczyć szukaną wartość wprowadźmy również zmienną d

(zagwarantuje nam ona poprawność równań w późniejszym etapie w przypadku gdy $\forall_{i=0}^{n-1}$ $x_i^{in} = x_i^{out} = 0$), gdzie:

$$d = \begin{cases} 0 & \text{gdy } \forall_{i=0,\cdots,n-1}\ x_i^{in} = 0 \text{ oraz } x_i^{out} = 0 \\ 1 & \text{gdy w p.p.} \end{cases}$$

Na podstawie definicji parametru *Differential Branch Number* (rozdz. 2.6.3), postaci operacji L oraz określonej wcześniej zmiennej d prawdziwe jest zatem stwierdzenie, że:

$$\begin{cases} x_0^{in} + x_1^{in} + \cdots + x_{n-1}^{in} + x_0^{out} + x_1^{out} + \cdots + x_{n-1}^{out} \geq \mathbb{D}_{bn} \cdot d \\ d \geq x_0^{in} \\ d \geq x_1^{in} \\ \cdots \\ d \geq x_{n-1}^{in} \\ d \geq x_0^{out} \\ d \geq x_1^{out} \\ \cdots \\ d \geq x_{n-1}^{out} \end{cases}$$

Aby wyznaczyć minimalną liczbę aktywnych skrzynek podstawieniowych należy opisać w ten sposób, każdą z badanych rund oraz wyznaczyć funkcję celu. Ważne jest, aby zmienna d wprowadzana była do układu w sposób niezależny dla każdej z rund algorytmu (a dokładniej mówiąc dla każdego wywołania operacji liniowej L). Zmienne wyjściowe funkcji L w rundzie r są zmiennymi wejściowymi funkcji L w rundzie $r + 1$. W badaniach przyjmuje się, że warstwa skrzynek nie zmienia aktywności zmiennych x_i. Funkcja celu dla tak przedstawionego zadania będzie wyglądała następująco:

$min(\sum_i^m x_i)$, gdzie m to liczba wszystkich zmiennych x.

Aby uniknąć trywialnego rozwiązania, gdzie wszystkie zmienne x_i równe są 0 (żadna skrzynka nie jest aktywna) należy dodać dodatkowe ograniczenie:

$$\sum_i^m x_i > 0$$

Utworzony zbiór równań odpowiada zbiorowi ograniczeń zadania `MILP` (rozdz. 4.4).

Przykład. Wyznaczenie minimalnej liczby aktywnych skrzynek podstawieniowych dla szyfru blokowego `AES` za pomoca zadania `MILP`

Przyjmijmy, że wektor $(x_0, x_1, \cdots, x_{15})$ odpowiada wektorowi bajtów, który jest wejściem algorytmu. Na rysunku nr 5.1 zaprezentowano jak będzie zmieniać się numeracja zmiennych podczas pierwszej rundy algorytmu. Przyjęto, że operacja nieliniowa *SB* (ang. *Subbytes*) nie zmienia wartości zmiennych x_i (jeżeli skrzynka i była aktywna dla $x_i \neq 0$

w operacji *SB*, po niej wartość x_i jest $\neq 0$). Operacje *SR* (ang. *ShiftRows*) oraz *MC* (ang. *MixColumns*) to przekształcenia liniowe, w którym operacja *SR* odpowiada jedynie za przesunięcie zmiennych. Nie jesteśmy w stanie podać wartości zmiennych x_i po operacji *MC* dlatego w tym momencie następuje przenumerowanie zmiennych stanu na $(x_{16}, \cdots, x_{31})$ (zmienne te stanowią wejście do następnej rundy).

$$\begin{bmatrix} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{bmatrix} \xrightarrow{SB} \begin{bmatrix} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{bmatrix} \xrightarrow{SR} \begin{bmatrix} x_0 & x_4 & x_8 & x_{12} \\ x_5 & x_9 & x_{13} & x_1 \\ x_{10} & x_{14} & x_2 & x_6 \\ x_{15} & x_3 & x_7 & x_{11} \end{bmatrix} \xrightarrow{MC} \begin{bmatrix} x_{16} & x_{20} & x_{24} & x_{28} \\ x_{17} & x_{21} & x_{25} & x_{29} \\ x_{18} & x_{22} & x_{26} & x_{30} \\ x_{19} & x_{23} & x_{27} & x_{31} \end{bmatrix}$$

Rysunek 5.1: Numeracja zmiennych w pierwszej rundzie algorytmu *AES*

Korzystając z informacji, że $\mathbb{D}_{bn}$ dla operacji *MC* równe jest 5 (jeżeli przynajmniej jeden bajt jest aktywny na wejściu to minimum cztery bajty są aktywne na wyjściu, a w każdym innym przypadku nie otrzymamy wartości mniejszej niż 5) możemy zatem zapisać równania liniowe dla pierwszej rundy:

$$\begin{cases} x_0 + x_5 + x_{10} + x_{15} + x_{16} + x_{17} + x_{18} + x_{19} - 5 \cdot d_0 \geq 0 \\ d_0 - x_0 \geq 0 \\ d_0 - x_5 \geq 0 \\ d_0 - x_{10} \geq 0 \\ d_0 - x_{15} \geq 0 \\ d_0 - x_{16} \geq 0 \\ d_0 - x_{17} \geq 0 \\ d_0 - x_{18} \geq 0 \\ d_0 - x_{19} \geq 0 \end{cases}$$

oraz kolejno dla następnych 3 rund algorytmu:

$$
\begin{cases}
x_4 + x_9 + x_{14} + x_3 + x_{20} + x_{21} + x_{22} + x_{23} - 5 \cdot d_1 \geq 0 \\
d_1 - x_4 \geq 0 \\
d_1 - x_9 \geq 0 \\
d_1 - x_{14} \geq 0 \\
d_1 - x_3 \geq 0 \\
d_1 - x_{20} \geq 0 \\
d_1 - x_{21} \geq 0 \\
d_1 - x_{22} \geq 0 \\
d_1 - x_{23} \geq 0 \\
x_8 + x_{13} + x_2 + x_7 + x_{24} + x_{25} + x_{26} + x_{27} - 5 \cdot d_2 \geq 0 \\
d_2 - x_8 \geq 0 \\
d_2 - x_{13} \geq 0 \\
d_2 - x_2 \geq 0 \\
d_2 - x_7 \geq 0 \\
d_2 - x_{16} \geq 0 \\
d_2 - x_{17} \geq 0 \\
d_2 - x_{18} \geq 0 \\
d_2 - x_{19} \geq 0 \\
x_{12} + x_1 + x_6 + x_{11} + x_{28} + x_{29} + x_{30} + x_{31} - 5 \cdot d_3 \geq 0 \\
d_3 - x_{12} \geq 0 \\
d_3 - x_1 \geq 0 \\
d_3 - x_6 \geq 0 \\
d_3 - x_{11} \geq 0 \\
d_3 - x_{28} \geq 0 \\
d_3 - x_{29} \geq 0 \\
d_3 - x_{30} \geq 0 \\
d_3 - x_{30} \geq 0
\end{cases}
$$

Równania dla następnych rund mają taką samą postać, a jedyna zmiana polega na numeracji zmiennych. Zmienne wyjściowe rundy r są zmiennymi wejściowymi rundy $r+1$. Dla 5 rund algorytmu *AES* będziemy mieć zatem 80 zmiennych x oraz 20 zmiennych d.

Po rozwiązaniu zadania *MILP* modelowanego osobno dla każdej liczby rund otrzymamy minimalną liczbę aktywnych skrzynek podstawieniowych po każdej z nich. Uzyskane wyniki zaprezentowano na rysunku nr 5.2.

Tablica 5.2: Minimalna liczba aktywnych skrzynek podstawieniowych w algorytmie *AES*

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\min(k_N)$	1	5	9	25	26	30	34	50	51	55	59	75	76	80

5.2.2 Metoda poszukiwania minimalnej liczby aktywnych skrzynek bazująca na grafie AIG i modelu SAT

Metoda opisana w rozdziale 5.2.1 świetnie nadaje się do szyfrów orientowanych na przetwarzanie kilkubitowych słów (np. bajtów lub słów heksadecymalnych). Niezaprzeczalnie jej zaletą jest różnież duża wydajność. Niestety nie jest to podejście ogólne, które bez względu na konstrukcję algorytmu będzie skuteczne i równie wydajne co w przypadku algorytmu `AES`.

Jedną z wad podejścia omówionego w rodziale 5.2.1 jest brak dowodu w postaci podania przykładowej charakterystyki różnicowej, ktora aktywuje minimalną liczbę skrzynek. Może to prowadzić do tego, że algorytm uznaje wszystkie przejścia zgodne z parametrem *Differential Branch Number* za pewne i nie rozpatruje sytuacji, kiedy niemożliwe jest podanie na warstwę liniową różniczki, która zachowa minimalną liczbę aktywnych skrzynek. W przypadku szyfru `AES` nie rzutuje to na poprawne rozwiązanie, jednak istnieją algorytmy, w których zabieg ten spowoduje *zaniżenie poprawnego rozwiązania* (np. szyfr blokowy `MIDORI`, gdzie wykorzystywany jest prymityw `aMDS` (ang. *almost Maximum Distance Separable* Matrix)). Oczywiście wpływa to na niekorzyść atakującego, który będzie starał się odnaleźć charakterystykę różnicową, która w rzeczywistości nie istnieje.

Z tego względu w ramach badań związanych niniejszą z rozprawą doktorską opracowano metodę, ktorą da się zastosować do każdej konstrukcji szyfru, a co więcej określenie minimalnej liczby skrzynek równoznaczne jest z podaniem konkretnej charakterystyki różnicowej, która aktywuje wyznaczoną liczbę skrzynek. Zakłada ona, że R rundowa charakterystyka różnicowa $(\Delta_0, \Delta_1, \Delta_2, \cdots, \Delta_R)$ to strumień $n \cdot R$ bitów od x_0 do $x_{n \cdot R-1}$, gdzie Δ_i to strumień bitów $(x_{n \cdot i}, x_{n \cdot i+1}, \cdots, x_{n \cdot i+n-1})$, a n to liczba bitów różnicy Δ_i. Strumień bitów $(x_0, x_1, \cdots, x_{n \cdot R-1})$ stanowi zbiór niewiadomych. Sposób propagacji skrzynek podstawieniowych w algorytmie blokowym określony jest poprzez funkcję $\mathbb{SE}$, a sposób propagacji skrzynek podstawieniowych w rundzie poprzez funkcję $\mathbb{SR}$. Rozwiązanie zadania polega na wyznaczeniu wartościowania modelu `SAT` przy zadanej liczbie całkowitej $x \neq 0$ lub przedziale liczbowym $< x_{low}, x_{up} >$, w którym powinna znajdować się minimalna liczba aktywnych skrzynek. Potwierdzenie, że dla zadanej liczby całkowitej $x \neq 0$ lub przedziału $< x_{low}, x_{up} >$ model nie jest spełnialny (taki model zwyczajowo określa się modelem `UNSAT`) jest jednoznaczne z dowodem, że nie istnieje charakterystyka różnicowa spełniająca zadane

kryterium (rozumiane jako liczba skrzynek aktywowana przez charakterystykę różnicową).

$$\mathbb{SR}(\Delta_i, \Delta_j) = \begin{cases} a, & \text{jeżeli } p > 0 \text{ gdzie } p\text{: } \Delta_i \xrightarrow{p} \Delta_j, \\ & a \in \mathbf{N}_+ \text{ liczba aktywnych skrzynek w różniczce } \Delta_i \\ -1, & \text{jeżeli } p = 0 \text{ gdzie } p\text{: } \Delta_i \xrightarrow{p} \Delta_j \end{cases} \tag{5.1}$$

$$\mathbb{SE}(\Delta_0, \Delta_1, \Delta_2, \cdots, \Delta_R) = \begin{cases} \sum_{i=0}^{i=R} \mathbb{SR}(\Delta_i, \Delta_{i+1}) & \text{jeżeli } \nexists\ \mathbb{SR}(\Delta_i, \Delta_{i+1}) = -1 \\ -1 & \text{jeżeli } \exists\ \mathbb{SR}(\Delta_i, \Delta_{i+1}) = -1. \end{cases} \tag{5.2}$$

Opracowana metoda składa się z trzech zasadniczych etapów:

1. Opisania propagacji aktywności skrzynek podstawieniowych w zadanym algorytmie za pomocą grafu `AIG`. W praktyce będzie to zdefiniowanie funkcji $\mathbb{SR}$ oraz $\mathbb{SE}$. Następnie zaprezentowanie funkcji $\mathbb{SE}$ za pomocą grafu `AIG`.

2. Konwersji grafu `AIG` do modelu `SAT`. Przed konwersją zalecane jest wykorzystanie metod pozwalających na zredukowanie rozmiaru grafu `AIG` takich jak *redukcja funkcjonalna* wierzchołków grafu czy też innych metod minimalizacji funkcji boolowskiej.

3. Wyznaczenia wartościowania modelu `SAT` przy zadanej liczbie x:

   $$\mathbb{SE}(\Delta_0, \Delta_1, \Delta_2, \cdots, \Delta_R) = x$$

   lub przedziale liczbowym $< x_{low}, x_{up} >$, w którym powinna znajdować się minimalna liczba aktywnych skrzynek:

   $$x_{low} \leq \mathbb{SE}(\Delta_0, \Delta_1, \Delta_2, \cdots, \Delta_R) \leq x_{up}.$$

Kod źródłowy 5.1: Skrypt języka `Cryptol` opisujacy propagację aktywnych skrzynek w algorytmie `Present`.

```
1  module PRESENT_64_CLASSIC_MINBOXES_SEARCH where
2
3  type BLOCK_SIZE = 64
4
5  pbox = [
6      0x00, 0x04, 0x08, 0x0c, 0x10, 0x14, 0x18, 0x1c,
7      0x20, 0x24, 0x28, 0x2c, 0x30, 0x34, 0x38, 0x3c,
8      0x01, 0x05, 0x09, 0x0d, 0x11, 0x15, 0x19, 0x1d,
9      0x21, 0x25, 0x29, 0x2d, 0x31, 0x35, 0x39, 0x3d,
10     0x02, 0x06, 0x0a, 0x0e, 0x12, 0x16, 0x1a, 0x1e,
11     0x22, 0x26, 0x2a, 0x2e, 0x32, 0x36, 0x3a, 0x3e,
12     0x03, 0x07, 0x0b, 0x0f, 0x13, 0x17, 0x1b, 0x1f,
13     0x23, 0x27, 0x2b, 0x2f, 0x33, 0x37, 0x3b, 0x3f
```

```
]

sublayer : [BLOCK_SIZE] -> [BLOCK_SIZE]
sublayer input = [ input!(pbox ! i) | i <- ([0,1..63]:[_][8]) ]

type DDT_TYPE = 16

DDT_is_possible : [16][16]Bit
DDT_is_possible = [
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0],
    [0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0],
    [0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0],
    [0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0],
    [0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1],
    [0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1],
    [0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    [0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0],
    [0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0],
    [0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0],
    [0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0],
    [0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0],
    [0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0],
    [0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1]
]

MAP_ACTIVATE_BOXES : ([4], [4]) -> Bit
MAP_ACTIVATE_BOXES (x, y) = (DDT_is_possible@x)@y

MAP_ACTIVATE_BOXES_BLOCK : ([BLOCK_SIZE],[BLOCK_SIZE]) -> [2][DDT_TYPE]
MAP_ACTIVATE_BOXES_BLOCK (in, out) = [ number_of_activate_boxes ! 0,
    differential_is_possible ]
    where x = groupBy`{4}(in)
        y = groupBy`{4}(out)
        differential_is_possible = [ MAP_ACTIVATE_BOXES(x@i, y@i) | i <-
            [0..15]:[_][4]]
        number_of_activate_boxes = [ 0:[DDT_TYPE] ] # [ if(x@i != 0) then a
            + 1:[DDT_TYPE] else a | i <- [0..15]:[_][4] | a <-
            number_of_activate_boxes ]

MAP_SOLUTION : [32][BLOCK_SIZE] -> [32][BLOCK_SIZE]
MAP_SOLUTION X = mapping
    where mapping = [ X@0 ] # [ sublayer(X@i) | i <- [1..31]:[_][5] ]

CALCULATE_MINBOXES : [32][BLOCK_SIZE] -> [2][DDT_TYPE]
CALCULATE_MINBOXES X = probability_e ! 0
    where box_input     = [ X@0 ] # [ sublayer(X@i) | i <- [1..30]:[_][5] ]
            box_output    = [ X@i | i <- [1..31]:[_][5] ]
            probability_r = [ MAP_ACTIVATE_BOXES_BLOCK(box_input@i,
                box_output@i) | i <- [0..30]:[_][5] ]
```

```
60          probability_e = [ [0:[DDT_TYPE], -1:[DDT_TYPE]] ] # [ [(p@0 +
                e@0), (p@1 && e@1)] | e <-probability_r | p <-probability_e ]
```

W zaprezentowanym kodzie źródłowym [5.1] funkcja `MAP_ACTIVATE_BOXES_BLOCK` odpowiada funkcji $\mathbb{SR}$, a funkcji $\mathbb{SE}$ odpowiada funkcja `CALCULATE_MINBOXES`. Za pomocą narzędzia `SAW` [21], funkcja `CALCULATE_MINBOXES` (wyznaczająca liczbę aktywnych skrzynek w zadanej charakterystyce różnicowej) przedstawiana jest za pomocą grafu `AIG`. Tak przygotowany graf jest poddawany redukcji funkcjonalnej za pomocą narzędzia `ABC` [18] oraz innym zabiegom pozwalającym maksymalnie zredukować jego rozmiar. Następnie wykonywana jest konwersja zredukowanego grafu `AIG` do postaci `CNF` (ang. *Conjunctive Normal Form*) tj. zbioru klauzul określających model `SAT`.

Tablica 5.3: Optymalna charakterystyka różnicowa (pod kątem minimalnej liczby aktywnych skrzynek podstawieniowych) dla algorytmu `AES` (blok 128 bitów) uzyskana w wyniku rozwiązania zadania bazującego na grafie `AIG` i modelu `SAT`.

R	$\boldsymbol{\Delta_i}$	**Optymalna pod kątem minimalnej liczby aktywnych skrzynek podstawieniowych charakterystyka różnicowa**	$\approx \mathbf{P}(\boldsymbol{\Delta_{in}} \rightarrow \boldsymbol{\Delta_{out}})$	$\approx$ **Czas [h]**
14	Δ_0	00000000020000000002000000000000	2^{560}	87
	Δ_1	0000000000AD80AD0000000000000000		
	Δ_2	121C0E0E00000000004040C089DBC219D		
	Δ_3	00A50023B1007A00009600067B008400		
	Δ_4	00000000000072000000000023000000		
	Δ_5	0000000000000000000000060E900A9		
	Δ_6	16163A2C00000000D967BEBEBFD2D26D		
	Δ_7	0023002A0D000E0000D900D13100B200		
	Δ_8	000000000000D2000000000048000000		
	Δ_9	00000000000000000000000000070501		
	Δ_{10}	858594111C24381C6F4A252500000000		
	Δ_{11}	000082C3DF000004527B000000F01200		
	Δ_{12}	00000000BA00000000BB000000000000		
	Δ_{13}	00000000E300D8E30000000000000000		
	Δ_{14}	000000008000511B0000000000000000		

Wydajność opracowanej metody jest w pełni akceptowalna na potrzeby obliczeń. Oczywiście czas potrzebny na wyznaczanie poprawnego wartościowania modelu jest znacznie większy niż w przypadku metody wykorzystującej model `MILP` [rozdz. 5.2.1]. Należy jednak pamiętać, że w przypadku opracowanej metody wynikiem końcowym jest również przykładowa charakterystyka różnicowa, która aktywuje wyznaczoną minimalną liczbę skrzynek podstawieniowych. Tabela 5.4 przedstawia wyniki badań przeprowadzonych na szyfrach blokowych `AES`, `PRESENT`, `MIDORI`, `KLEIN` oraz `PYJAMASK`. Podczas obliczeń zakładano, że nie będą wykorzystywane żadne założenia dotyczące granicy dolnej, od której rozpocznie

Tablica 5.4: Minimalna liczba aktywnych skrzynek podstawieniowych i czas w jakim wyznaczono optymalną charakterystykę różnicową (pod kątem minimalnej liczby skrzynek podstawieniowych) dla wybranych szyfrów blokowych.

