UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Gorana Vučić

ALGORITMI ZA GENERISANJE MINIMALNIH BULOVIH FUNKCIJA

master rad

Beograd, 2022.

Mentor:

dr Miodrag Živković, redovni profesor
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Stefan Mišković, docent
Univerzitet u Beogradu, Matematički fakultet

dr Nina Radojičić Matić, docent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: septembar 2022.

Naslov master rada: Algoritmi za generisanje minimalnih Bulovih funkcija

Rezime: Važna oblast istraživanja u kriptografiji predstavljaju Bulove funkcije. One su važna komponenta brojnih kriptografskih sistema. Istraživanje ovih funkcija je od suštinskog značaja za obezbeđivanje bezbedne komunikacije. U samom radu akcenat je na istraživanju i pronalaženju minimalnih (eng. *bent*) Bulovih funkcija od parnog broja n promenljivih, odnosno funkcija koje postižu maksimalnu nelinearnost (rastojanje od skupa afinih Bulovih funkcija) $2^{n-1} - 2^{\frac{n}{2}-1}$. Od velikog značaja su za kriptografiju zbog otpornosti na korelacioni napad, odnosno linearnu kriptoanalizu. U okviru ovog rada implementiran je algoritam za generisanje minimalnih Bulovih funkcija iz tačke 5.2 disertacije [4] i prikazani su rezultati izvršavanja ovog algoritma za vrednosti n od 2 do 32.

Ključne reči: minimalne Bulove funkcije, algebarska normalna forma, Volš-Adamardova transformacija, nelinearnost

Sadržaj

Glava 1

Uvod

Minimalne Bulove funkcije, bent funkcije (eng. *bent functions*) imaju svojstvo, da se svaka od njih nalazi na maksimalnoj mogućoj Hemingovoj udaljenosti od skupa svih afinih Bulovih funkcija. Bent funkcije su posebno zanimljive zbog njihove sposobnosti da maksimizuju važno kriptografsko svojstvo, nelinearnost. Otkrivene su od strane kriptografa koji su tražili funkcije koje je teško aproksimirati linearnim ili afinim funkcijama. Ovo ih izdvaja kao posebnu klasu i dovodi do brojnih primena u kombinatorici, teoriji kodiranja i kriptografiji. Uvedene su od strane O. Rothausa, američkog matematičara 1960-ih godina. Konstrukcija ovakvih funkcija predstavlja težak zadatak, jer zbog izuzetno velikog broja Bulovih funkcija neizvodljivo je vršiti nasumičnu pretragu i stoga postoji potreba za efikasnim metodama konstrukcije ovakvih funkcija.

Prethodna istraživanja su često bila ograničena poteškoćama koje proizilaze iz veličine prostora za pretragu. Zato su bile neophodne neke nove tehnike kako bi se prevazišla ovakva ograničenja. U ovom radu je predstavljena tehnika za generisanje bent funkcija iz doktorske disertacije Dž. Fuler [4].

Kako bi se razumeo sam algoritam koji je predstavljen u glavi 3, neophodno je upoznati se sa nekim osnovnim reprezentacijama, svojstvima i transformacijama Bulovih funkcija.

U drugoj glavi opisane su osnovne reprezentacije Bulovih funkcija kao što su tablica istinitosti, polarna tablica istinitosti i algebarska normalna forma koja, se izvodi iz binarne tablice istinitosti. Opisana je Volš-Adamardova transformacija

(eng. *Walsh Hadamard Transform*), koja pruža još jedan vid reprezentacije Bulovih funkcija i jedinstvena je za svaku funkciju. Opisana su i najvažnija svojstva bent funkcija, nelinearnost i Hemingova težina podfunkcija. U odeljku 2.7 dat je pregled svih bitnih definicija i teorema koje važe za bent funkcije.

U trećoj glavi pojašnjena je ideja za opšti algoritam konstrukcije bent funkcije. Argumenti algoritma su n, paran broj ulaznih promenljivih i $maxDeg$, maksimalni stepen. Na izlazu se dobija niz bent funkcija stepena $\leq maxDeg$.