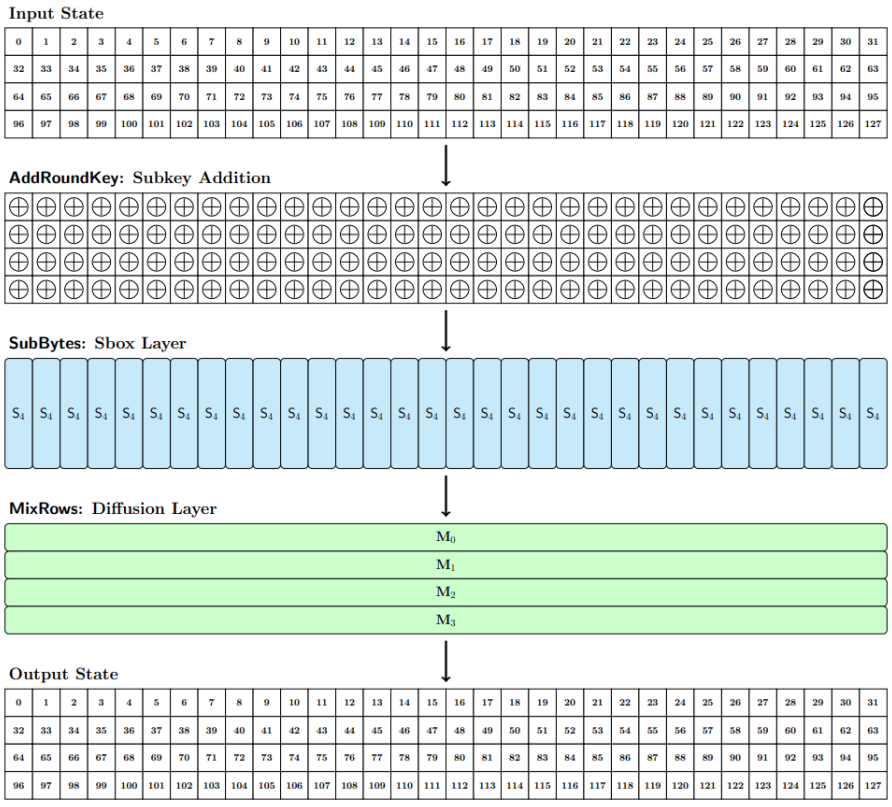
#R	AES (blok 128 bitów)		PRESENT (blok 64 bity)		MIDORI (blok 64 bity)		KLEIN (blok 64 bity)		PYJAMASK (blok 128 bitów)	
	#SBOX	≈ Czas	#SBOX	≈ Czas	#SBOX	≈ Czas	#SBOX	≈ Czas	#SBOX	≈ Czas
1	1	60 s	1	< 1 s	1	< 1 s	1	< 1 s	1	< 1 s
2	5	820 s	2	< 1 s	4	< 3 s	5	5 s	12	875 s
3	9	1240 s	4	4 s	7	8 s	8	15 s	19	120 h
4	25	2 h	6	11 s	16	36 s	15	60 s		
5	26	3 h	10	18 s	23	95 s	16	80 s		
6	30	4 h	12	30 s	30	220 s	20	130 s		
7	34	7 h	14	58 s	35	365 s	24	250 s		
8	50	14 h	16	80 s	38	460 s	30	470 s		
9	51	16 h	18	105 s	41	615 s	32	600 s		
10	55	19 h	20	140 s	50	1405 s	35	1000 s		
11	59	24 h	22	190 s	57	2940 s	39	1200 s		
12	75	68 h	24	251 s	62	4650 s	45	2200 s		
13	76	78 h	26	315 s	67	2 h				
14	80	87 h	28	405 s	72	3 h				
15			30	490 s	75	4 h				
16			32	580 s	84	6 h				
17			34	650 s						
18			36	770 s						
19			38	860 s						
20			40	980 s						
21			42	1190 s						
22			44	1340 s						
23			46	1575 s						
24			48	1670 s						
25			50	1770 s						
26			52	1820 s						
27			54	1980 s						
28			56	2100 s						
29			58	2250 s						
30			60	2380 s						
31			62	2637 s						
	charakterystyka różnicowa optymalna pod kątem minimalnej liczby aktywnych skrzynek podstawieniowych nie wskazana wcześniej w literaturze									

się poszukiwanie minimalnej liczby skrzynek podstawieniowych[1]. W przypadku pierwszych czterech wymienionych szyfrów udało się wyznaczyć minimalną liczbę aktywnych skrzynek oraz przykładową charakterystykę różnicową na pełne wersje algorytmów w relatywnie krótkim czasie [rys. 5.3]. Niestety w przypadku szyfru PYJAMASK wydajność metody jest

[1]Podczas pierwszej iteracji zakładano, że minimalna liczba aktywnych skrzynek podstawieniowych wynosi 1, niezależnie od aktualnie badanej rundy

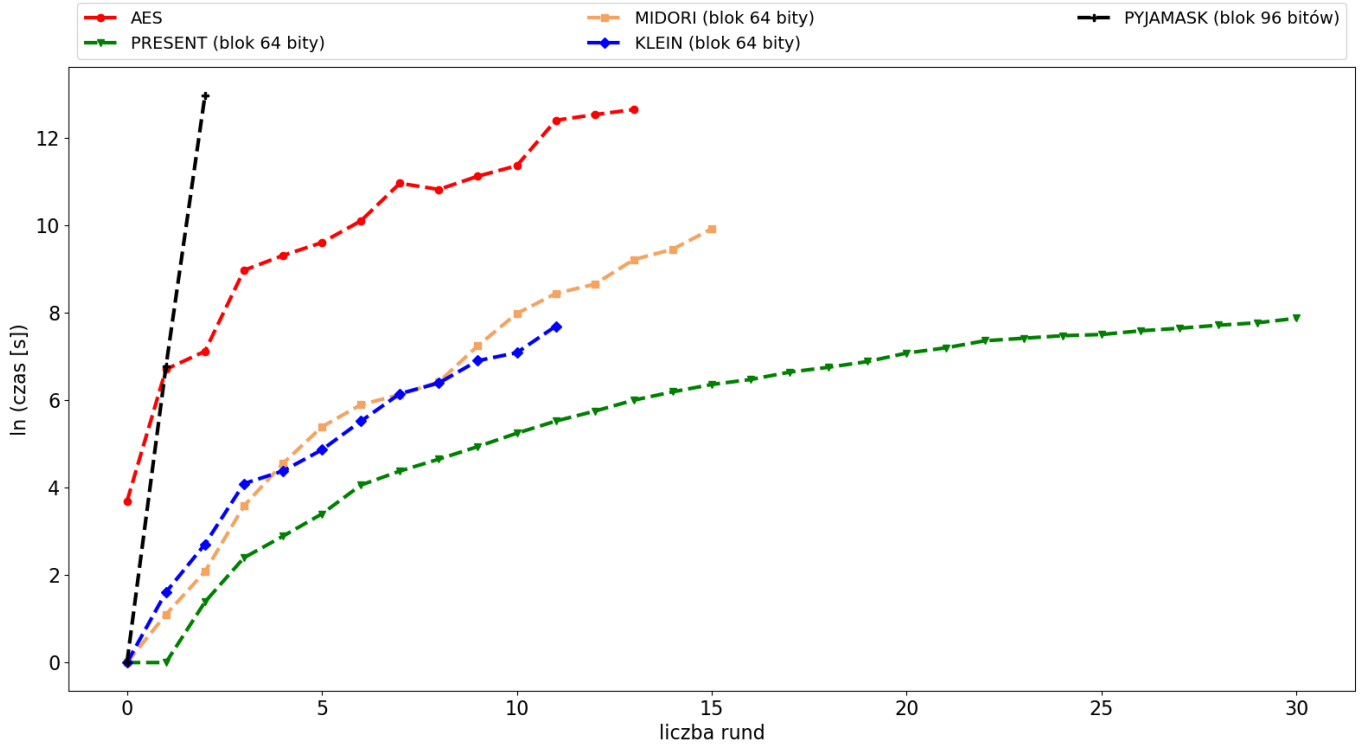
znacznie gorsza w porównaniu do innych rozpatrywanych szyfrów.

Przeprowadzane badania potwierdzają, że określenie minimalnej liczby skrzynek podstawieniowych jest zadaniem zdecydowanie mniej czasochłonnym w przypadku gdy algorytm blokowy jest zorientowany na przetwarzanie kilkubitowych słów[2]. Ważnym aspektem, który wpływa na złożoność obliczeniową jest również konstrukcja warstwy liniowej. W przypadku szyfru `AES` oraz `MIDORI` warstwa ta składa się z permutacji oraz mnożenia przez macierz `MDS` zorientowanych odpowiednio na 8 lub 4 bitowe słowa, a w przypadku szyfru `PRESENT` jedynie z permutacji bitów. Szyfr `PYJAMASK`, dla którego minimalną liczbę skrzynek udało się dokładnie wyznaczyć maksymalnie dla 4 rundy, zorientowany jest na przetwarzanie pojedynczych bitów. Warstwę liniową stanowi mnożenie poszczególnych części (wierszy) przetwarzanego bloku (trzech lub czterech odpowiednio dla długości bloku 96 oraz 128 bitów) przez przypisaną do niej macierz `MDS`. Co więcej 3 bitowa (lub odpowiednio 4 bitowa w wersji ze 128 blokiem danych) skrzynka podstawieniowa aktywowana jest poprzez bity z każdej wykorzystanej macierzy [rys. 5.2].

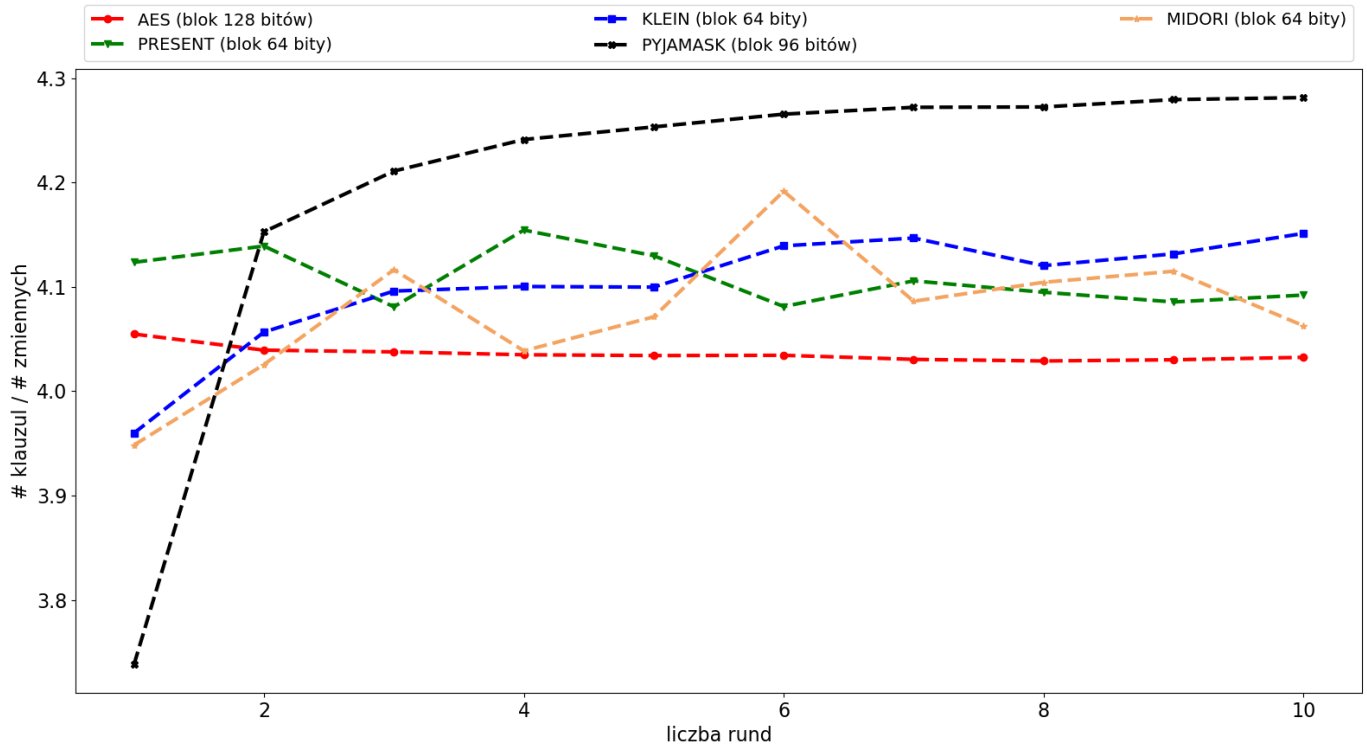


Rysunek 5.2: Runda szyfru blokowego `PYJAMASK` wykorzystującego 128 bitowy blok [35].

[2]W przypadku szyfru blokowego `AES` są to słowa 8 bitowe, a w przypadku `MIDORI` odpowiednio do długości bloku słowa 4 lub 8 bitowe



Rysunek 5.3: Wydajność narzędzia do poszukiwania charakterystyki różnicowej aktywującej minimalną liczbę skrzynek podstawieniowych w sieci `SPN` opartego na grafach `AIG` i modelu `SAT` i wykorzystaniu *SAT solver* `cadical` (skala logarytmiczna).



Rysunek 5.4: Wartość wskaźnika ρ dla modeli `SAT` generowanych poprzez grafy `AIG`, opisujących propagację charakterystyki różnicowej dla pierwszych 10 rund (minimalna liczba skrzynek).

Strona celowo pozostawiona pusta.

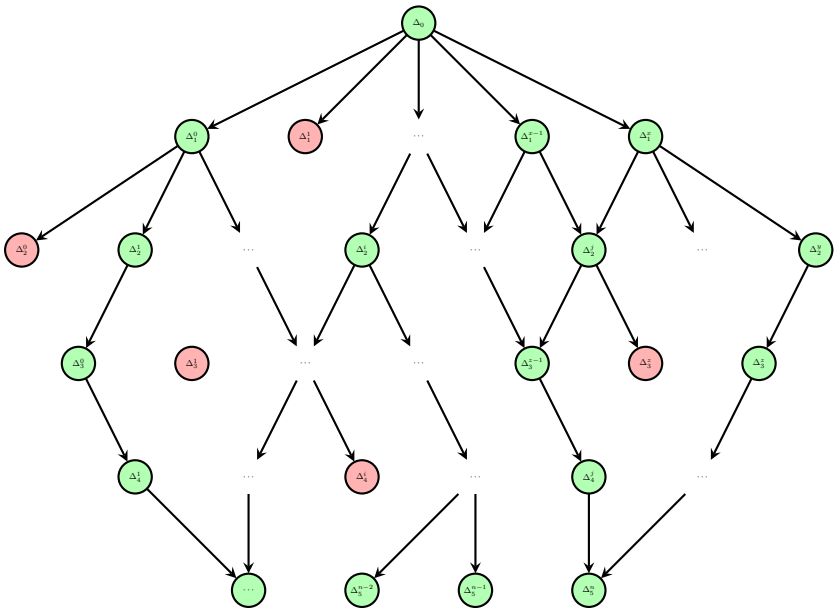
Rozdział 6

Metody zautomatyzowanego poszukiwania różniczek i charakterystyk różnicowych

6.1 Modyfikacja algorytmu *branch and bound*

Naiwny algorytm poszukiwania różniczek dla szyfrów blokowych może wykorzystywać metodę podziału i ograniczeń (ang. *branch and bound*), która polega na analizie drzewa przestrzeni stanów. Teoretycznie takie drzewo powinno rozpatrywać wszystkie możliwe charakterystyki różnicowe, jakimi może podążyć algorytm poszukujący rozwiązania. Algorytm zakłada przeszukiwanie drzewa wszerz, rozpoczynając od ustalonego korzenia drzewa. W rozpatrywanym problemie obliczeniowym, jakim jest poszukiwanie optymalnej różniczki, korzeniem drzewa jest ustalona niezerowa wejściowa charakterystyka różnicowa Δ_{in}. Oczywiście sprawdzenie wszystkich możliwych charakterystyk różnicowych początkowych nie jest możliwe ze względu ograniczeń czasowych (w przypadku bloku o długości n bitów będzie ich aż $2^n - 1$). Z tego względu zbiór ten jest inteligentnie ograniczony. Ograniczenie polega na wybraniu niewielkiego zbioru charakterystyk początkowych za pomocą rozwiązania zadania programowania całkowitoliczbowego `MILP` [rozdz. 5.2.1], które zdefiniowane jest tak, aby wskazać najmniejszą liczbę aktywnych skrzynek podstawieniowych. Tego typu zadanie określi jednak jedynie, które skrzynki powinny być aktywne. Niestety nie wskaże konkretnej charakterystyki różnicowej.

Po każdym wyznaczeniu następników charakterystyki różnicowej Δ_i wykonywana jest redukcja, która ma na celu odrzucenie *źle rokujących* charakterystyk. Warunek pozostawienia w drzewie perspektywicznej różniczki może być oparty na takich własnościach jak prawdopodobieństwo i liczba aktywnych skrzynek podstawieniowych. Ważnym założeniem algorytmu jest również zasada, która mówi o tym, że jeżeli w zbiorze perspektywicznych różniczek znajdą się dwie różniczki o tej samej postaci, są one scalane do jednej (prawdopodobieństwa charakterystyk różnicowych są sumowane, ponieważ zakładamy, że zdarzenia te będą niezależne). Idea algorytmu została zaprezentowana na rysunku nr 6.1. Przedstawione



Rysunek 6.1: Idea algorytmu poszukiwania różniczek dla szyfrów blokowych opartego na przeszukiwaniu drzewa binarnego.

założenia wykraczają poza zakres, w jakim stara się definiować bezpieczeństwo szyfru blokowego pod kątem kryptoanalizy różnicowej strategia szerokiej ścieżki [rozdz. 2.4], która nie rozważa łączenia niezależnych ścieżek różnicowych. Wyniki, które uzyskano po wykonaniu wstępnej wersji algorytmu zaimplementowanej w `C++` przedstawiono w tabeli 6.1 oraz 6.2. Oznaczenie P_{teo} dotyczy prawdopodobieństwa teoretycznego wyznaczonego przez algorytm. P_{emp} to prawdopodobieństwo wyznaczone empirycznie podczas symulacji ataku różnicowego.

Tablica 6.1: Różniczki dla algorytmu `KLEIN` (blok 64 bity) wyznaczone zmodyfikowanym algorytmem *podziału i ograniczeń*.

R	$\mathbf{\Delta_{in}}$	$\mathbf{\Delta_{out}}$	$\approx \mathbf{P_{teo}}(\mathbf{\Delta_{in}} \to \mathbf{\Delta_{out}})$	$\approx \mathbf{P_{emp}}(\mathbf{\Delta_{in}} \to \mathbf{\Delta_{out}})$	$\approx$ `Czas [s]`
3	0100000000000E01	0305060309000509	$2^{-17,00}$	$2^{16,50}$	< 1
4	30105050500000E0	A0C0606030104050	$2^{-30,95}$	$2^{29,70}$	10
5	0506000000000000	050F0A0506040202	$2^{-41,00}$	$2^{40,60}$	180
6	0000000000000020	F04070B030104050	$2^{-51,96}$	–	300
7	0000400040400000	405030106060A0C0	$2^{-60,50}$	–	550
8	0000000E030E0000	010B111014110A02	$2^{-72,95}$	–	800

Tablica 6.2: Różniczki dla algorytmu MIDORI (blok 64 bity) wyznaczone zmodyfikowanym algorytmem podziału i ograniczeń.

R	$\boldsymbol{\Delta_{\text{in}}}$	$\boldsymbol{\Delta_{\text{out}}}$	$\approx \mathbf{P_{teo}}(\boldsymbol{\Delta_{\text{in}}} \rightarrow \boldsymbol{\Delta_{\text{out}}})$	$\approx \mathbf{P_{emp}}(\boldsymbol{\Delta_{\text{in}}} \rightarrow \boldsymbol{\Delta_{\text{out}}})$	$\approx$ Czas [s]
3	00A00000A0000A00	0000AA0A0AAAAAA0	$2^{-11,00}$	$2^{10,50}$	< 1
4	0020020200200202	2022002220220022	$2^{-23,79}$	$2^{21,50}$	500
5	0200000222222000	0000990909999990	$2^{-35,05}$	$2^{34,20}$	2000
6	0000000220220222	2202020222202220	$2^{-46.94}$	–	2700
7	0002002000020020	CCCCCC0CCCC0CCC0	$2^{-62,00}$	–	3600
8	0000000000000220	0000000000000220	$2^{-69,00}$	–	6500

6.1.1 Wydajność zaimplementowanego narzędzia

Niestety wydajność algorytmu, w którym dostosowano metodę podziału i ograniczeń do poszukiwania różniczek nie jest satysfakcjonująca. Zakładając, że badany szyfr będzie algorytmem opartym na sieci SPN, największym wyzwaniem jest wyznaczenie wszystkich możliwych charakterystyk wyjściowych po rundzie na bazie zadanej charakterystyki Δ_i. Skrzynki podstawieniowe wykorzystywane w szyfrach blokowych mają w każdym z wierszy profilu różnicowego średnio połowę elementów niezerowych. Jeżeli przyjmiemy optymistyczne założenie, że przeglądane w ramach postępów poszukiwania różniczki będą aktywowały maksymalnie połowę ze wszystkich skrzynek (#mean sbox_{active}) średnia liczba operacji, którą należy wykonać, wyniesie:

$$(d-1)^{\#\text{mean sbox}_{active}} \text{ gdzie } d = 2^{sbox_{size}-1}.$$

W przypadku pesymistycznym należy zakładać, że aktywne będą wszystkie skrzynki w bloku (#max sbox_{active}) co daje nam liczbę operacji rzędu:

$$(d-1)^{\#\text{max sbox}_{active}} \text{ gdzie } d = 2^{sbox_{size}-1}.$$

W tabeli 6.3 zaprezentowano szacowaną średnią i maksymalną liczbę operacji jaka musi zostać wykonana, aby przejrzeć wszystkie możliwe charakterystyki różnicowe Δ_j, które możliwe są do uzyskana z danej charakterystyki Δ_i (wyznaczyć wszystkie następniki charakterystyki Δ_i po rundzie algorytmu blokowego). Niestety ze względu na dużą złożoność obliczeniową zaprezentowane podejście w praktyce nie nadaje się do analizy algorytmów z 8 bitową skrzynką podstawieniową np. szyfru AES.

Wadą algorytmu, która znacznie utrudnia jego automatyzację jest wyznaczanie perspektywicznych początkowych charakterystyk różnicowych Δ_{in}. Zadanie to należy wykonać przed przystąpieniem do właściwych obliczeń, a jego wykonanie wydaje się być dość specyficzne dla szyfrów różniących się w konstrukcji (sieć SPN, sieć Feistela, struktura Lai - Massey'a, ARX). Zbiór perspektywicznych początkowych charakterystyk różnicowych może okazać się również zbyt duży, aby wykonać przedstawiony algorytm na każdej z nich. W związku z redukcją *źle rokujących* następników nie mamy również pewności czy w niektórych przypadkach algorytm nie odrzuci ścieżki, która w etapie końcowym dawałaby

Tablica 6.3: Szacowana średnią i maksymalną liczbę operacji jaką należy wykonać w celu przejrzenia wszystkich następników charakterystyki różnicowej Δ_i.

Nazwa szyfru	**Rozmiar bloku [bity]**	**Rozmiar skrzynki [bity]**	**Średnia liczba operacji**	**Maksymalna liczba operacji**
MIDORI	64	4	$7^{64/4/2} \approx 2^{22.46}$	$7^{64/4} \approx 2^{44.92}$
KLEIN	64	4	$7^{64/4/2} \approx 2^{22.46}$	$7^{64/4} \approx 2^{44.92}$
AES	128	8	$127^{128/8/2} \approx 2^{55.91}$	$127^{128/8} \approx 2^{111.82}$

optymalne rozwiązanie. W szyfrach opartych na sieci SPN prawdopodobieństwo takiego zdarzenia nie wydaje się jednak znaczące. Trudno jednak uznać, że algorytm zwraca różniczki (Δ_{in}, Δ_{out}), które możemy z całą pewnością uznać za optymalne.

6.2 Metoda poszukiwania optymalnych charakterystyk różnicowych bazująca na modelu SMT

Metoda zakłada, że R rundowa charakterystyka różnicowa $(\Delta_0, \Delta_1, \Delta_2, \cdots, \Delta_R)$ to strumień $n \cdot R$ bitów od x_0 do $x_{n \cdot R-1}$, gdzie Δ_i to strumień bitów $(x_{n \cdot i}, x_{n \cdot i+1}, \cdots, x_{n \cdot i+n-1})$, a n to liczba bitów różnicy Δ_i. Strumień bitów $(x_0, x_1, \cdots, x_{n \cdot R-1})$ stanowi zbiór niewiadomych. Propagacja charakterystyki różnicowej określona jest poprzez funkcję $\mathbb{PE}$, a propagacja różnicy rundowej poprzez funkcję $\mathbb{PR}$. Rozwiązanie zadania polega na wyznaczeniu wartościowania, przy którym funkcja $\mathbb{PE}$ osiągnie wartość minimalną. Model opisujący rozważany problem przedstawiony jest jako zadanie optymalizacji SMT. W zaimplementowanym oprogramowaniu wykorzystano SMT solver Z3 [56] oraz algorytm SYMBA [47]. Decyzja ta została poprzedzona wstępnymi testami, w których sprawdzano różne algorytmy optymalizacji dostępne w Z3 i czas, w jakim radziły sobie z poszukiwaniem optymalnych ścieżek różnicowych. Konfigurację SMT solvera Z3 wykonaną z poziomu języka Python prezentuje kod źródłowy nr 6.1.

W celu uproszczenia modelu i uniknięcia operacji mnożenia wykorzystywane są tzw. zlogarytmowane profile różnicowe operacji nieliniowych, a w rzeczywistości szukana jest odwrotność prawdopodobieństwa. Zabieg ten zdecydowanie przyspiesza obliczenia i jest również stosowany w oprogramowaniu CryptoSMT [61]. Szukane prawdopodobieństwo zawsze będzie liczbą postaci 2^{-m}, dlatego przyjęcie za podstawę logarytmu liczby 2 nie wpłynie na wynik obliczeń. Graf AIG wykorzystywany do konstrukcji modelu SMT poddawany był również zabiegom pozwalającym zredukować jego rozmiar. W tym celu wykorzystano narzędzie ABC [18], które oferuje możliwość redukcji funkcjonalnej grafu AIG.

$$\mathbb{PR}(\Delta_i, \Delta_j) = \begin{cases} \log_2 \frac{1}{p}, & \text{jeżeli } p > 0 \text{ gdzie } p\text{: } \Delta_i \xrightarrow{\quad p \quad} \Delta_j \\ -1, & \text{jeżeli } p = 0 \text{ gdzie } p\text{: } \Delta_i \xrightarrow{\quad p \quad} \Delta_j \end{cases} \tag{6.1}$$

$$\mathbb{PE}(\Delta_0, \Delta_1, \Delta_2, \cdots, \Delta_n) = \begin{cases} \sum_{i=0}^{i=R} \mathbb{PR}(\Delta_i, \Delta_{i+1}), & \text{jeżeli } \nexists\ \mathbb{PR}(\Delta_i, \Delta_{i+1}) = -1 \\ -1, & \text{jeżeli } \exists\ \mathbb{PR}(\Delta_i, \Delta_{i+1}) = -1. \end{cases} \tag{6.2}$$

Opracowana metoda składa się z trzech zasadniczych etapów:

1. Opisania propagacji różniczek w zadanym algorytmie za pomocą grafu `AIG`. W praktyce będzie to zdefiniowanie funkcji $\mathbb{PR}$ oraz $\mathbb{PE}$, następnie zaprezentowanie funkcji $\mathbb{PE}$ za pomocą grafu `AIG`.

2. Konwersji grafu `AIG` do modelu definiującego zadanie optymalizacji `SMT`. Przed konwersją zalecane jest wykorzystanie metod pozwalających na zredukowanie rozmiaru grafu `AIG` takich jak *redukcja funkcjonalna* wierzchołków grafu czy też innych metod minimalizacji funkcji boolowskiej.

3. Wyznaczenia optymalnego rozwiązania modelu `SMT` (maksymalizacja wartości prawdopodobieństwa charakterystyki różnicowej).

Kod źródłowy 6.1: Powołanie obiektu `SMT`-solvera `Z3` oraz ustawienie specjalistycznych parametrów algorytmu za pomocą `API` języka `Python`.

```
1  # -*- coding: utf-8 -*-
2  from z3 import *
3
4
5  if __name__ == "__main__":
6
7      opt = Optimize()
8
9      opt.set(priority              = 'lex'       )
10     opt.set(optsmt_engine         = 'symba'     )
11     opt.set(maxsat_engine         = 'pd-maxres')
12     opt.set(enable_sls            = True        )
13     opt.set(enable_sat            = True        )
14     opt.set(dump_benchmarks       = True        )
15     opt.set(dump_models           = False       )
16     opt.set(elim_01               = False       )
17
18     opt.set("maxres.max_core_size"          ,  1024)
19     opt.set("maxres.max_correction_set_size",  1024)
20     opt.set("maxres.max_num_cores"          , 65536)
```

W ramach testów wydajności przedstawionej metody wykonano szereg badań polegających na wyszukaniu optymalnych charakterystyk różnicowych i charakterystyk różnicowych iteracyjnych dla szyfrów opartych na sieci `SPN` (`AES`, `MIDORI`, `KLEIN` oraz `PRESENT`). W tym celu zaimplementowano narzędzie, w którym w sposób automatyczny generowany jest model `SMT` opisujący propagację charakterystyki różnicowej w zadanym szyfrze na

wskazaną liczbę rund. Funkcję celu stanowi funkcja $\mathbb{PE}$. Następnie za pomocą *SMT solvera* poszukiwana jest minimalna wartość funkcji $\mathbb{PE}$ różna od -1. Wyniki badań zaprezentowano odpowiednio w tabelach 6.4 oraz 6.5. Do wyznaczenia rozwiązania modelu opisującego zadanie optymalizacji `SMT` wykorzystano *SMT solver* `Z3` w wersji 4.8.12 w konfiguracji zaprezentowanej w kodzie źródłowym nr 6.1.

Tablica 6.4: Czas poszukiwania charakterystyki różnicowej optymalnej pod kątem maksymalnej wartości prawdopodobieństwa w przypadku modelu opisującego zadanie optymalizacji i wykorzystaniu *SMT solver* `Z3`.

#R	AES		PRESENT		MIDORI		KLEIN	
	$\log_2(p)$	$\approx$ Czas	$\log_2(p)$	$\approx$ Czas	$\log_2(p)$	$\approx$ Czas	$\log_2(p)$	$\approx$ Czas
1	-6	8 h	-2	$<$ 1 s	-2	$<$ 1 s	-2	$<$ 1 s
2	-30	2 h	-4	$<$ 3 s	-8	$<$ 2 s	-10	9 s
3	-54	3 h	-8	8 s	-14	9 s	-17	150 s
4	-150	162 h	-12	20 s	-32	100 s	-31	11 h
5			-20	1190 s	-46	700 s		
6			-24	29 h	-60	27 h		
7			-28	44 h	-70	107 h		
8			-32	960 h				

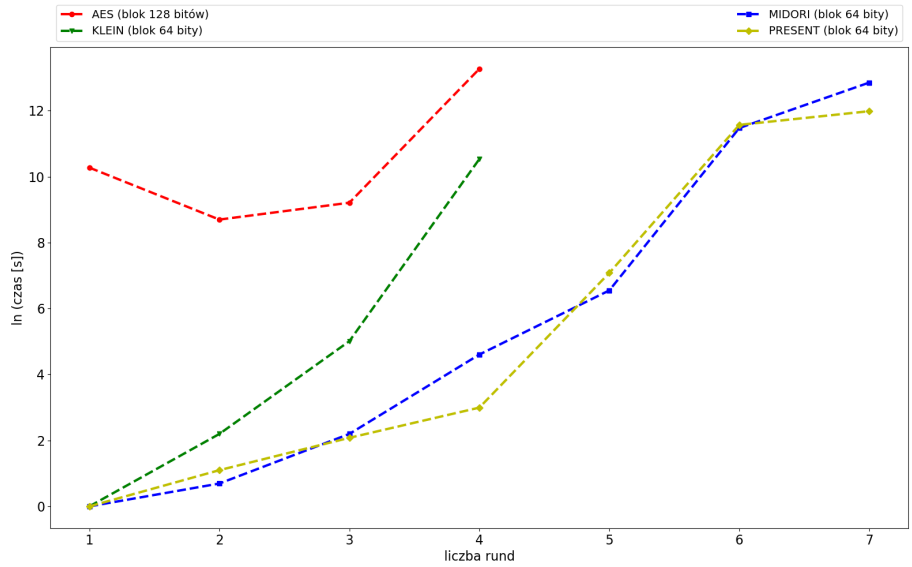
Tablica 6.5: Czas poszukiwania charakterystyki różnicowej iteracyjnej optymalnej pod kątem maksymalnej wartości prawdopodobieństwa w przypadku modelu opisującego zadanie optymalizacji i wykorzystaniu *SMT solver* `Z3`.

#R	AES		PRESENT		MIDORI		KLEIN	
	$\log_2(p)$	$\approx$ Czas	$\log_2(p)$	$\approx$ Czas	$\log_2(p)$	$\approx$ Czas	$\log_2(p)$	$\approx$ Czas
1	-52	16 h	-8	$<$ 2 s	-12	$<$ 2 s	-13	$<$ 5 s
2			-10	22 s	-24	8 s	-24	150 s
3			-24	2700 s	-36	75 s	-34	17 h
4			-18	3 h	-48	1730 s		
5			-29	185 h	-54	4 h		
6			-30	1076 h	-68	57 h		

6.2.1 Wydajność metody

Niestety przy poszukiwaniu w pełni określonych charakterystyk różnicowych wydajność[1] opracowanego narzędzia spada bardzo szybko wraz ze wzrostem liczby rund badanego szyfru blokowego. Czas potrzebny na rozwiązanie zadania rośnie wykładniczo, co zilustrowano na rysunku nr 6.2. Biorąc pod uwagę możliwości narzędzia `CryptoSMT` [61], zastosowane podejście okazało się lepsze jedynie w początkowej fazie obliczeń (wyznaczania optymalnych charakterystyk różnicowych na kilka początkowych rund szyfru). Warto

[1]Obliczenia wykonywane były przy pomocy *SMT solvera* `Z3` [56] na jednym wątku komputera wyposażonego w procesor Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50 GHz



Rysunek 6.2: Wydajność narzędzia do poszukiwania optymalnych charakterystyk różnicowych opartego na modelu `SMT` opisującym zadanie optymalizacji przy wykorzystaniu *SMT solver* `Z3` (skala logarytmiczna).

jednak zaznaczyć, że nawet przy tak słabej wydajności udało się wyznaczyć optymalne charakterystyki różnicowe dla 3 i 4 rund szyfru `AES`. Wyznaczenie optymalnej charakterystyki różnicowej na pełne wersje badanych szyfrów przy wykorzystaniu omówionego podejścia jest więc w praktyce nieosiągane ze względu ograniczeń czasowych.

6.3 Metody poszukiwania charakterystyk różnicowych bazujące na grafie `AIG` i modelu `SAT`

6.3.1 Metoda poszukiwania optymalnych charakterystyk różnicowych

Metoda zakłada, że R rundowa charakterystyka różnicowa $(\Delta_0, \Delta_1, \Delta_2, \cdots, \Delta_R)$ to strumień $n \cdot R$ bitów od x_0 do $x_{n \cdot R-1}$, gdzie Δ_i to strumień bitów $(x_{n \cdot i}, x_{n \cdot i+1}, \cdots, x_{n \cdot i+n-1})$, a n to liczba bitów różnicy Δ_i. Strumień bitów $(x_0, x_1, \cdots, x_{n \cdot R-1})$ stanowi zbiór niewiadomych. Propagacja charakterystyki różnicowej określona jest poprzez funkcję $\mathbb{PE}$, a propagacja charakterystyki rundowej poprzez funkcję $\mathbb{PR}$. Rozwiązanie zadania polega na wyznaczeniu wartościowania modelu `SAT` przy zadanym prawdopodobieństwie p lub przedziale liczbowym $< p_{low}, p_{up} >$, w którym powinno znajdować się prawdopodobieństwo p. Potwierdzenie, że dla zadanego prawdopodobieństwa p lub przedziału $< p_{low}, p_{up} >$ model nie jest spełnialny (taki model zwyczajowo określa się modelem `UNSAT`) jest jednoznaczne

z dowodem, że nie istnieje charakterystyka różnicowa spełniająca zadane kryterium (rozumiane jako zadana wartość prawdopodobieństwa charakterystyki różnicowej lub zadany przedział, w którym powinna znaleźć się wartość prawdopodobieństwa charakterystyki różnicowej).