U četvrtoj glavi su prikazani detalji implementacije samog algoritma i svih bitnih funkcija za izvršavanje istog. Opisane su optimizovane konstrukcije algebarske normalne forme, Volš-Adamardove transformacije i određivanja Hemingovih težina podfunkcija.

U petoj glavi najpre se prikazuju rezultati dobijeni realizovanim algoritmom za sve kombinacije broja ulaznih promenljivih i maksimalnog stepena, $8 \leq n \leq 16$ i $3 \leq maxDeg \leq \frac{n}{2}$, kao u disertaciji Dž. Fuler [4], da bi se proverila ispravnost implementacije. Zatim se istražuje učinkovitost primenjenih optimizacija, tako što se širi domen pretrage bent funkcija na Bulove funkcije sa brojem ulaza $18 \leq n \leq 32$ i maksimalnim stepenom $3 \leq maxDeg \leq \frac{n}{2}$. Prikazani su eksperimentalni rezultati merenja brzine rada programa.

Glava 2

Reprezentacija i svojstva Bulovih funkcija

2.1 Uvod

Bulove funkcije mogu imati različite reprezentacije, gde svaka ima neku svoju funkciju u pogledu kriptografske analize. U ovom odeljku su predstavljeni osnovni oblici reprezentacije Bulovih funkcija kao što su tablica istinitosti, algebarska normalna forma (u daljem tekstu ANF (eng. *Algebraic normal form transformation*)), Volš-Adamardova transformacija (u daljem tekstu WHT (eng. *Walsh Hadamard Transform*)). Ove reprezentacije su od posebnog značaja za algoritam 3. Pored toga uvode se i kriptografske mere koje se odnose na predstavljene reprezentacije Bulovih funkcija. Definisan je odnos između tablice istinitosti i ANF, kao i tablice isitnitosti i WHT. Bulova funkcija je preslikavanje $f(x) : \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2$. Neka β_n predstavlja skup svih 2^{2^n} Bulovih funkcija od n promenljivih. Sve definicije kriptografskih svojstava i reprezentacija Bulovih funkcija preuzete su iz rada [4].

2.2 Tablica istinitosti

Tablica istinitosti, TT (eng. *trurh table*), kao osnovni prikaz Bulovih funkcija je binarna tabela koja predstavlja listu rezultata za svaku od 2^n mogućih vrednosti ulaza za tu Bulovu funkciju. Naziva se binarnom tablicom istinitosti jer su vrednosti funkcije elementi iz skupa $\{0, 1\}$. Umesto nje može se koristiti polarna tablica istinitosti PTT (eng. *polarity truth table*) čije su vrednosti funkcije

elementi iz skupa $\{-1, 1\}$. Polarna tablica istinitosti se može dobiti iz binarne tablice istinitosti koristeći formulu $\hat{f}(x) = 1 - 2f(x)$. Primer kako izgledaju binarna i polarna tablica istinitosti može se videti u tabeli 2.1.

x_1	x_2	x_3	$f(x)$	$\hat{f}(x)$
0	0	0	0	1
0	0	1	1	-1
0	1	0	0	1
0	1	1	1	-1
1	0	0	0	1
1	0	1	1	-1
1	1	0	1	-1
1	1	1	0	1

Tabela 2.1: Primer binarne i polarne tablice istinitosti.

Na osnovu tablice istinitosti mogu se definisati nekoliko važnih svojstava, među kojima su Hemingova težina i Hemingova udaljenost.