W celu uproszczenia modelu i uniknięcia operacji mnożenia wykorzystywane są tzw. zlogarytmowane profile różnicowe operacji nieliniowych, a w rzeczywistości szukana jest odwrotność prawdopodobieństwa. Zabieg ten zdecydowanie przyspiesza obliczenia i jest również stosowany w oprogramowaniu `CryptoSMT` [61]. Szukane prawdopodobieństwo zawsze będzie liczbą postaci 2^{-m}, dlatego przyjęcia za podstawę logarytmu liczby 2 nie wpłynie na dokładność obliczeń.

$$\mathbb{PR}(\Delta_i, \Delta_j) = \begin{cases} \log_2 \frac{1}{p}, & \text{jeżeli } p > 0 \\ -1, & \text{jeżeli } p = 0 \end{cases} \tag{6.3}$$

$$\mathbb{PE}(\Delta_0, \Delta_1, \Delta_2, \cdots, \Delta_n) = \begin{cases} \sum_{i=0}^{i=R} \mathbb{PR}(\Delta_i, \Delta_{i+1}), & \text{jeżeli } \nexists\ \mathbb{PR}(\Delta_i, \Delta_{i+1}) = -1 \\ -1, & \text{jeżeli } \exists\ \mathbb{PR}(\Delta_i, \Delta_{i+1}) = -1. \end{cases} \tag{6.4}$$

Opracowana metoda składa się z trzech zasadniczych etapów:

1. Opisania propagacji różniczek w zadanym algorytmie za pomocą grafu `AIG`. W praktyce będzie to zdefiniowanie funkcji $\mathbb{PR}$ oraz $\mathbb{PE}$, następnie zaprezentowanie funkcji $\mathbb{PE}$ za pomocą grafu `AIG`.

2. Konwersji grafu `AIG` do modelu `SAT`. Przed konwersją zalecane jest wykorzystanie metod pozwalających na zredukowanie rozmiaru grafu `AIG` takich jak *redukcja funkcjonalna* wierzchołków grafu czy też innych metod minimalizacji funkcji boolowskiej.

3. Wyznaczenia wartościowania modelu przy zadanym prawdopodobieństwie p

   $$\mathbb{PE}(\Delta_0, \Delta_1, \Delta_2, \cdots, \Delta_n) = \log_2 \frac{1}{p}$$

   lub przedziale liczbowym $< p_{low}, p_{up} >$, w którym powinno znajdować się prawdopodobieństwo p

   $$\log_2 \frac{1}{p_{low}} \leq \mathbb{PE}(\Delta_0, \Delta_1, \Delta_2, \cdots, \Delta_n) \leq \log_2 \frac{1}{p_{up}}.$$

Poniżej zaprezentowany został kod źródłowy języka `Cryptol` [34], w którym za pomocą zlogarytmowanego profilu różnicowego opisano propagację charakterystyki różnicowej w algorytmie szyfrowania `PRESENT`.

Kod źródłowy 6.2: Skrypt języka `Cryptol` opisujacy propagację charakterystyki różnicowej w algorytmie `Present`.

```
1 module PRESENT_64_CLASSIC_DIFFERENTIAL_SEARCH where
```

```
2
3  type BLOCK_SIZE = 64
4  type DDT_TYPE = 16
5
6  pbox = [
7      0x00, 0x04, 0x08, 0x0c, 0x10, 0x14, 0x18, 0x1c,
8      0x20, 0x24, 0x28, 0x2c, 0x30, 0x34, 0x38, 0x3c,
9      0x01, 0x05, 0x09, 0x0d, 0x11, 0x15, 0x19, 0x1d,
10     0x21, 0x25, 0x29, 0x2d, 0x31, 0x35, 0x39, 0x3d,
11     0x02, 0x06, 0x0a, 0x0e, 0x12, 0x16, 0x1a, 0x1e,
12     0x22, 0x26, 0x2a, 0x2e, 0x32, 0x36, 0x3a, 0x3e,
13     0x03, 0x07, 0x0b, 0x0f, 0x13, 0x17, 0x1b, 0x1f,
14     0x23, 0x27, 0x2b, 0x2f, 0x33, 0x37, 0x3b, 0x3f
15 ]
16
17 sublayer : [BLOCK_SIZE] -> [BLOCK_SIZE]
18 sublayer input = [ input!(pbox ! i) | i <- ([0,1..63]:[_][8]) ]
19
20 DDT : [16][16][DDT_TYPE]
21 DDT = [
22     [ 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
23     [-1, -1, -1,  2, -1, -1, -1,  2, -1,  2, -1, -1, -1,  2, -1, -1],
24     [-1, -1, -1,  3, -1,  2,  3, -1, -1, -1,  3, -1,  3,  3,  3, -1],
25     [-1,  3, -1,  3,  3, -1,  2,  3, -1, -1,  3,  3, -1, -1, -1, -1],
26     [-1, -1, -1, -1, -1,  2,  3,  3, -1,  3,  3, -1,  3, -1,  3, -1],
27     [-1,  3, -1, -1,  3, -1, -1, -1, -1,  3,  3,  3,  2,  3, -1, -1],
28     [-1, -1,  3, -1, -1, -1,  3, -1,  3, -1, -1,  2,  3, -1, -1,  2],
29     [-1,  2,  3, -1, -1, -1,  3, -1,  3, -1, -1, -1,  3, -1, -1,  2],
30     [-1, -1, -1,  3, -1, -1, -1,  3, -1,  3, -1,  2, -1,  3, -1,  2],
31     [-1, -1,  3, -1,  2, -1,  3, -1,  3, -1, -1, -1,  3, -1,  2, -1],
32     [-1, -1,  3,  3, -1,  2, -1, -1,  3, -1,  3, -1, -1,  3,  3, -1],
33     [-1,  3, -1, -1,  3, -1, -1, -1,  2,  3,  3,  3, -1,  3, -1, -1],
34     [-1, -1,  3, -1, -1,  2, -1,  3,  3,  3,  3, -1, -1, -1,  3, -1],
35     [-1,  3,  2,  3,  3, -1, -1,  3, -1, -1,  3,  3, -1, -1, -1, -1],
36     [-1, -1,  3,  3, -1, -1,  3,  3,  3,  3, -1, -1,  3,  3, -1, -1],
37     [-1,  2, -1, -1,  2, -1, -1, -1, -1, -1, -1, -1, -1, -1,  2,  2]
38 ]
39
40 MAP_PROBABILITY : ([4], [4]) -> [DDT_TYPE]
41 MAP_PROBABILITY (x, y) = (DDT@x)@y
42
43 MAP_PROBABILITY_BLOCK : ([BLOCK_SIZE],[BLOCK_SIZE]) -> [2][DDT_TYPE]
44 MAP_PROBABILITY_BLOCK (in, out) = [sum(P), C]
45   where x = groupBy`{4}(in)
46         y = groupBy`{4}(out)
47         P = [0:[DDT_TYPE]] # [MAP_PROBABILITY(x@i, y@i) | i <- [0..15]:[_
               ][4]]
48         C = [ ((-1:[DDT_TYPE]) == P@i) | i <- [1..16]:[_][5] ]
49
50 MAP_SOLUTION : [32][BLOCK_SIZE] -> [32][BLOCK_SIZE]
51 MAP_SOLUTION X = mapping
```

```
52      where mapping = [ X@0 ] # [ sublayer(X@i) | i <- [1..31]:[_][5] ]
53
54  CALCULATE_PROBABILITY : [32][BLOCK_SIZE] -> [2][DDT_TYPE]
55  CALCULATE_PROBABILITY X = probability_e ! 0
56      where box_input     = [ X@0 ] # [ sublayer(X@i) | i <- [1..30]:[_][5] ]
57            box_output    = [ X@i | i <- [1..31]:[_][5] ]
58            probability_r = [ MAP_PROBABILITY_BLOCK(box_input@i, box_output@i)
                  | i <- [0..30]:[_][5] ]
59            probability_e = [ [0:[DDT_TYPE], 0:[DDT_TYPE]] ] # [ [(p@0 + e@0),
                  (p@1 || e@1)] | e <-probability_r | p <-probability_e ]
```

W przypadku szyfrów, gdzie za konfuzję odpowiadają operacje, dla których wyznaczenie profilu różnicowego jest zbyt czasochłonne, należy wykorzystać wzory opisujące ich własności różnicowe. W trakcie badań nie zidentyfikowano przekształcenia, przy którym propagacja charakterystyki różnicowej nie mogłaby być opisana za pomocą wzoru czy też profilu różnicowego. Poniżej znajduje się kod źródłowy języka `Cryptol`, w którym opisano propagację charakterystyki różnicowej w algorytmie szyfrowania `SIMON`.

Kod źródłowy 6.3: Skrypt języka `Cryptol` opisujacy propagację charakterystyki różnicowej w algorytmie `SIMON`.

```
1   module SIMON_32_CLASSIC_DIFFERENTIAL_SEARCH where
2
3   type BLOCK_SIZE   = 32
4   type H_BLOCK_SIZE = 16
5   type DDT_TYPE     = 16
6
7   hw : [H_BLOCK_SIZE] -> [DDT_TYPE]
8   hw x = h ! 0
9     where h = [ (0:[DDT_TYPE]) ] # [ if (x@i) then (t + (1:[DDT_TYPE])) else (
          t) | i <- [0..15] | t <- h ]
10
11  probability_and : ([H_BLOCK_SIZE], [H_BLOCK_SIZE], [H_BLOCK_SIZE]) -> [
        DDT_TYPE]
12  probability_and (L, B, G) = log2p
13    where log2p = if ((G && (~(L || B))) != 0) then (-1:[DDT_TYPE]) else (hw(L
          || B))
14
15  MAP_PROBABILITY_BLOCK : ([H_BLOCK_SIZE], [H_BLOCK_SIZE], [H_BLOCK_SIZE]) ->
        [2][DDT_TYPE]
16  MAP_PROBABILITY_BLOCK (L, B, G) = [ log2p, C ]
17    where log2p = probability_and(L, B, G)
18          C     = if (log2p == (-1:[DDT_TYPE])) then (-1:[DDT_TYPE]) else (0:[
                DDT_TYPE])
19
20  MAP_SOLUTION : [34][H_BLOCK_SIZE] -> [33][32]
21  MAP_SOLUTION X = __mapping__
22      where diff_maping = [ [(X@0), (X@1), ((X@2) ^ (X@1) ^ ((X@0) <<< 2))] ]
          # [ [d@2, (X@i), ((d@0) ^ (X@i) ^ ((d@2) <<< 2))] | i <- [3..33]:[_
          ][6] | d <- diff_maping ]
23            __mapping__ = [join([X@0, X@2])] # [ join([d@2, d@0]) | d <-
```

```
                diff_maping ]
24
25  CALCULATE_PROBABILITY : [34][H_BLOCK_SIZE] -> [2][DDT_TYPE]
26  CALCULATE_PROBABILITY X = probability_e ! 0
27      where diff_maping   = [ [(X@0), (X@1), ((X@2) ^ (X@1) ^ ((X@0) <<< 2))]
            ] # [ [d@2, (X@i), ((d@0) ^ (X@i) ^ ((d@2) <<< 2))] | i <- [3..33]:[_
            ][6] | d <- diff_maping ]
28             probability_r = [ MAP_PROBABILITY_BLOCK((d@0) <<< 1, (d@0) <<< 8,
                   (d@1)) | d <- diff_maping ]
29             probability_e = [ [0:[DDT_TYPE], 0:[DDT_TYPE]] ] # [ [((p@0) + (
                   e@0)), ((p@1) || (e@1))] | e <-probability_r | p <-
                   probability_e ]
```

W zaprezentowanych kodach źródłowych języka `Cryptol` [kod źródłowy nr 6.2 oraz 6.3] funkcja `MAP_PROBABILITY_BLOCK` odpowiada funkcji $\mathbb{PR}$, a funkcji $\mathbb{PE}$ odpowiada funkcja `CALCULATE_PROBABILITY`. Za pomocą narzędzia `SAW` [21], wskazana funkcja (wyznaczająca prawdopodobieństwo wskazanej charakterystyki różnicowej) z kodu języka `Cryptol` przedstawiana jest za pomocą grafu `AIG`. Tak przygotowany graf jest poddawany redukcji funkcjonalnej za pomocą narzędzia `ABC` [18] oraz innym zabiegom pozwalającym maksymalnie zredukować jego rozmiar. Następnie wykonywana jest konwersja zredukowanego grafu `AIG` do postaci `CNF` (ang. *Conjunctive Normal Form*).

```
Treść komunikatu                                  | AIGER info                                | Czas [s]
--------------------------------------------------- | ----------------------------------------- | ----------------
Maksymalny czas iteracji                          |                                           |      295200.00
Solver 'cadical'                                  |                                           |
Wygenerowano graf AIG                             | Liczba bramek AND                19624    |           1.44
Przekonwertowano graf AIG do układu równań ANF    | Liczba bramek AND                12046    |          35.03
                                                  | Fraig wsk                           38.62 |

Nazwa szyfru         |   R | log2(1/P) |         Status |       Czas konwersji [s] |        Czas iteracji [s] |        Czas obliczeń [s]
PRESENT-64-80        |   4 |         1 |   UNSATISFIABLE |                     0.21 |                     1.11 |                     1.33
PRESENT-64-80        |   4 |         2 |   UNSATISFIABLE |                     0.31 |                     1.02 |                     2.66
PRESENT-64-80        |   4 |         3 |   UNSATISFIABLE |                     0.32 |                     0.92 |                     3.91
PRESENT-64-80        |   4 |         4 |   UNSATISFIABLE |                     0.23 |                     1.21 |                     5.35
PRESENT-64-80        |   4 |         5 |   UNSATISFIABLE |                     0.31 |                     1.21 |                     6.88
PRESENT-64-80        |   4 |         6 |   UNSATISFIABLE |                     0.34 |                     1.41 |                     8.63
PRESENT-64-80        |   4 |         7 |   UNSATISFIABLE |                     0.33 |                     1.23 |                    10.21
PRESENT-64-80        |   4 |         8 |   UNSATISFIABLE |                     0.30 |                     1.24 |                    11.74
PRESENT-64-80        |   4 |         9 |   UNSATISFIABLE |                     0.33 |                     1.40 |                    13.48
PRESENT-64-80        |   4 |        10 |   UNSATISFIABLE |                     0.34 |                     1.39 |                    15.21
PRESENT-64-80        |   4 |        11 |   UNSATISFIABLE |                     0.34 |                     1.45 |                    17.00
PRESENT-64-80        |   4 |        12 |     SATISFIABLE |                     1.34 |                     1.23 |                    19.13

Wejście grafu AIG: [0x000000000000700F, 0x0000000000001001, 0x0000000000000004, 0x0000000300000000, 0x000000000D000700]

Wejście   | 0x000000000000700F
--------- | ------------------
Runda   1 | 0x0000000000000009
Runda   2 | 0x0000000100000000
Runda   3 | 0x0000000001000100
Runda   4 | 0x0040004400040044
```

Rysunek 6.3: Wynik programu po uruchomieniu zadania polegającego na znalezieniu optymalnej charakterystyki różnicowej dla 4 rund algorytmu `PRESENT`.

6.3.2 Metoda poszukiwania charakterystyk różnicowych obciętych

W przypadku poszukiwania charakterystyk różnicowych obciętych wykorzystano analogiczne podejście, w którym zamiast zlogarytmowanego profilu różnicowego wykorzystano zlogarytmowany dyfuzyjny profil różnicowy. Oczywiście wartości prawdopodobieństwa, z jakim propaguje się charakterystyka różnicowa obcięta nie są już liczbami postaci 2^{-m}, dlatego wykorzystano również inną podstawę logarytmu (równą 10).

$$\mathbb{PR}_{trunc}(\Delta_i, \Delta_j) = \begin{cases} \log_{10} \frac{1}{p}, & \text{jeżeli } p > 0 \text{ gdzie } p\text{: } \Delta_i \xrightarrow{\quad p \quad} \Delta_j \\ -1, & \text{jeżeli } p = 0 \text{ gdzie } p\text{: } \Delta_i \xrightarrow{\quad p \quad} \Delta_j \end{cases} \tag{6.5}$$

$$\mathbb{PE}_{trunc}(\Delta_0, \Delta_1, \cdots, \Delta_n) = \begin{cases} \sum_{i=0}^{i=R} \mathbb{PR}_{trunc}(\Delta_i, \Delta_{i+1}), & \text{jeżeli } \nexists\ \mathbb{PR}_{trunc}(\Delta_i, \Delta_{i+1}) = -1 \\ -1, & \text{jeżeli } \exists\ \mathbb{PR}_{trunc}(\Delta_i, \Delta_{i+1}) = -1. \end{cases} \tag{6.6}$$

Odróżnienie w pełni określonej charakterystyki różnicowej od szumu jest intuicyjne, ponieważ wiadomo, że prawdopodobieństwo jej wystąpienia powinno być wyższe niż 2^{-n}, gdzie liczba n określa długość bloku w bitach. W przypadku charakterystyk różnicowych obciętych należy zwrócić uwagę na końcową postać różniczki, a dokładniej na rozkład symboli w niej zawartych. Wystąpienie każdego symbolu oznaczonego poprzez 0 zachodzi z prawdopodobieństwem 2^{-l}, a symbolu oznaczonego poprzez $*$ z prawdopodobieństwem $\frac{2^l-1}{2^l}$. Liczba l określa długość słowa liczoną w bitach, na jakim agregowana była różniczka obcięta (dla bajtów $l = 8$, dla wartości heksadecymalnych $l = 4$). Prawdopodobieństwo uzyskania różniczki Δ_i, możemy określić zatem za pomocą następującego wzoru:

$$\mathbb{P}_{noice}(\Delta_i) = 2^{-l \cdot ||0||} \cdot \frac{2^l - 1}{2^l}^{||*||}, \tag{6.7}$$

gdzie $||0||$ to liczność symboli równych 0, a $||*||$ to liczność symboli $*$ (różnych od 0) w różniczce Δ_i. Użyteczna charakterystyka różnicowa obcięta $(\Delta_0, \Delta_1, \cdots, \Delta_n)$ powinna spełniać następującą zależność:

$$\mathbb{PE}_{trunc}(\Delta_0, \Delta_1, \cdots, \Delta_n) < \log_{10} \mathbb{P}_{noice}(\Delta_n). \tag{6.8}$$

Kod źródłowy 6.4: Skrypt języka `Cryptol` opisujacy propagację charakterystyki różnicowej obciętej w algorytmie `AES`.

```
1  module AES_TRUNC_PROPAGATION where
2
3  // precision = 5
4  // logarithm_base = 10
5
6  type AES128           = 4
7  type Nk               = AES128
```

```
type Nb               = 4
type Nr               = 6 + Nk
type STATE            = [4][Nb][1]
type TRUNC_BLOCK_SIZE = 16
type DDT_TYPE_TRUNC   = 32

MSG_TO_STATE : [TRUNC_BLOCK_SIZE] -> STATE
MSG_TO_STATE msg = transpose (split (split msg))

STATE_TO_MSG : STATE -> [TRUNC_BLOCK_SIZE]
STATE_TO_MSG st = join (join (transpose st))

SHIFT_ROWS : STATE -> STATE
SHIFT_ROWS state = [ row <<< shiftAmount | row <- state | shiftAmount <- [0
    .. 3] ]

AES_TRUNC_LINEAR_PART : [TRUNC_BLOCK_SIZE] -> [TRUNC_BLOCK_SIZE]
AES_TRUNC_LINEAR_PART x = STATE_TO_MSG((SHIFT_ROWS(MSG_TO_STATE(x))))

DDT_TRUNC : [16][16][DDT_TYPE_TRUNC]
DDT_TRUNC = [
    [
             0,     -1,     -1,     -1,     -1,     -1,     -1,     -1,
            -1,     -1,     -1,     -1,     -1,     -1,     -1,     -1
    ],
    [
            -1,     -1,     -1,     -1,     -1,     -1,     -1,     -1,
            -1,     -1,     -1,     -1,     -1,     -1,     -1,      0
    ],
    [
            -1,     -1,     -1,     -1,     -1,     -1,     -1,     -1,
            -1,     -1,     -1,     -1,     -1,     -1,     -1,      0
    ],
    [
            -1,     -1,     -1,     -1,     -1,     -1,     -1, 240654,
            -1,     -1,     -1, 240654,     -1, 240654, 240654,    171
    ],
    [
            -1,     -1,     -1,     -1,     -1,     -1,     -1,     -1,
            -1,     -1,     -1,     -1,     -1,     -1,     -1,      0
    ],
    [
            -1,     -1,     -1,     -1,     -1,     -1,     -1, 240654,
            -1,     -1,     -1, 240654,     -1, 240654, 240654,    171
    ],
    [
            -1,     -1,     -1,     -1,     -1,     -1,     -1, 240654,
            -1,     -1,     -1, 240654,     -1, 240654, 240654,    171
    ],
    [
            -1,     -1,     -1, 481308,     -1, 481308, 481308, 240825,
```

```
            -1, 481308, 481308, 240825, 481308, 240825, 240825,   685
    ],
    [
            -1,     -1,     -1,     -1,     -1,     -1,     -1,     -1,
            -1,     -1,     -1,     -1,     -1,     -1,     -1,      0
    ],
    [
            -1,     -1,     -1,     -1,     -1,     -1,     -1, 240654,
            -1,     -1,     -1, 240654,     -1, 240654, 240654,    171
    ],
    [
            -1,     -1,     -1,     -1,     -1,     -1,     -1, 240654,
            -1,     -1,     -1, 240654,     -1, 240654, 240654,    171
    ],
    [
            -1,     -1,     -1, 481308,     -1, 481308, 481308, 240825,
            -1, 481308, 481308, 240825, 481308, 240825, 240825,   685
    ],
    [
            -1,     -1,     -1,     -1,     -1,     -1,     -1, 240654,
            -1,     -1,     -1, 240654,     -1, 240654, 240654,    171
    ],
    [
            -1,     -1,     -1, 481308,     -1, 481308, 481308, 240825,
            -1, 481308, 481308, 240825, 481308, 240825, 240825,    685
    ],
    [
            -1,     -1,     -1, 481308,     -1, 481308, 481308, 240825,
            -1, 481308, 481308, 240825, 481308, 238325, 240825,    685
    ],
    [
            -1, 721962, 721962, 481479, 721962, 481479, 481479, 241339,
        721962, 481479, 481479, 241339, 481479, 241339, 241339,    680
    ]
]

ZERO_P = 240824:[DDT_TYPE_TRUNC]
ONE__P = 170:[DDT_TYPE_TRUNC]

P_NOISE : [TRUNC_BLOCK_SIZE] -> [DDT_TYPE_TRUNC]
P_NOISE delat_in = p ! 0
  where p = [ 0:[DDT_TYPE_TRUNC] ] # [ if delat_in@i then (v + ONE__P) else
     (v + ZERO_P) | i <- [0..15]:[_][4] | v <- p ]

MAP_TRUNC_PROBABILITY : ([4], [4]) -> [DDT_TYPE_TRUNC]
MAP_TRUNC_PROBABILITY (x, y) = (DDT_TRUNC@x)@y

MAP_TRUNC_PROBABILITY_BLOCK : ([TRUNC_BLOCK_SIZE],[TRUNC_BLOCK_SIZE]) ->
   [2][DDT_TYPE_TRUNC]
MAP_TRUNC_PROBABILITY_BLOCK (delat__in, delta_out) = [sum(log10p), valid]
  where x      = transpose(MSG_TO_STATE(delat__in))
```

```
107         y      = transpose(MSG_TO_STATE(delta_out))
108         log10p = [ MAP_TRUNC_PROBABILITY(join(x@i), join(y@i)) | i <-
                [0..3]:[_][2] ]
109         is_pos = [ ((-1:[DDT_TYPE_TRUNC]) == (log10p@i)) | i <- [0..3]:[_
                ][2] ]
110         valid  = if ((is_pos && 0xF) == 0x0) then 0:[DDT_TYPE_TRUNC] else
                -1:[DDT_TYPE_TRUNC]
111
112 CALCULATE_TRUNC_PROBABILITY : [4][TRUNC_BLOCK_SIZE] -> [2][DDT_TYPE_TRUNC]
113 CALCULATE_TRUNC_PROBABILITY X = [ (probability_e ! 0)@0, is_valid ]
114     where char__input   = [ AES_TRUNC_LINEAR_PART(X@i) | i <- [0..2]:[_][2]
            ]
115           char_output   = [ X@i | i <- [1..3]:[_][2] ]
116           probability_r = [ MAP_TRUNC_PROBABILITY_BLOCK(char__input@i,
                  char_output@i) | i <- [0..2]:[_][2] ]
117           probability_e = [ [0:[DDT_TYPE_TRUNC], 0:[DDT_TYPE_TRUNC]] ] # [
                  [(p@0 + e@0), (p@1 || e@1)] | e <-probability_r | p <-
                  probability_e ]
118           probability_n = P_NOISE(char_output ! 0)
119           is_noise      = ((probability_e ! 0)@0 >= probability_n) || (
                  char__input@0 == 0)
120           is_valid      = if is_noise then (-1:[DDT_TYPE_TRUNC]) else ((
                  probability_e ! 0)@1)
121
122 MAP_SOLUTION : [4][TRUNC_BLOCK_SIZE] -> [4][TRUNC_BLOCK_SIZE]
123 MAP_SOLUTION X = mapping
124     where mapping = X
```

Zaprezentowany kod źródłowy języka `Cryptol` [kod źródłowy nr 6.4] opisuje propagację charakterystyki różnicowej obciętej w algorytmie `AES` zredukowanym do 3 rund.

Funkcja o nazwie `MAP_TRUNC_PROBABILITY_BLOCK` odpowiada funkcji $\mathbb{PR}_{trunc}$, funkcja o nazwie `CALCULATE_TRUNC_PROBABILITY` funkcji $\mathbb{PE}_{trunc}$, a funkcja `P_NOISE` funkcji $\mathbb{P}_{noice}$. W celu uniknięcia operacji zmiennoprzecinkowych wartości w dyfuzyjnym profilu różnicowym (zmienna `DDT_TRUNC`) po zlogarytmowaniu zostały pomnożone przez stałą postaci 10^n (w przypadku zaprezentowanego kodu była to wartość 10^5), a następnie zaokrąglone w górę do najbliższej liczby całkowitej. Zabieg ten pozwala przyspieszyć czas określenia wartościowania modelu `SAT` generowanego na bazie grafu `AIG` opisującego funkcję `CALCULATE_TRUNC_PROBABILITY`.

6.3.3 Wyniki badań związanych z poszukiwaniem charakterystyk różnicowych i charakterystyk różnicowych obciętych

W ramach testów opracowanej metody automatycznego poszukiwania optymalnych charakterystyk różnicowych wybrano kilka współczesnych szyfrów blokowych opartych na różnych konstrukcjach. Był to aktualny standard szyfrowania `AES` (*Rijndael*) i algorytmy o podobnej konstrukcji, czyli `MIDORI` oraz `KLEIN`. Kolejnym szyfrem był jeden z flagowych algorytmów kryptografii lekkiej, a mianowicie szyfr blokowy `PRESENT` oraz inni przedstawi-

```
Treść komunikatu                                  | AIGER info                                | Czas [s]
------------------------------------------------- | ----------------------------------------- | -----------------
Maksymalny czas iteracji                          |                                           |      295200.00
Solver 'cadical'                                  |                                           |
Wygenerowano graf AIG                             | Liczba bramek AND                 9878    |           1.24
Przekonwertowano graf AIG do układu równań ANF    | Liczba bramek AND                 5793    |          18.97
                                                   | Fraig wsk                           41.35 |

Nazwa szyfru         |   R | log2(1/P) |         Status |      Czas konwersji [s] |       Czas iteracji [s] |       Czas obliczeń [s]
AES-128-128          |   3 |         1 |   UNSATISFIABLE |                    0.14 |                    0.49 |                    2.14
AES-128-128          |   3 |         2 |   UNSATISFIABLE |                    1.15 |                    0.58 |                    4.46
AES-128-128          |   3 |         3 |   UNSATISFIABLE |                    0.14 |                    0.27 |                    6.44
AES-128-128          |   3 |         4 |   UNSATISFIABLE |                    0.09 |                    0.26 |                    8.31
AES-128-128          |   3 |         5 |   UNSATISFIABLE |                    0.10 |                    0.36 |                    9.77
AES-128-128          |   3 |         6 |   UNSATISFIABLE |                    0.14 |                    0.27 |                   11.66
AES-128-128          |   3 |         7 |   UNSATISFIABLE |                    0.14 |                    0.46 |                   13.61
AES-128-128          |   3 |         8 |   UNSATISFIABLE |                    0.14 |                    0.48 |                   15.85
AES-128-128          |   3 |         9 |   UNSATISFIABLE |                    0.14 |                    0.63 |                   18.24
AES-128-128          |   3 |        10 |   UNSATISFIABLE |                    0.14 |                    0.55 |                   20.48
AES-128-128          |   3 |        11 |   UNSATISFIABLE |                    0.15 |                    0.56 |                   22.73
AES-128-128          |   3 |        12 |   UNSATISFIABLE |                    0.15 |                    0.63 |                   24.94
AES-128-128          |   3 |        13 |   UNSATISFIABLE |                    0.15 |                    0.48 |                   27.18
AES-128-128          |   3 |        14 |   UNSATISFIABLE |                    0.15 |                    0.87 |                   29.57
AES-128-128          |   3 |        15 |   UNSATISFIABLE |                    0.15 |                    0.96 |                   32.25
AES-128-128          |   3 |        16 |   UNSATISFIABLE |                    0.17 |                    0.90 |                   34.66
AES-128-128          |   3 | 16.062154 |     SATISFIABLE |                    0.13 |                    0.76 |                   37.14

Wejście grafu AIG: [0x0080, 0x00F0, 0xFFFF, 0x3FFF]


Wejście   | 0x0080
--------- | ------
Runda   1 | 0x00F0
Runda   2 | 0xFFFF
Runda   3 | 0x3FFF
```

Rysunek 6.4: Wynik programu po uruchomieniu zadania polegającego na znalezieniu charakterystyki różnicowej obciętej dla 3 rund algorytmu AES.

ciele tego nurtu w kryptografii, a konkretnie szyfry SPECK, SIMON, PYJAMASK oraz SATURNIN. Dwa ostatnie z wymienionych szyfrów znalazły się w drugiej rundzie konkursu organizowanego przez NIST w celu wyłonienia lekkiego standardu szyfrowania z uwierzytelnieniem.

Dla szyfrów blokowych KLEIN oraz MIDORI (z 64-bitowym blokiem), PRESENT oraz SIMON udało się wyznaczyć optymalne charakterystyki różnicowe na pełne wersje algorytmów. Przykładowe charakterystyki, ich prawdopodobieństwo oraz czas, w jakim określono postać charakterystyki zaprezentowano kolejno w tabelach 6.6, 6.7, 6.8 oraz 6.9. Pierwsze trzy z wymienionych szyfrów oparte są na sieci SPN, a w celu osiągnięcia odpowiedniego poziomu konfuzji wykorzystują warstwę 4-bitowych skrzynek podstawieniowych. Każdy z tych szyfrów wykorzystuje w warstwie liniowej odmienny zestaw prymitywów kryptograficznych. W algorytmie PRESENT mamy do czynienia z prostą permutacją bitową. W przypadku szyfru MIDORI wykorzystywany jest aMDS (ang. *almost Maximum Distance Separable*) oraz permutacja zorientowana na znaki heksadecymalne, a w przypadku szyfru KLEIN macierz MDS (identyczna jak w specyfikacji algorytmu AES) oraz rotacja 2-bajtowa. Szyfr SIMON jest szyfrem opartym na sieci Feistela, na którego rundę składają się jedynie bitowe rotacji oraz operacje AND [rysunek nr 2.7].

Tablica 6.6: Optymalna charakterystyka różnicowa (pod kątem maksymalnej wartości prawdopodobieństwa) dla pełnej wersji algorytmu KLEIN (blok 64 bity) uzyskana w wyniku rozwiązania zadania bazującego na grafie AIG i modelu SAT.

R	$\boldsymbol{\Delta_i}$	Optymalna charakterystyka różnicowa	$\mathbf{P}(\boldsymbol{\Delta_{in}} \rightarrow \boldsymbol{\Delta_{out}})$	≈ Czas [s]
12	Δ_0	010E000000000500	2^{-111}	44
	Δ_1	0000000000000D0B		
	Δ_2	000000000E0E0400		
	Δ_3	000E07090C06060A		
	Δ_4	0A0E000000000B0E		
	Δ_5	0000000003020000		
	Δ_6	04060D0B00000000		
	Δ_7	0206000405030C0F		
	Δ_8	0005060106000307		
	Δ_9	00000C0A090D0000		
	Δ_{10}	0000030300000000		
	Δ_{11}	0103000200000000		
	Δ_{12}	0F0A050502040501		

Tablica 6.7: Optymalna charakterystyka różnicowa (pod kątem maksymalnej wartości prawdopodobieństwa) dla pełnej wersji algorytmu MIDORI (blok 64 bity) uzyskana w wyniku rozwiązania zadania bazującego na grafie AIG i modelu SAT.

R	$\boldsymbol{\Delta_i}$	Optymalna charakterystyka różnicowa	$\mathbf{P}(\boldsymbol{\Delta_{in}} \rightarrow \boldsymbol{\Delta_{out}})$	≈ Czas [h]
16	Δ_0	A0000A0000F00000	2^{-168}	18
	Δ_1	000A000000000000		
	Δ_2	00000000A0AA0000		
	Δ_3	5055AA0A00005550		
	Δ_4	DA7AAA000AA0000A		
	Δ_5	F00A0505DD005505		
	Δ_6	0A000000000A00A0		
	Δ_7	00000A0000000000		
	Δ_8	AA0A000000000000		
	Δ_9	0AAAAAA0F0FF0000		
	Δ_{10}	0FF0A000AFA500AA		
	Δ_{11}	0A0A0A500AAA00AA		
	Δ_{12}	5000050000A00000		
	Δ_{13}	000A000000000000		
	Δ_{14}	00000000A0AA0000		
	Δ_{15}	505555050000AAA0		
	Δ_{16}	7DADAA000AA0DD07		

Tablica 6.8: Optymalna charakterystyka różnicowa (pod kątem maksymalnej wartości prawdopodobieństwa) dla pełnej wersji algorytmu PRESENT (blok 64 bity) uzyskana w wyniku rozwiązania zadania bazującego na grafie AIG i modelu SAT.

R	$\boldsymbol{\Delta_i}$	**Optymalna charakterystyka różnicowa**	$\mathbf{P(\Delta_{in} \rightarrow \Delta_{out})}$	**≈ Czas [h]**
31	Δ_0	0000000000001001	2^{-136}	3
	Δ_1	0009000000000009		
	Δ_2	0000100100000000		
	Δ_3	0900000000000900		
	Δ_4	0000400400000000		
	Δ_5	0000090000000900		
	Δ_6	0000040400000000		
	Δ_7	0000050000000500		
	Δ_8	0000000000000404		
	Δ_9	0000000500000005		
	Δ_{10}	0000000000000101		
	Δ_{11}	0005000000000005		
	Δ_{12}	0000000000001001		
	Δ_{13}	0009000000000009		
	Δ_{14}	0000100100000000		
	Δ_{15}	0900000000000900		
	Δ_{16}	0000400400000000		
	Δ_{17}	0000090000000900		
	Δ_{18}	0000040400000000		
	Δ_{19}	0000050000000500		
	Δ_{20}	0000000000000404		
	Δ_{21}	0000000500000005		
	Δ_{22}	0000000000000101		
	Δ_{23}	0005000000000005		
	Δ_{24}	0000000000001001		
	Δ_{25}	0009000000000009		
	Δ_{26}	0000100100000000		
	Δ_{27}	0900000000000900		
	Δ_{28}	0000400400000000		
	Δ_{29}	0000090000000900		
	Δ_{30}	0000040400000000		
	Δ_{31}	0000050000000500		

Tablica 6.9: Optymalna charakterystyka różnicowa (pod kątem maksymalnej wartości prawdopodobieństwa) dla pełnej wersji algorytmu SIMON (blok 32 bity) uzyskana w wyniku rozwiązania zadania bazującego na grafie AIG i modelu SAT.

R	$\boldsymbol{\Delta_i}$	**Optymalna charakterystyka różnicowa**	$\mathbf{P(\Delta_{in} \rightarrow \Delta_{out})}$	**≈ Czas [s]**
31	Δ_0	00800222	2^{-90}	1900
	Δ_1	00220080		
	Δ_2	00080022		
	Δ_3	00020008		
	Δ_4	00000002		
	Δ_5	00020000		

	Δ_6	00080002		
	Δ_7	00220008		
	Δ_8	00800022		
	Δ_9	02220080		
	Δ_{10}	08080222		
	Δ_{11}	22020808		
	Δ_{12}	80002202		
	Δ_{13}	22008000		
	Δ_{14}	08002200		
	Δ_{15}	02000800		
	Δ_{16}	00000200		
	Δ_{17}	02000000		
	Δ_{18}	08000200		
	Δ_{19}	22000800		
	Δ_{20}	80002200		
	Δ_{21}	22028000		
	Δ_{22}	08082202		
	Δ_{23}	02220808		
	Δ_{24}	00800222		
	Δ_{25}	00220080		
	Δ_{26}	00080022		
	Δ_{27}	00020008		
	Δ_{28}	00000002		
	Δ_{29}	00020000		
	Δ_{30}	00080002		
	Δ_{31}	00220008		
	Δ_{32}	00800022		

W przypadku algorytmu SPECK optymalną charakterystykę różnicową na pełną wersję algorytmu udało się wyznaczyć jedynie dla wersji z 32-bitowym blokiem, a rezultat tego badania zaprezentowano w tabeli 6.10. Dla wersji ze 96 oraz 128 bitowym blokiem udało się określić optymalną charakterystykę różnicową odpowiednio maksymalnie dla 11 i 10 rund szyfru. Zabieg ten nie udał się w przypadku badań opublikowanych w pracy [62]. Operacją nieliniową warunkującą sposób propagacji charakterystyki różnicowej w tym szyfrze jest dodawanie arytmetyczne w $GF(2^{\frac{n}{2}})$, gdzie n to długość przetwarzanego bloku. W rundzie szyfru wyróżnia się również rotacje bitowe (rysunek nr 2.6).

Tablica 6.10: Optymalna charakterystyka różnicowa (pod kątem maksymalnej wartości prawdopodobieństwa) dla pełnej wersji algorytmu SPECK (blok 32 bity) uzyskana w wyniku rozwiązania zadania bazującego na grafie AIG i modelu SAT.