Definicija 2.2.1. *Hemingova težina Bulove funkcije je definisana kao broj jedinica u binarnoj istinitosnoj tablici ili suma elemenata u polarnoj istinitosnoj tablici. Za notaciju se koristi* $\mathrm{wt}(f)$ *za binarnu:*

$$\mathrm{wt}(f) = \sum_x f(x) = \frac{1}{2}(2^n - \sum_x \hat{f}(x))$$

dok je $\mathrm{wt}(\hat{f})$ *za polarnu:*

$$\mathrm{wt}(\hat{f}) = \sum_x \hat{f}(x)$$

Definicija 2.2.2. *Hemingova udaljenost između dve funkcije* $f \in \beta_n$ *i* $g \in \beta_n$ *definisana je kao broj pozicija u kojima se funkcije razlikuju u tablici isitnitosti i može se izraziti kao Hemingova težina XOR sume dve funkcije:*

$$\mathrm{dist}(f, g) = \mathrm{wt}(f \oplus g)$$

Definicija 2.2.3. *Korelacija između dve funkcije* $f \in \beta_n$ *i* $g \in \beta_n$ *je data narednom formulom*

$$c(f, g) = 1 - \frac{\mathrm{dist}(f, g)}{2^{n-1}}$$

Korelacija predstavlja racionalni broj u opsegu $[-1, 1]$. Iz definicije se vidi da se gornja granica 1 postiže kada je Hemingova udaljenost između dve funkcije jednaka nuli. Slično tome donja granica -1 postiže se kada je Hemingova udaljenost između dve funkcije jednaka 2^n. Korelacija predstavlja važan alat u analizi funkcija, posebno u odnosu na koncept neuravnoteženosti (eng. *imbalance*) za Bulovu funkciju.

Definicija 2.2.4. *Za funkciju se kaže da je uravnotežena kada je polovina vrednosti funkcije jednaka jedinici odnosno važi* $\mathrm{wt}(f) = 2^{n-1}$ *ili alternativno* $\mathrm{wt}(\hat{f}) = 0$.

Definicija 2.2.5. *Neuravnoteženost Bulove funkcije je definisana kao*

$$I(f) = |\,\mathrm{wt}(f) - 2^{n-1}| = 2^{n-1}|c(f(x), 0)|$$

gde 0 označava konstantnu, nula Bulovu funkciju.

Neuravnoteženost je definisana kao minimalna udaljenost od neke uravnotežene funkcije i stoga je direktno proporcionalna veličini korelacije sa konstantnom nulom Bulove funkcije. Dakle, kada je neuravnoteženost jednaka nuli, funkcija je uravnotežena. Uravnoteženost (eng. *balance*) je osnovni kriptografski kriterijum, jer neuravnotežena funkcija korelira sa konstantnom funkcijom.

2.3 Algebarska normalna forma

Algebarska normalna forma, ANF (*Algebraic Normal Form*), je jedan prikaz Bulove funkcije. ANF svaku Bulovu funkciju na jedinstven način predstavlja kao XOR sumu proizvoda promenljivih (monoma).

Definicija 2.3.1. *Algebarska normalna forma izražava Bulovu funkciju kao sumu (XOR) proizvoda (AND) podskupova ulaznih promenljivih, odnosno ako je* $S = 1, 2, ..., n$ *tada je:*

$$f(x_1, x_2, ..., x_n) = \bigoplus_{I \subseteq S} a_I \prod_{i \in I} x_i$$

Koeficijenti a_I uzimaju vrednosti iz skupa $\{0, 1\}$. Problem izračunavanja ANF zadate Bulove funkcije svodi se na izračunavanje ovih koeficijenata. Tabela 2.2 je primer računanja koeficijenata za $n = 3$. Redom se zamenjuju različite vrednosti za x, pa se na osnovu toga određuje jedan po jedan koeficijent ANF.