R	$\mathbf{\Delta_i}$	**Optymalna charakterystyka różnicowa**	$\mathbf{P(\Delta_{in} \rightarrow \Delta_{out})}$	**≈ Czas [h]**
22	Δ_0	A0000010	2^{-85}	490
	Δ_1	00400000		
	Δ_2	80008000		
	Δ_3	81008102		
	Δ_4	8004840E		

	Δ_5	95028538		
	Δ_6	800294E0		
	Δ_7	9120C2A2		
	Δ_8	0280080B		
	Δ_9	083C2810		
	Δ_{10}	A0000040		
	Δ_{11}	01000000		
	Δ_{12}	00020002		
	Δ_{13}	0402040A		
	Δ_{14}	0006102E		
	Δ_{15}	1422549A		
	Δ_{16}	1062420B		
	Δ_{17}	02290A04		
	Δ_{18}	28000010		
	Δ_{19}	00400000		
	Δ_{20}	80008000		
	Δ_{21}	81008102		
	Δ_{22}	8000840A		

Tablica 6.11: Optymalna charakterystyka różnicowa (pod kątem maksymalnej wartości prawdopodobieństwa) dla 10 rund algorytmu SPECK (blok 128 bitów) uzyskana w wyniku rozwiązania zadania bazującego na grafie AIG i modelu SAT.

R	$\boldsymbol{\Delta_i}$	Optymalna charakterystyka różnicowa	$\mathbf{P}(\boldsymbol{\Delta_{in}} \rightarrow \boldsymbol{\Delta_{out}})$	$\approx$ Czas [h]
10	Δ_0	0808000000800A0808000000124A0848	2^{-49}	582
	Δ_1	00000000924000404000000000104200		
	Δ_2	00000000008202000000000000001202		
	Δ_3	00000000000090000000000000000010		
	Δ_4	00000000000008000000000000000000		
	Δ_5	80000000000000008000000000000000		
	Δ_6	80800000000000008080000000000004		
	Δ_7	80008000000000048400800000000020		
	Δ_8	80808080000000020A084808000000124		
	Δ_9	80040000800001248420040080000801		
	Δ_{10}	A0A000008080080081A020048080480C		

Dla szyfru AES z powodzeniem udało się określić optymalną charakterystykę różnicową maksymalnie dla 5 rund. Wyniki badania zaprezentowano w tabeli 6.12. Warto zwrócić uwagę, że dla 5, 6, jak i 7 rund nie istnieje charakterystyka różnicowa aktywująca minimalną liczbę skrzynek podstawioniowych, dla której przejście na każdej ze skrzynek wystąpiłoby z prawdopodobieństwem 2^{-6}. Biorąc pod uwagę minimalną liczbę aktywnych skrzynek wyznaczoną dla omówionej liczby rund (tabela 5.4) prawdopodobieństwo takiej charakterystyki byłoby równe odpowiednio 2^{-156} dla 5, 2^{-180} dla 6 rund oraz 2^{-204} dla 7.

Tablica 6.12: Optymalne charakterystyki różnicowe (pod kątem maksymalnej wartości prawdopodobieństwa) dla algorytmu AES (blok 128 bitów) uzyskane w wyniku rozwiązania zadania bazującego na grafie AIG i modelu SAT.

R	$\boldsymbol{\Delta_i}$	**Optymalna charakterystyka różnicowa**	$\mathbf{P}(\boldsymbol{\Delta_{in}} \rightarrow \boldsymbol{\Delta_{out}})$	**≈ Czas**
1	Δ_0	00000200000000000000000000000000	2^{-6}	300 s
	Δ_1	00000000000000000143C281400000000		
2	Δ_0	00000000000053000000000BB00000000	2^{-30}	4050 s
	Δ_1	00000000000000000000000007008787		
	Δ_2	74749CE8749CE8740000000057A6A6F1		
3	Δ_0	000000A39100000000A800000000FA00	2^{-54}	12800 s
	Δ_1	00000000000064000000000000000000		
	Δ_2	00000000000000000000000020604020		
	Δ_3	D4D467B36ABED46ACE7DB3B3B3D4D467		
4	Δ_0	00000059A000000000420000000006900	2^{-150}	25 h
	Δ_1	000000000FE0000000000000000000000		
	Δ_2	00000000ABD8D8730000000000000000		
	Δ_3	0604020202010103ECEC2FC302060402		
	Δ_4	5BBCC35541E9BC285991A1C4D2416C28		
5	Δ_0	000000000000000E0000000000000000	2^{-169}	1205 h
	Δ_1	000000000000000C8C8438B00000000		
	Δ_2	43C58643860D8B8B241212365E5EE2BC		
	Δ_3	000000F87F00000000D500000000DC00		
	Δ_4	0000000000004C000000000000000000		
	Δ_5	00000000000000000000000004ADE944A		
6	--	--------------------------------	$2^{-189} \leq \mathbf{P} \leq 2^{-192}$	–
7	--	--------------------------------	$2^{-215} \leq \mathbf{P} \leq 2^{-217}$	–

Przy szyfrach blokowych, dla których proces poszukiwania optymalnej charakterystyki różnicowej na pełen algorytm zakończył się niepowodzeniem, wykonano badanie, w którym poszukiwano charakterystyk na $R + k$ rund budowanych na podstawie R rundowej optymalnej charakterystyce różnicowej. Wyniki tego doświadczenia przedstawia tabela 6.13.

W ramach testów metody poszukiwania charakterystyk różnicowych obciętych przebadano szyfry blokowe PRESENT, AES, MIDORI (blok 64 bity) oraz KLEIN (blok 64 bity). W pierwszej kolejności wyznaczono dyfuzyjne profile różnicowe dla operacji liniowych w tych algorytmach[2]. Model poszukiwania charakterystyki różnicowej obciętej zakładał, że prawdopodobieństwo wystąpienia charakterystyki na wyjściu musi być większe, niż wynika to z prawdopodobieństwa pojawienia się tej charakterystyki w szumie (własność opisana poprzez równanie nr 6.8). Wyniki badań zaprezentowano w tabeli 6.14.

[2]Dla algorytmu PRESENT dyfuzyjny profil różnicowy został wyznaczony w sposób przybliżony ze względu na rozmiar bloku, na jakim działa wykorzystywana w nim operacja permutacji bitowej tj. 64 bity. Aby umożliwić przechowanie go w pamięci, RAM podczas budowy grafu AIG został zorientowany na 8-bitowe słowa

Tablica 6.13: Logarytm przy podstawie 2 z prawdopodobieństwa najlepszej charakterystyki różnicowej wyznaczonej na zadaną liczbę rund za pomocą metody opartej na grafie AIG i model SAT dla wybranych szyfrów blokowych.

#R	AES (blok 128 bitów)	PYJAMASK (blok 96 bitów)	PYJAMASK (blok 128 bitów)	SPECK (blok 96 bitów)	SPECK (blok 128 bitów)	SATURNIN (blok 256 bitów)
	$\log_2 \mathbf{P}$	$\log_2 \mathbf{P}$	$\log_2 \mathbf{P}$	$\log_2 \mathbf{P}$	$\log_2 \mathbf{P}$	$\log_2 \mathbf{P}$
1	-6	-2	-2	0	0	-13
2	-30	-24	-24	-1	-1	-77
3	-54	-38	-38	-3	-3	
4	-150	-68	-70	-6	-6	
5	-169	-98	-116	-10	-10	
6	-192		-160	-15	-15	
7	-217			-21	-21	
8	-318			-30	-30	
9	-333			-39	-39	
10	-358			-49	-49	
11	-381			-58	-58	
12				-66	-66	
13				-68	-73	
14				-72	-81	
15				-81	-87	
16				-88	-96	
17				-96	-104	
18					-113	
19					-119	
20					-128	

	charakterystyka różnicowa optymalna pod kątem maksymalnej wartości prawdopodobieństwa
	charakterystyka różnicowa optymalna pod kątem maksymalnej wartości prawdopodobieństwa nie wskazana wcześniej w literaturze
	charakterystyka różnicowa o prawdopodobieństwie większym niż charakterystyki wskazywane w literaturze

6.3.4 Wydajność metody oraz własności modeli SAT opisujących propagację charakterystyki różnicowej generowanych na bazie grafów AIG

Konwersja grafu AIG do modelu SAT odbywała się pośrednio poprzez postać ANF. Do konwersji ANF do CNF wykorzystano zmodyfikowane na potrzeby prowadzonych badań naukowych oprogramowania typu *open source* dostępne pod adresem:

`https://www.lukbettale.ze.cx/anf2cnf`.

Możliwe, że inne metody konwersji mogą wpłynąć na skuteczność rozwiązywania modeli SAT określonych poprzez zbiór klauzul formacie CNF. Niewykluczone, że w przyszłości w ramach usprawnienia zaimplementowanego narzędzia wykonane zostanie badanie pozwalające wybrać optymalny sposób konwersji.

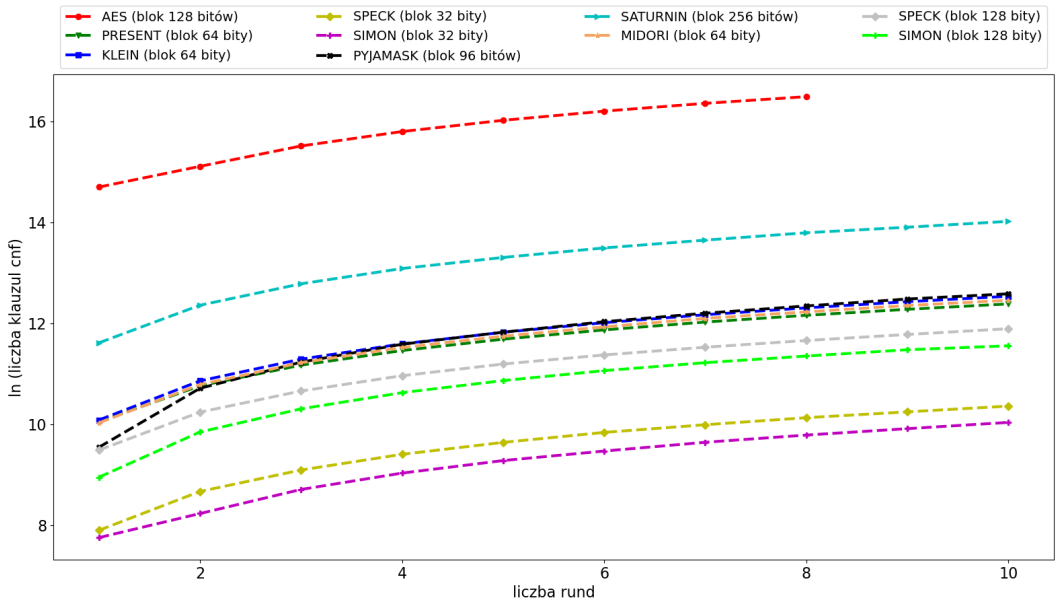
Tablica 6.14: Wyniki badań związanych z poszukiwaniem charakterystyki różnicowej obciętej za pomocą metody bazującej na grafie AIG i modelu SAT dla wybranych szyfrów blokowych (przy założeniu, że prawdopodobieństwo szukanej charakterystyki różnicowej obciętej ma być niższe niż prawdopodobieństwo wystąpienia tej charakterystyki w szumie).

#R	AES orientowane bajtowo		PRESENT orientowane bajtowo		MIDORI orientowane heksadecymalnie		KLEIN orientowane heksadecymalnie	
	$\approx \mathbb{P}$	$\approx$ Czas	$\approx \mathbb{P}$	$\approx$ Czas	$\approx \mathbb{P}$	$\approx$ Czas	$\approx \mathbb{P}$	$\approx$ Czas
2	$2^{-0.051158}$	< 3 s	$2^{-4.746670}$	1100 s	1.0	< 3 s	$2^{-4.236123}$	2950 s
3	$2^{-16.062154}$	40 s	$2^{-11.580673}$	3400 s	$2^{-0.720028}$	< 5 s	$2^{-9.830084}$	4 h
4	$2^{-64.022672}$	200 s	$2^{-17.961732}$	2 h	$2^{-16.657875}$	110 s	$2^{-20.033851}$	17 h
5	$2^{-120.005642}$	540 s	$2^{-24.434043}$	4 h	$2^{-50.889048}$	830 s	$2^{-26.686078}$	35 h
6	UNSAT	620 s	$2^{-30.825167}$	6 h	UNSAT	840 s	$2^{-46.104043}$	113 h
7	UNSAT	670 s	$2^{-43.320234}$	12 h	UNSAT	850 s	UNSAT	245 h
8	UNSAT	690 s	$2^{-51.297679}$	18 h	UNSAT	780 s	UNSAT	305 h
9	UNSAT	710 s	UNSAT	27 h	UNSAT	760 s	UNSAT	400 h
10	UNSAT	780 s	UNSAT	32 h	UNSAT	700 s	UNSAT	450 h
11	UNSAT	830 s			UNSAT	690 s		
12	UNSAT	850 s			UNSAT	680 s		
13	UNSAT	940 s			UNSAT	690 s		
14	UNSAT	1100 s			UNSAT	720 s		

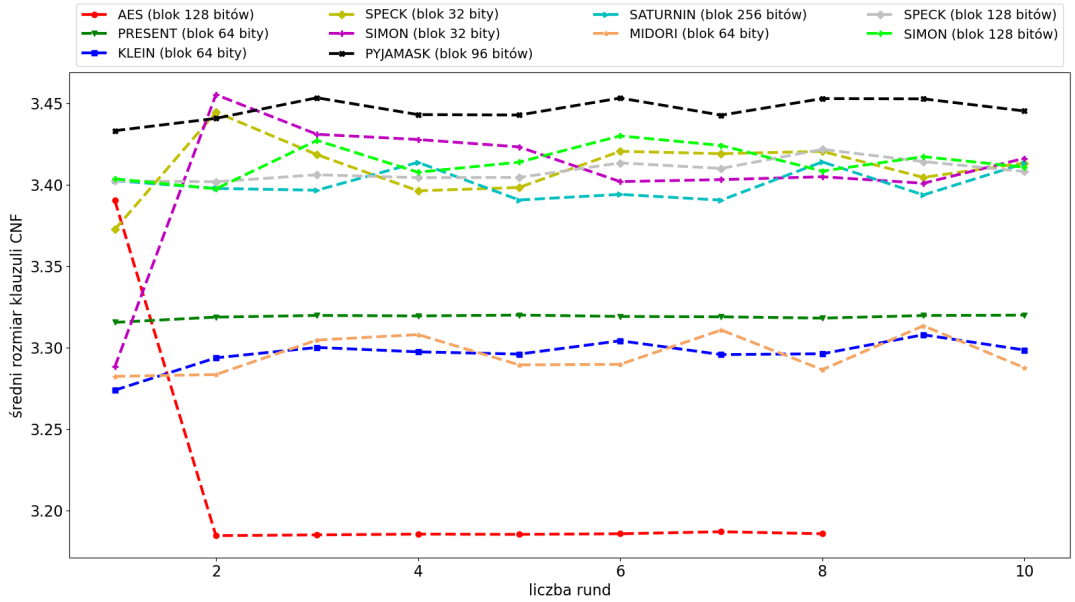
Zgodnie z przypuszczeniami liczba zmiennych oraz liczba klauzul rośnie liniowo dla każdego z badanych algorytmów wraz z liczbą rund. Rysunek nr 6.5 przedstawia ową zależność. Warto jednak podkreslić, że rozmiar modelu SAT (wyrażony w liczbie zmiennych i liczbie klauzul CNF) nie jest czynnikiem decydującym o czasie rozwiązania. Dobrym przykładem jest tutaj algorytm PYJAMASK, dla którego propagacja charakterystyki różnicowej została opisana poprzez model SAT o rozmiarach podobnych jak w przypadku szyfrów KLEIN, MIDORI oraz PRESENT. Pomimo tego czas potrzebny na określenia wartościowania modelu rośnie znacznie szybciej niż w przypadku wymienionych wcześniej szyfrów.

Na rysunku nr 6.6 przedstawiono również średni rozmiar klauzuli CNF w modelach SAT generowanych na bazie grafów AIG, opisujących propagację charakterystyki różnicowej dla pierwszych 10 rund dla każdego z badanych szyfrów. Widać na nim wyraźnie, że własność ta jest zbliżona dla szyfrów o podobnej konstrukcji oraz wykorzystujących podobne prymitywy kryptograficzne. W przypadku szyfrów blokowych PRESENT, KLEIN oraz MIDORI będzie to struktura oparta na sieci SPN wykorzystująca 4-bitową skrzynkę podstawieniową. W przypadku algorytmów SATURNIN, SPECK oraz SIMON wspólnym mianownikiem jest konstrukcja wykorzystująca operacje charakterystyczne dla ARX (m.in. dodawanie arytmetyczne, operacja bitowa AND). Szyfr PYJAMASK jako jedyny w zbiorze badanych algorytmów wykorzystuje kilka różnych macierzy MDS, a szyfr AES 8-bitową skrzynkę podstawieniową.

Wydajność zaimplementowanego narzędzia jest bardzo dobra i zdecydowanie lepsza niż oprogramowania CryptoSMT [61]. Rysunek nr 6.7 przedstawia wykresy, na których po-



Rysunek 6.5: Dynamika wzrostu liczby klauzul `CNF` w modelach `SAT` generowanych na bazie grafów `AIG`, opisujących propagację charakterystyki różnicowej względem liczby rund (skala logarytmiczna).

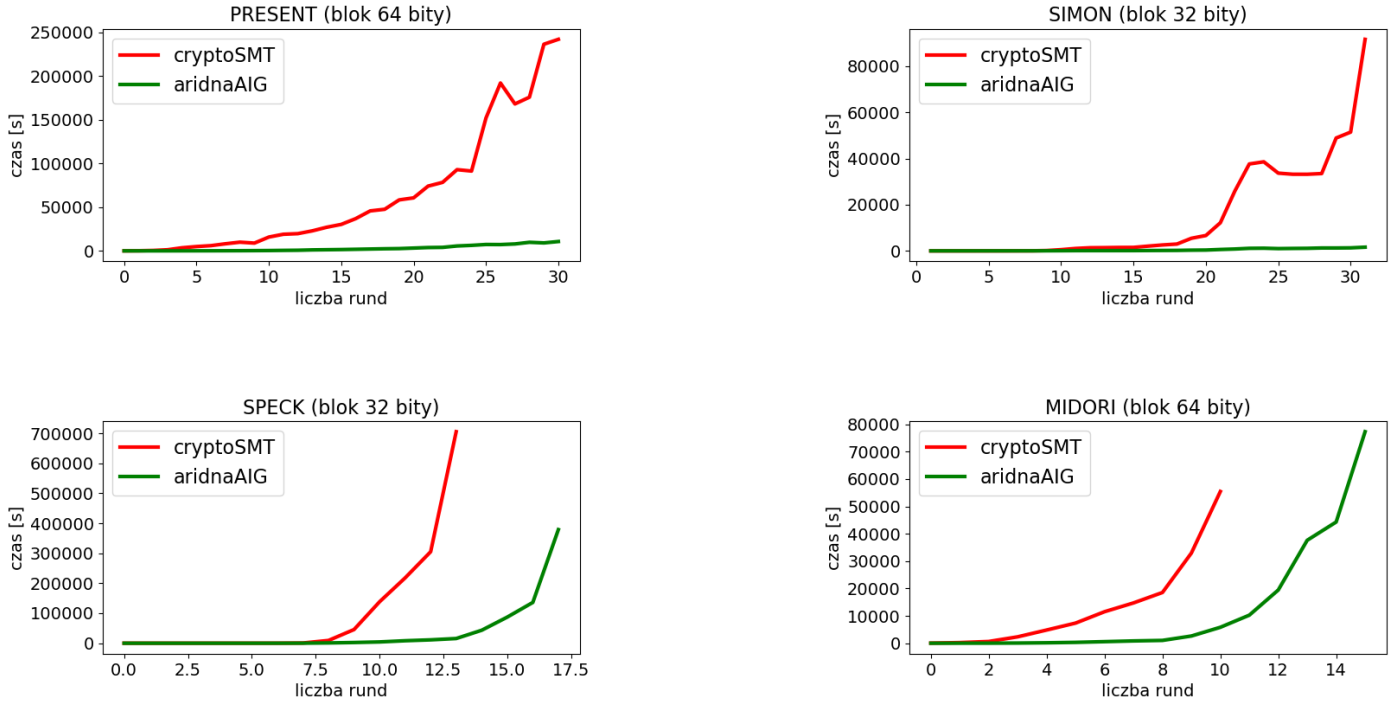


Rysunek 6.6: Średni rozmiar klauzuli `CNF` w modelach `SAT` generowanych na bazie grafów `AIG`, opisujących propagację charakterystyki różnicowej dla pierwszych 10 rund dla każdego z badanych szyfrów.

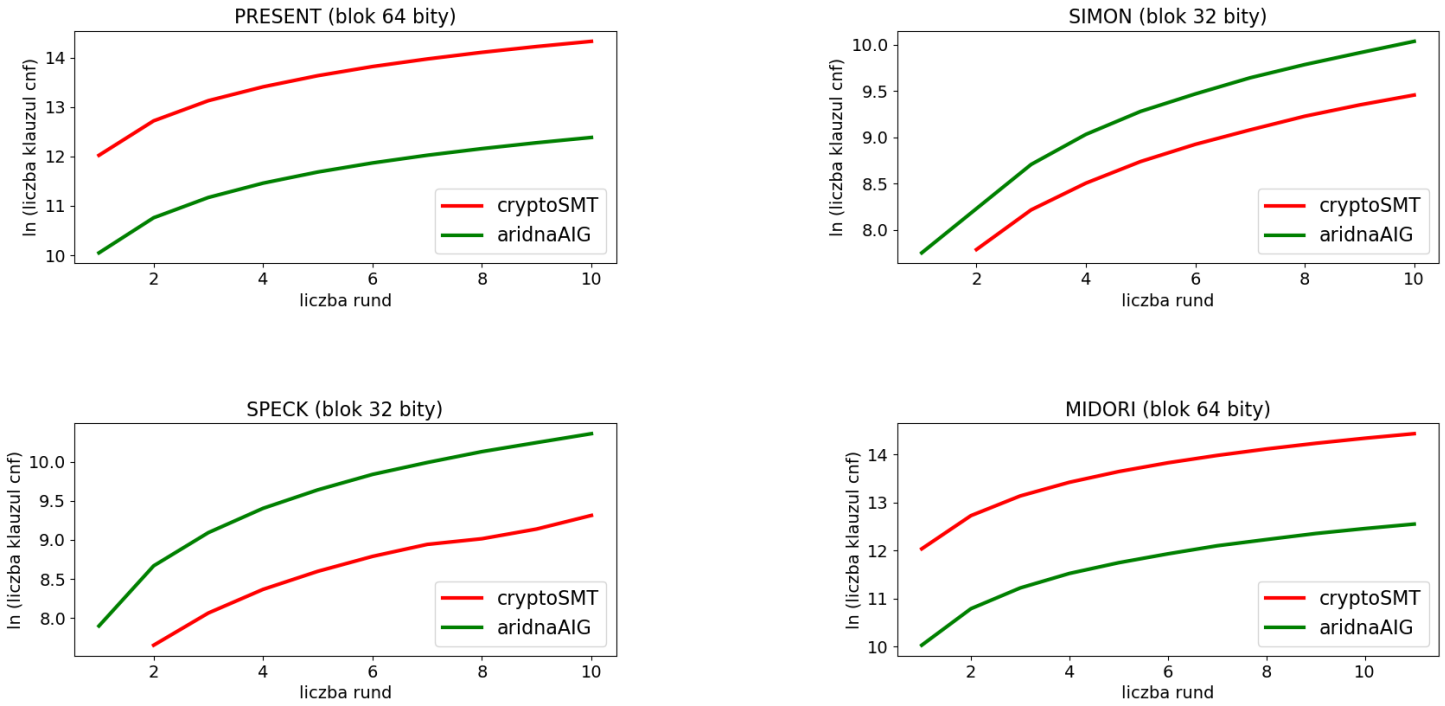
równano czas poszukiwania optymalnej charakterystyki różnicowej dla szyfrów `PRESENT`, `SIMON`, `SPECK` oraz `MIDORI` za pomocą narzędzia wykonanego w ramach mojej implementacji oraz `CryptoSMT` dostępnego pod adresem `https://github.com/kste/cryptosmt`. Porównanie wykonano na tym samym sprzęcie komputerowym[3] przy wykorzystaniu jednego wątku. Podczas badania zakładano, że nie będą wykorzystywane żadne założenia wynikające z konstrukcji szyfru blokowego dotyczące granicy od jakiej szukamy optymalnego prawdopodobieństwa[4]. Takie same założenia wykorzystane zostały w `CryptoSMT`. Do rozwiązywania modelu `SAT` wykorzystano jednowątkowy *SAT solver* `Cadical` [10]. Wykonane narzędzie okazało się na tyle wydajne, że udało się określić optymalną 5 rundową charakterystykę różnicową dla algorytmu `AES` (na chwilę obecną nieznane są publikacje, w którym zostałaby ona wskazana). Udało się rownież potwierdzić, że nie istnieje charakterystyka różnicowa na 6 rund algorytmu, której prawdopodobieństwo byłoby równe lub wyższe niż 2^{-189}.

[3]Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz

[4]Podczas pierwszej iteracji zakładano, że prawdopodobieństwo optymalnej charakterystyki różnicowej wynosi 2^0, niezależnie od aktualnie badanej rundy



Rysunek 6.7: Wydajność narzędzia do poszukiwania optymalnych charakterystyk różnicowych opartego na zaproponowanej metodzie wykorzystującej grafy `AIG` i model `SAT` względem narzędzia `CryptoSMT` [61].

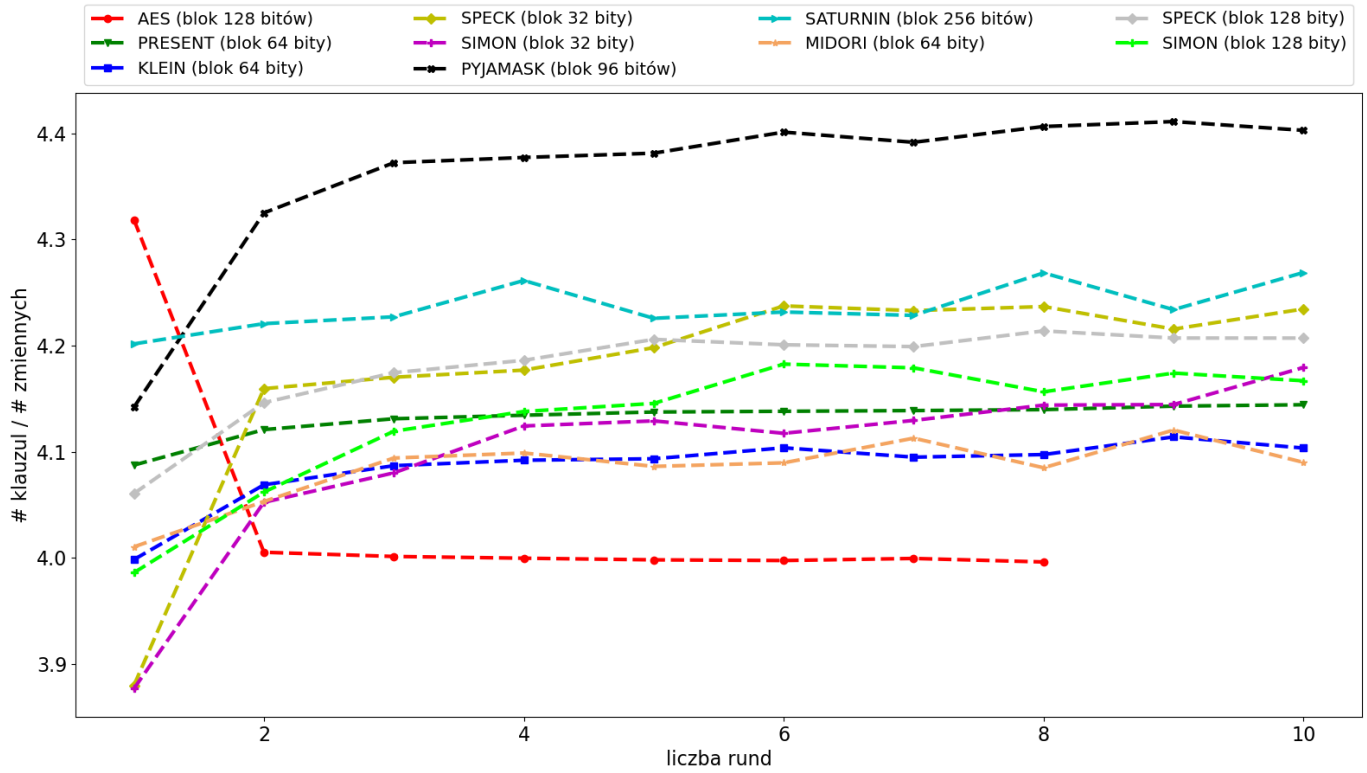


Rysunek 6.8: Liczba klauzul w modelach `SAT` uzyskiwanych za pomocą zaproponowanej metody wykorzystującej grafy `AIG` i odpowiadających im modelach `SAT` generowanych za pomocą narzędzia `CryptoSMT` [61] (skala logarytmiczna).

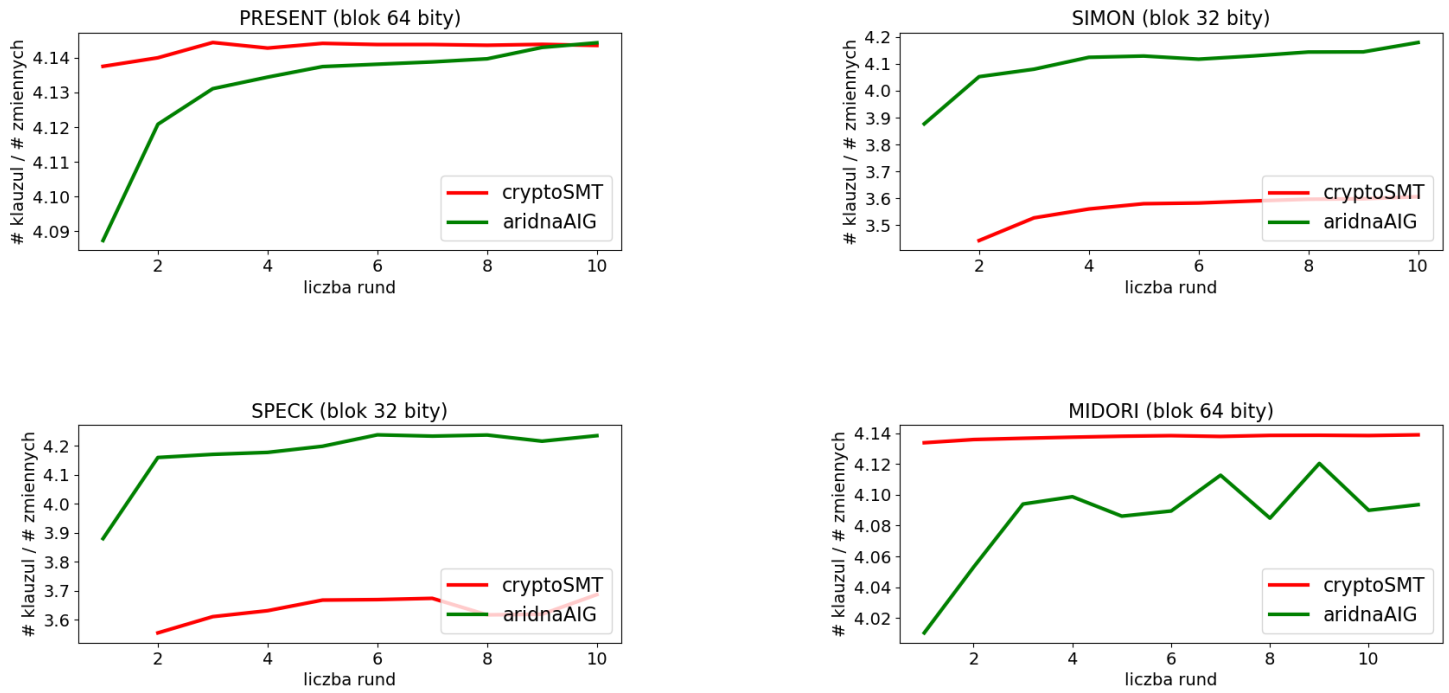
Czas potrzebny do określenia optymalnej charakterystyki różnicowej szyfru blokowego jest zdecydowanie mniejszy w przypadku modeli `SAT` (reprezentowanych w postaci zbioru klauzul *CNF*) generowanych za pomocą grafów `AIG`. W przypadku szyfrów blokowych, w których wykorzystywana jest skrzynka podstawieniowa, modele te są również zdecydowanie mniejsze pod kątem liczby klauzul i liczby zmiennych niż w przypadku analogicznych modeli generowanych w `CryptoSMT` (dla szyfru `MIDORI` oraz `PRESENT` model dla każdej z rund okazał się około 7-krotnie mniejszy). Dla szyfrów `SIMON` oraz `SPECK` model był odpowiednio około 2-krotnie oraz 3-krotnie większy niż analogiczny model generowany poprzez oprogramowanie `CryptoSMT` (rysunek nr 6.8). Maksymalna długość klauzuli w zbiorze klauzul *CNF* wygenerowanych za pośrednictwem grafu `AIG` wynosi 4, zaś w przypadku `CryptoSMT` równa jest 5. Może to być jeden z czynników wpływających na czas wymagany na określenie wartościowania modelu `SAT`. Drugim czynnikiem, który zgodnie z [8] może mieć niebagatelny wpływ na złożoność obliczeniową jest „*gęstość*" modelu `SAT` wyrażana poprzez wskaźnik ρ zdefiniowany następująco:

$$\rho = \frac{\text{liczba klauzul}}{\text{liczba zmiennych}}$$

Na rysunku nr 6.9 przedstawiono wartość wskaźnika ρ dla modeli `SAT` generowanych poprzez grafy `AIG`, opisujących propagację charakterystyki różnicowej dla pierwszych 10 rund w badanych szyfrach blokowych. Na rysunku nr 6.10 porównano natomiast wskaźnik ρ dla modeli `SAT` uzyskiwanych w zaproponowanej metodzie wykorzystującej grafy `AIG` i odpowiadających im modeli `SAT` generowanych za pomocą narzędzia `CryptoSMT`. Warto zaznaczyć również, że maksymalna długość klauzuli *CNF* w modelach `SAT` generowanych za pomocą zaprezentowanej w niniejszej rozprawie doktorskiej metody wynosi 4. W analogicznych modelach generowanych za pomocą narzędzia `CryptoSMT` najdłuższe klauzule składały się z 5 czynników.



Rysunek 6.9: Wartość wskaźnika ρ dla modeli `SAT` generowanych poprzez grafy `AIG`, opisujących propagację charakterystyki różnicowej dla pierwszych 10 rund (maksymalna wartość prawdopodobieństwa).



Rysunek 6.10: Wartość wskaźnika ρ dla modeli `SAT` uzyskiwanych w zaproponowanej metodzie wykorzystującej grafy `AIG` i odpowiadających im modeli `SAT` generowanych za pomocą narzędzia `CryptoSMT` [61].

Widać wyraźnie, że szyfry blokowe, dla których wyznaczenie optymalnej charakterystyki różnicowej wymaga intensywanego wysiłku obliczeniowego mają najwyższy lub najniższy wskaźnik ρ. Należy pamiętać również, że w przypadku szyfru `AES` model opisujący propagację różnicową jest ponad 100-krotnie większy niż w przypadku szyfrów `PRESENT`, `KLEIN`, `MIDORI`, `PYJAMASK` oraz około 1000-krotnie większy niż w przypadku szyfrów `SIMON` oraz `SPECK`. Fakt ten z pewnością nie wpływa pozytywnie na złożoność obliczeniową.

Przeprowadzone badania nasuwają hipotezę, że rozmiar problemu `SAT` nie determinuje jednoznacznie czasu wymaganego na rozwiązanie zadania. Złożoność obliczeniowa zależy przynajmniej od kilku czynników. Podczas testów brano pod uwagę m.in. rozmiar i gęstość modelu `SAT` oraz średni rozmiar klauzuli `CNF`.

Strona celowo pozostawiona pusta.

Rozdział 7

Podsumowanie i kierunki dalszych badań

Przedłożona rozprawa doktorska w głównej mierze dotyczy automatyzacji metod badania podatności szyfrów na najważniejsze znane metody kryptoanalizy. W jej ramach skupiono się przede wszystkim na kryptoanalizie różnicowej szyfrów blokowych, jednak przedstawione podejście można wykorzystać również do automatyzacji procesów związanych z kryptoanalizą liniową i algebraiczną. Przedstawiony sposób automatyzacji procesów związanych z analizą bezpieczeństwa szyfrów blokowych pod kątem badania podatności na ataki opierające się na założeniach kryptoanalizy różnicowej wykorzystuje opis propagacji charakterystyki różnicowej reprezentowany w postaci grafu `AIG`. Opis ten można uzyskać w sposób automatyczny na bazie implementacji w odpowiednich językach programowania. W badaniach wykorzystywano implementacje w języku funkcyjnym *Cryptol* [34]. Aby dostosować opisane metody do założeń kryptoanalizy liniowej, należy w analogiczny sposób opisać propagację charakterystyki liniowej przy wykorzystaniu profili liniowych odpowiednich dla operacji wykorzystywanych w analizowanym algorytmie.

W przypadku kryptoanalizy algebraicznej opis funkcji szyfrowania za pomocą grafu `AIG` może posłużyć do wygenerowania układu równań nad ciałem binarnym. Taki układ równań można próbować rozwiązać przy znanym tekście jawnym i odpowiadającym mu szyfrogramie lub kilku tekstach jawnych i odpowiadającym im szyfrogramom. Podejście to zostało zaprezentowane w autorskim artykule [2], w którym porównano czas, w jakim odzyskano zestaw podkluczy rundowych przy wykorzystaniu modeli `SAT`, generowanych na podstawie grafów `AIG` budowanych na bazie implementacji szyfru za pomocą różnych języków programowania (`Cryptol`, `C` oraz `VHDL`).

Grafy `AIG` można wykorzystać również do analizy szyfrów strumieniowych. W autorskim artykule [30] przedstawiono metodę, gdzie za ich pomocą generowano modele `SAT` umożliwiające poszukiwanie krótkich cykli w algorytmach takich jak `Trivium` oraz `Bivium`. Opracowana metoda jest jednak na tyle ogólna, że można ją zastosować również w innych szyfrach strumieniowych.