$x = (x_1, x_2, x_3)$	**formula za** $f(x)$	**rešenje za** a_I
000	a_0	$a_0 = f(000)$
100	$a_0 \oplus a_1$	$a_1 = a_0 \oplus f(100)$
010	$a_0 \oplus a_2$	$a_2 = a_0 \oplus f(010)$
001	$a_0 \oplus a_3$	$a_3 = a_0 \oplus f(001)$
110	$a_0 \oplus a_1 \oplus a_2 \oplus a_{12}$	$a_{12} = a_0 \oplus a_1 \oplus a_2 \oplus f(110)$
101	$a_0 \oplus a_1 \oplus a_3 \oplus a_{13}$	$a_{13} = a_0 \oplus a_1 \oplus a_3 \oplus f(101)$
011	$a_0 \oplus a_2 \oplus a_3 \oplus a_{23}$	$a_{23} = a_0 \oplus a_2 \oplus a_3 \oplus f(011)$
111	$a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus$ $a_{12} \oplus a_{13} \oplus a_{23} \oplus a_{123}$	$a_{123} = a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus$ $a_{12} \oplus a_{13} \oplus a_{23} \oplus f(111)$

Tabela 2.2: Izračunavanje ANF za $n = 3$.

Tabela 2.3 prikazuje izračunatu ANF funkcije $f(x)$. Indeksi I izračunatih koeficijenata a_I ANF su kodirani vektorom ulaznih promenljivih $x = (x_1, x_2, x_3)$, tako da je $x_i = 0$ ako ono ne ulazi u podskup (indeks) I, ili ulazi ako je $x_i = 1$. Na primer ako je koeficijent a_{13} uključen u ANF onda će on u istinitosnoj tablici stajati u redu gde su x_1 i x_3 ($f(101)$) jednako 1.

x_1	x_2	x_3	$f(x)$	ANF
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

Tabela 2.3: Bulova funkcija i njena ANF.

Niz vrednosti funkcije $f(x)$ i niz koeficijenata njene ANF su vektori iz $\mathbb{Z}_2^{2^n}$. Za proizvoljno n postoji transformacija iz $f(x)$ u njenu ANF zadatu binarnom matricom, dimenzija $2^n \cdot 2^n$, koja se naziva ANF transformacija (eng. *Algebraic normal form transformation*), u daljem tekstu ANFT i primenjuje se tako što se funkcija množi ovom binarnom matricom. ANFT je samoj sebi inverzna operacija i primenom ANFT na ANF Bulove funkcije dobija se njena binarna istinitosna tablica.

Definicija 2.3.2. *Kronekerov proizvod matrice A, dimenzija $p \cdot q$, sa matricom B, dimenzija $r \cdot s$, definiše se na sledeći način:*

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1q}B \\ \vdots & & \vdots \\ a_{p1}B & \dots & a_{pq}B \end{bmatrix} \quad [7]$$

Teorema 2.3.1. *Relacija između tablice istinitosti funkcije $f(x)$ i njene ANF je data u obliku $ANF_f = A_n \cdot f \mod 2$. A_n je matrica ANFT, veličine $2^n \cdot 2^n$, a definisana je rekurzivno korišćenjem Kronekerovog proizvoda matrica na sledeći način:*

$$A_n = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes A_{n-1} \text{ gde je } A_0 = [1].$$

Da bi se za tabelu zadatu tabelom 2.3 izračunala ANF, potrebno je konstruisati matricu A_3, tako da može da se odredi vrednost za $ANF_f = A_3 \cdot f \mod 2$, videti sliku 2.1.

$$A_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \mod 2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 2 \\ 0 \\ 2 \\ 1 \\ 4 \end{bmatrix} \mod 2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Slika 2.1: Izračuavanje ANF zadate tabelom 2.3.

Dve važne osobine koje su u relaciji sa ANF su algebarska težina (eng. *algebraic weight*) i stepen Bulove funkcije.

Definicija 2.3.3. *Algebarska težina ili* $\mathrm{awt}(f)$ *definiše se kao broj koeficijenata* a_I *u algebarskoj normalnoj formi funkcije f, čija je vrednost jednaka jedan.*

Definicija 2.3.4. *Stepen Bulove funkcije ili* $\mathrm{degree}(f)$ *se definiše kao ukupan broj promenljivih unutar terma najvećeg reda u ANF funkcije f.*

Stepen terma jednak je broju ulaznih promenljivih koji sačinjavaju taj term. Stepen Bulove funkcije predstavlja važno svojstvo jer pruža meru za složenost Bulove funkcije. Tako funkcije drugog stepena nazivamo kvadratnim funkcijama, dok funkcije stepena tri nazivamo kubnim funkcijama.