Podsumowując badania przeprowadzone w ramach niniejszej rozprawy doktorskiej, do wymiernych sukcesów można zaliczyć przede wszystkim:

- opracowanie automatycznej metody wyznaczania charakterystyk różnicowych opartej na grafach `AIG`, aktywujących minimalną liczbę skrzynek podstawieniowych w szyfrach blokowych [rozdz. 5.2.2];
- opracowanie automatycznej metody wyznaczania optymalnych charakterystyk różnicowych opartej na grafach `AIG`, której bardzo dobrą wydajność potwierdziły testy przeprowadzone na popularnych obecnie szyfrach blokowych takich jak `PRESENT`, `MIDORI`, `KLEIN`, `SPECK`, `SIMON` oraz `AES` [rozdz. 6.3.1];
- opracowanie automatycznej metody poszukiwania obciętych charakterystyk różnicowych opartej na grafach `AIG` [rozdz. 6.3.2];
- w ramach przeprowadzonych obliczeń udało się wyznaczyć nieznane dotąd w literaturze optymalne charakterystyki różnicowe dla zredukowanych co do liczby rund szyfrów blokowych `AES`, `SPECK`, `SATURNIN` oraz `PYJAMASK`;
- w ramach przeprowadzonych obliczeń udało się wyznaczyć nieznane dotąd w literaturze charakterystyki różnicowe obcięte dla zredukowanych co do liczby rund szyfrów blokowych `AES`, `MIDORI`, `KLEIN` oraz `PRESENT`;
- przeprowadzenie badań i doświadczeń, których wyniki empiryczne świadczą o tym, że wykorzystanie grafów `AIG` bardzo dobrze nadaje się do automatyzacji procesów związanych z kryptoanalizą i analizą bezpieczeństwa algorytmów kryptograficznych;
- przeprowadzenie badań i doświadczeń, które potwierdzają użyteczność *SMT* i *SAT solverow* w procesach związanych z kryptoanalizą i oceną bezpieczeństwa algorytmów kryptograficznych;
- zaimplementowanie narzędzi do poszukiwania charakterystyk różnicowych (optymalnych pod kątem wartości prawdopodobieństwa oraz minimalnej liczby aktywnych skrzynek podstawieniowych) i charakterystyk różnicowych obciętych (wyniki uzyskane przy wykorzystaniu tego oprogramowania dostępne są w publicznym repozytorium udostępnionym pod adresem `https://gitlab.com/Witek1809/Ariadna-Results`).

Ciekawym spostrzeżeniem wynikającym z przeprowadzonych obliczeń jest fakt, że wydajność opracowanych metod poszukiwania optymalnych charakterystyk różnicowych lub charakterystyk różnicowych aktywujących minimalną liczbę skrzynek jest zdecydowanie gorsza w przypadku szyfrów blokowych zorientowanych stricte bitowo (np. `PYJAMASK`, `SATURNIN` czy też `SPECK`). Po sukcesie algorytmu `Rijndael` większość projektantów wykorzystywała operacje zorientowane na przetwarzanie kilkubitowych słów (najczęściej bajtów lub znaków heksadecymalnych). Obecnie widać wyraźnie, że trendy konstrukcyjne odwracją się właśnie w kierunku schematów kładących nacisk na ściśle bitowe zorientowanie warstwy liniowej i odpowiednie wplecenie jej w warstwę skrzynek podstawieniowych. Coraz częściej wykorzystuje się również dodawanie arytmetyczne jako operację nieliniową.

Potwierdza to konkurs zorganizowany przez NIST[1] w celu wyłonienia standardów kryptografii lekkiej (ang. *Lightweight Cryptography*). Oczywiście większą niż zwykle liczbę takich konstrukcji można tłumaczyć również charakterem tego konkursu i nacisku na wydajność algorytmów cechujących się również niewielkim zapotrzebowaniem na zasoby sprzętowe wymagane do wdrożenia szyfru. Nie zmienia to jednak faktu, że oparcie konstrukcji szyfru blokowego na stricte bitowo zorientowanej warstwie liniowej może wydawać się nieco archaiczne, biorąc pod uwagę architekturę współczesnych procesorów i wszechobecne wsparcie sprzętowe dla instrukcji związanych z szyfrem AES (np. zestaw instrukcji AES-NI). Warto jednak podkreślić, że zarówno wyniki uzyskane w ramach przeprowadzonych przeze mnie badań oraz udostępnione w innych źródłach [61] czy [62] wskazują, że umiejętne ściśle bitowe zorientowanie szyfru blokowego utrudnia dokładne określenie optymalnej charakterystyki różnicowej, a przede wszystkim charakterystyki różnicowej obciętej. Przykładem ściśle bitowo zorientowanych algorytmów blokowych, w których osiągnięto taki efekt są bez wątpienia szyfry PYJAMASK oraz SATURNIN. Przykładem konstrukcji, w której ściśle bitowo zorientowana warstwa liniowa nie utrudnia poszukiwania optymalnych charakterystyk różnicowej i charakterystyk różnicowych obciętych jest natomiast szyfr PRESENT. Bierze się to stąd, że w algorytmie PRESENT wykorzystano prostą permutację bitową, dla której parametr $\mathbb{D}_{bn}$ wynosi 2. Natomiast w algorytmie PYJAMASK wykorzystywane są cztery macierze MDS, gdzie dla każdej z nich parametr $\mathbb{D}_{bn}$ wynosi 12.

Ważnym etapem w przedstawionych metodach jest minimalizacja grafu AIG opisującego propagację charakterystyki różnicowej oraz charakterystyki różnicowej obciętej. Zasadne jest zatem, aby zwrócić szczególną uwagę na rozwój algorytmów pozwalających redukować graf przy jednoczesnym zachowaniu jego funkcjonalości. Dalszy rozwój algorytmów związanych z minimalizacją funkcji boolowskiej może skutkować skróceniem czasu, jaki potrzebny jest na ustalenie wartościowania modeli SAT generowanych w sposób przedstawiony w rozdziałach 5.2.2, 6.3.1 oraz 6.3.2. W tym obszarze warto zwrócić szczególną uwagę na metody redukcji funkcjonalnej grafu AIG oparte na tzw. relacjach boolowskiech (ang. *Boolean Relation*, BR). Optymalizacja logiki grafu z uwzględnieniem BR może wykorzystywać potencjalną elastyczność sieci w celu redukcji liczby wierzchołków grafu [46].

Ostatecznie najlepsze wyniki uzyskano przy podejściu, w którym propagację charakterystyki różnicowej przedstawiano w postaci grafu AIG, który następnie konwertowano do układu równań w postaci ANF. Na potrzeby wyznaczenia rozwiązania układ równań konwertowano do problemu SAT reprezentowanego przez zbiór klauzul w postaci CNF. Do rozwiązania problemu SAT najczęściej wykorzystywano takie *SAT solvery* jak *cadical* oraz *plingeling*. Oczywistym jest fakt, że sam sposób konwersji może wpływać na własności modelu opisującego problem SAT, a tym samym na czas jego rozwiązania. Nie było to jednak przedmiotem badań, gdyż głównym celem było zautomatyzowanie samego procesu opisu własności różnicowych szyfru blokowego, tak aby wydajność poszukiwania charakterystyk różnicowych była nie gorsza niż w przypadku opisu wykonanego odręcznie przez kryptoanalityka.

[1]Konkurs ogłoszony został w 2015 roku i trwa do chwili obecnej, a jego przebieg można śledzić pod adresem `https://csrc.nist.gov/projects/lightweight-cryptography`

Perspektywicznym kierunkiem rozwoju przedstawionego podejścia wydaje się konwersja układu równań w postaci `ANF` do problemu `QUBO`, a następnie wykorzystanie komputera kwantowego do wyznaczenia rozwiązania tak postawionego zadania. Obiecujące wyniki w kontekście tak realizowanych ataków algebraicznych prezentowano w [19].

Dobra wydajność opracowanych metod poszukiwania charakterystyk różnicowych i charakterystyk różnicowych obciętych zachęca do analizy innych szyfrów blokowych za pomocą metod zaproponowanych w ramach przedłożonej rozprawy doktorskiej. Szczególnie efektywna może być analiza lekkich algorytmów, które wykorzystują prymitywy kryptograficzne działające na stosunkowe niewielkich słowach, np. czterobitowe skrzynki podstawieniowe. W przyszłości należy również rozważyć dostosowanie metod do poszukiwania charakterystyk różnicowych niemożliwych. Kolejnym przedsięwzięciem będzie zapewne dostosowanie opracowanych metod do poszukiwania charakterystyk różnicowych dla funkcji skrótu (zabieg ten powinien być stosunkowo prosty), a także dla szyfrów strumieniowych.

Bibliografia

[1] Martin Albrecht. Algorithmic Algebraic Techniques and their Application to Block Cipher Cryptanalysis. 4:120–141, 04 2011.

[2] M Andrzejczak and W Dudzic. SAT Attacks on ARX Ciphers with Automated Equations Generation. *Infocommunications*, 2019.

[3] K Auguste. La cryptographie militaire. *Journal des sciences militaires*, 1883.

[4] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. Analysis of NORX: Investigating Differential and Rotational Properties. In Diego F. Aranha and Alfred Menezes, editors, *Progress in Cryptology - LATINCRYPT 2014*, pages 306–324, Cham, 2015. Springer International Publishing.

[5] S Banik, A Bogdanov, T Isobe, K Shibutani, H Hiwatari, and T Akishita. Midori: A Block Cipher for Low Energy. *International Conference on the Theory and Application of Cryptology and Information Security*, 2015.

[6] Gregory V. Bard. Algebraic Cryptanalysis.

[7] Daniel J. Bernstein. Cryptographic competitions. *IACR Cryptol. ePrint Arch.*, 2020.

[8] A Biere, M Heule, and H van Maaren. *Handbook of Satisfiability*. books.google.com, 2009.

[9] Armin Biere. *CADICAL, LINGELING, PLINGELING, TREENGELING and YALSAT Entering the SAT Competition 2017*. 2017.

[10] Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximillian Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In Tomas Balyo, Nils Froleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors, *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, pages 51–53. University of Helsinki, 2020.

[11] E Biham, A Biryukov, and A Shamir. *Cryptanalysis of Skipjack Reduced to 31 Rounds using Impossible Differentials, Technion*. CS Dept, Tech Report CS0947, 1998.

[12] E Biham, A Biryukov, and A Shamir. Miss in the Middle Attacks on IDEA and Khufu. *International Conference on Fast Software Encryption*, 1999.

[13] Eli Biham and Adi Shamir. Differential Cryptanalysis of Feal and N-Hash. In Donald W. Davies, editor, *Advances in Cryptology — EUROCRYPT '91*, pages 1–16, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[14] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, Berlin, Heidelberg, 1993.

[15] Alex Biryukov and Ivica Nikolić. Search for Related-Key Differential Characteristics in DES-Like Ciphers. In Antoine Joux, editor, *Fast Software Encryption*, pages 18–34, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[16] Alex Biryukov, Arnab Roy, and Vesselin Velichkov. Differential Analysis of Block Ciphers SIMON and SPECK. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption*, pages 546–570, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[17] J Borst, LR Knudsen, and V Rijmen. Two attacks on reduced IDEA. *International Conference on the Theory and Application of Cryptology and Information Security*, 1997.

[18] R. Brayton and A. Mishchenko. ABC: An Academic Industrial-Strength Verification Tool. *International Conference on Computer Aided Verification*, pages 24–40, 2010.

[19] E Burek, M Wroński, and M Misztal. Algebraic attacks on block ciphers using quantum annealing. *eprint.iacr.org*, 2021.

[20] Anne Canteaut. *Linear Cryptanalysis for Stream Ciphers*, pages 725–726. Springer US, Boston, MA, 2011.

[21] K. Carter, A. Foltzer, J. Hendrix, B. Huffman, and A. Tomb. SAW: the software analysis workbench. *ACM SIGAda Ada Letters*, pages 15–18, 2013.

[22] Carlos Cid and Ralf-Philipp Weinmann. Block Ciphers: Algebraic Cryptanalysis and Grobner Bases. In Massimiliano Sala, Shojiro Sakata, Teo Mora, Carlo Traverso, and Ludovic Perret, editors, *Grobner Bases, Coding, and Cryptography*, pages 307–327. Springer Berlin Heidelberg, 2009.

[23] S. Cook. The complexity of theorem proving procedures. *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, 1971.

[24] N Courtois and J Pieprzyk. International Conference on the Theory and Application of Cryptology and Information Security. *International Conference on the Theory and Application of Cryptology and Information Security*, 2002.

[25] N. T. Courtois and G. V. Bard. Algebraic cryptanalysis of thedata encryption standard. page 152–169, 2007.

[26] Joan Daemen and Vincent Rijmen. The wide trail design strategy. In *In Proceedings of the 8th IMA International Conference on Cryptography and Coding (IMA '01*, pages 222–238, 2001.

[27] M. Davis and H. Putnam. A Computing Procedere for Quantification Theory. *Journal of the ACM*, 1960.

[28] Xiang-zhong Dong and Jie Guan. Differential distribution properties of the SIMON block cipher family. In *2016 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 1–6, 2016.

[29] W Dudzic. Kryptoanaliza algebraiczna rodziny szyfrów blokowych Speck. *Przegląd Telekomunikacyjny - Wiadomości Telekomunikacyjne*, 2018.

[30] W Dudzic and K Kanciak. Using SAT solvers to finding short cycles in cryptographic algorithms. *International Journal of Electronics and Communications*, 2020.

[31] M. Rogawski E. Homsirikamol, P. Morawiecki and M. Srebrny. Security Margin Evaluation of SHA-3 Contest Finalists through SAT-Based Attacks. *Computer Information Systems and Industrial Management*, page 56–67, 2012.

[32] Eugene C. Freuder and Alan K. Mackworth. Constraint Satisfaction: An Emerging Paradigm. In P. van Beek F. Rossi and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 2. Elsevier, 2006.

[33] Kai Fu, Meiqin Wang, Yinghua Guo, Siwei Sun, and L. Hu. MILP-Based Automatic Search Algorithms for Differential and Linear Trails for Speck. In *FSE*, 2016.

[34] Inc Galois. Cryptol The Language of Cryptography, 2018.

[35] D Goudarzi, J Jean, S Kölbl, T Peyrin, M Rivain, Y. Sasaki, and S Meng Sim. Pyjamask: Block cipher and authenticated encryption with highly efficient masked implementation. *IACR Transactions on Symmetric Cryptology*, 2020.

[36] S. Hassoun and S. Tsutomu. Logic Synthesis and Verification. 2002.

[37] T Jakobsen. A fast method for cryptanalysis of substitution ciphers. *Cryptologia*, 1995.

[38] Jérémy Jean. TikZ for Cryptographers. `https://www.iacr.org/authors/tikz/`, 2016.

[39] Lars R Knudsen. Truncated and higher order differentials. *International Workshop on Fast Software Encryption*, 1994.

[40] Lars R Knudsen. DEAL-a 128-bit block cipher. *complexity*, 1998.

[41] Lars R. Knudsen and Matthew J. B. Robshaw. *The Block Cipher Companion*. Springer Publishing Company, Incorporated, 2011.

[42] LR Knudsen and TA Berson. Truncated differentials of SAFER. *International Workshop on Fast Software Encryption*, 1996. Query date: 2022-03-23 19:17:31.

[43] LR Knudsen, MJB Robshaw, and D Wagner. Truncated differentials and Skipjack. *Annual International Cryptology Conference*, 1999.

[44] Stefan Kölbl, Gregor Leander, and Tyge Tiessen. Observations on the SIMON Block Cipher Family. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 161–185, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[45] Z Kotulski. Budowanie szyfrów blokowych: nowe możliwości. Citeseer, 2003.

[46] Tung-Yuan Lee, Chia-Cheng Wu, Chia-Chun Lin, Yung-Chih Chen, and Chun-Yao Wang. Logic optimization with considering boolean relations. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 761–766, 2018.

[47] Y. Li, A. Albarghouthi, Z. Kincaid, A. Gurfinkel, and M. Chechik. Symbolic optimization with SMT solvers. *ACM SIGPLAN*, 2014.

[48] H Lipmaa and S Moriai. Efficient algorithms for computing differential properties of addition. *International Workshop on Fast Software Encryption*, 2001.

[49] D. Loveland M. Davis, G. Logemann. A Machine Program for Theorem Proving. *Communications of the ACM*, 1962.

[50] M Matsui. Linear cryptanalysis method for DES cipher. *Workshop on the Theory and Application of of Cryptographic Techniques*, 1993.

[51] Mitsuru Matsui. On correlation between the order of S-boxes and the strength of DES. In Alfredo De Santis, editor, *Advances in Cryptology — EUROCRYPT'94*, pages 366–375, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[52] Ilya Mironov and Lintao Zhang. Applications of SAT Solvers to Cryptanalysis of Hash Functions. *IACR Cryptology ePrint Archive*, 2006:102–115, 01 2006.

[53] A. Mishchenko, R. Jiang, S. Chatterjee, and R. Brayton. FRAIGs: Functionally reduced AND-INV graphs. *International Conference on Computer Aided Verification*, 2004.

[54] Nicky Mouha and Bart Preneel. Towards Finding Optimal Differential Characteristics for ARX: Application to Salsa20, 2013. `https://eprint.iacr.org/2013/328`.

[55] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In Chuan-Kun Wu, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology*, pages 57–76, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[56] L. De Moura and N. Bjørner. Z3: An efficient SMT solver. *International conference on Tools and Algorithms*, 2008.

[57] R Posteuca and T Ashur. How to Backdoor a Cipher. *IACR Cryptol. ePrint Arch.*, 2021.

[58] Sumanta Sarkar and Habeeb Syed. Bounds on Differential and Linear Branch Number of Permutations. In Willy Susilo and Guomin Yang, editors, *Information Security and Privacy*, pages 207–224, Cham, 2018. Springer International Publishing.

[59] A Shamir. Impossible differential attacks. *CRYPTO 1998*, 1998.

[60] C. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal, Vol 28, pp. 656–715*, Oktober 1949.

[61] Stefan Kölbl. CryptoSMT: An easy to use tool for cryptanalysis of symmetric primitives. `https://github.com/kste/cryptosmt`.

[62] L Sun, W Wang, and M Wang. Accelerating the Search of Differential and Linear Characteristics with the SAT Method. *IACR Transactions on Symmetric Cryptology*, 2021.

[63] S Sun, L Hu, P Wang, K Qiao, X Ma, and L Song. Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block cipher. *Advances in Cryptology - ASIACRYPT 2014 -20th International Conference on the Theory and Application of Cryptologyand Information Security*, 2014.

[64] Siwei Sun, Lei Hu, Meiqin Wang, Peng Wang, Kexin Qiao, Xiaoshuang Ma, Danping Shi, Ling Song, and Kai Fu. Towards Finding the Best Characteristics of Some Bit-oriented Block Ciphers and Automatic Enumeration of (Related-key) Differential and Linear Characteristics with Predefined Properties. *IACR Cryptology ePrint Archive*, 2014. `https://eprint.iacr.org/2014/747`.

[65] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 158–178, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[66] Shengbao Wu and Mingsheng Wang. Security Evaluation against Differential Cryptanalysis for Block Cipher Structures. IACR Cryptology ePrint Archive, 2011. https://eprint.iacr.org/2011/551.