Definicija 2.3.5. *Linearna funkcija se definiše kao XOR suma podskupa ulaznih promeljivih, to jest funkcija oblika*

$$L_w(x) = w_1x_1 \oplus w_2x_2 \oplus ... \oplus w_nx_n$$

gde je $w = (w_1, w_2, ...w_n) \in \mathbb{Z}_2^n$.

Definicija 2.3.6. *Afina funkcija je linearna funkcija ili komplement linearne funkcije, to jest funkcija oblika*

$$A_{w,c}(x) = L_w(x) \oplus c$$

gde $c \in \{0, 1\}$.

2.4 Volš-Adamardova transformacija

Volš-Adamardova transformacija, WHT (eng. *Walsh Hadamard Transform*), predstavlja još jedan vid reprezentacije Bulovih funkcija. WHT izražava Bulovu funkciju u smislu njene korelacije sa skupom svih linearnih funkcija i jedinstvena je za svaku funkciju.

Definicija 2.4.1. *WHT Bulove funkcije $f(x)$ se označava sa $\hat{F}(w)$ i računa se na osnovu polarne tablice istinitosti na sledeći način $\hat{F}(w) = \sum_x \hat{f}(x)\hat{L}_w(x)$, $w \in \mathbb{Z}_2^n$, $\hat{L}_w(x) = 1 - 2L_w(x)$.*

Primer WHT vrednosti na osnovu polarne tablice istinitosti se može videti u tabeli 2.4.

x_1	x_2	x_3	$f(x)$	$\hat{f}(x)$	$\hat{L}_0$	$\hat{L}_1$	$\hat{L}_2$	$\hat{L}_3$	$\hat{L}_4$	$\hat{L}_5$	$\hat{L}_6$	$\hat{L}_7$	$\hat{F}$
0	0	0	0	1	0	1	1	1	1	1	1	1	0
0	0	1	1	-1	0	-1	1	-1	1	-1	1	-1	4
0	1	0	0	1	0	1	-1	-1	1	1	-1	-1	0
0	1	1	1	-1	0	-1	-1	1	1	-1	-1	1	4
1	0	0	0	1	0	1	1	1	-1	-1	-1	-1	0
1	0	1	1	-1	0	-1	1	-1	-1	1	-1	1	4
1	1	0	1	-1	0	1	-1	-1	-1	-1	1	1	0
1	1	1	0	1	0	-1	-1	1	-1	1	1	-1	-4

Tabela 2.4: Primer za WHT.

Iz definicije se vidi da važi $-2^n \leq \hat{F}(w) \leq 2^n$ za svako w. Svaka vrednost $\hat{F}(w)$ je direktno proporcionalno korelaciji osnovne funkcije $f(x)$ sa odgovarajućom linearnom funkcijom $L_w(x)$. Ako je $\hat{F}(0) = 0$, funkcija je uravnotežena. Maksimalna apsolutna vrednost WHT je u vezi sa važnom kriptografskom merom Bulove funkcije poznatoj kao nelinearnost, o čemu će biti reči u sledećem odeljku.

WHT pruža osnovu za definiciju spektra snage (eng. *power spectrum*).

Definicija 2.4.2. *Spektar snage Bulove funkcije se definiše kao kvadrat polarne transformacije WHT i obeležava se sa $P_{\hat{f}}(w)$, tako da važi $P_{\hat{f}}(w) = \hat{F}^2(w)$.*

Parselova jednačina $\sum_w (\hat{F}(w))^2 = 2^{2n}$ pokazuje da je suma kvadrata svih koeficijenata WHT konstanta. Ovo predstavlja važan rezultat, jer znači da svaka

promena Bulove funkcije koja smanjuje korelaciju sa nekom afinom funkcijom (nekim afinim funkcijama), dovedi do povećanja korelacije sa nekom drugom afinom funkcijom (nekim drugim afinim funkcijama) [5]. Iz ovog kompromisa sledi da je minimalna udaljenost do bilo koje afine funkcije ograničena.

2.5 Nelinearnost

Bulove funkcije koje se koriste u kriptografskim sistemima treba da poseduju svojstva takva da smanjuju efekte savremenih kriptoanalitičkih napada kao što je linearna kriptoanaliza. Kao fundamentalna tehnika postizanja sigurnosti u kriptografskim sistemima koristi se metoda iteriranja konfuzije i difuzije. Konfuzija se ogleda u nelinearnosti Bulovih funkcija. Nelinearnost predstavlja možda i najvažnije svojstvo prilikom određivanja ocenjivanja kriptografskih karakteristika Bulovih funkcija.

Postoji nekoliko kriterijuma za procenu nelinearnosti Bulovih funkcija, uključujući minimalnu udaljenost do bilo koje afine funkcije i stepen Bulove funkcije. Ovi kriterijumi analizirani su u radu [5]. Ustanovljeno je da je rasotojanje od afinih funkcija najrobusnija mera nelinearnosti: male promene u istinitosnoj tablici rezultiraju malim promenama ove udaljenosti. Stoga se minimalna udaljenost do bilo koje afine funkcije koristi za definisanje nelinearnosti, tako da važi da što je manja minimalna udaljenost do bilo koje afine funkcije to je veća nelinearnost.

Definicija 2.5.1. *Nelinearnost Bulovih funkcija definiše se kao minimalna Hemingova udaljenost od skupa afinih funkcija. Nelinearnost se dobija direktno posmatranjem $|\hat{F}_{max}|$, maksimalne apsolutne vrednosti koja se javlja u $\hat{F}_w$ i računa se na sledeći način: $N(f) = \frac{1}{2}(2^n - |\hat{F}_{max}|)$.*

Treba napomenuti da određivanje maksimalne nelinearnosti za uravnotežene Bulove funkcije sa parnim brojem ulaznih promenljivih većih od osam predstavlja otvoren problem. Otvoreni problem ostaje i računanje maksimalne nelinearnosti za Bulove funkcije sa neparnim brojem ulaznih promenljivih većim od sedam.

2.6 Hemingova težina podfunkcija

WHT se može koristiti za definisanje Hemingovih težina podfunkcija (eng. *Hamming weights of subfunctions*). Tablica istinitosti za Bulovu funkciju $f(x)$ od n promenljivih može se posmatrati kao konkatenacija dve podfunkcije f_0 i f_1 od $n-1$ promenljive. Ovo se može uopštiti na sledeći način korišćenjem pojma podele, gde je pravac podele definisan n-bitnim vektorom w i linearnom funkcijom $\langle w, x\rangle = L_w(x) = w_1x_1 \oplus w_2x_2 \oplus ... \oplus w_nx_n$.

Definicija 2.6.1. *Za Bulovu funkciju $f(x)$ se kaže da je podeljena duž pravca w na podfunkcije f_0 i f_1, čija se vrednost definiše na sledeći način:*

$$f_0 = \begin{cases} f(x) \text{ ako važi } \langle w, x\rangle = 0 \\ \text{nedefinisano u drugim slučajevima} \end{cases}$$

i

$$f_1 = \begin{cases} f(x) \text{ ako važi } \langle w, x\rangle = 1 \\ \text{nedefinisano u drugim slučajevima} \end{cases}$$

Podela funkcije f duž vektora w se obeležava sa $f = [f_0|f_1]_w$.

Kada je $w = e_1$, to odgovara konkatenaciji dve polovine tablice istinitosti funkcije. Ova definicija se može koristiti za ispitivanje Hemingove težine podfunkcija.

Sledeća teorema definiše Hemingovu težinu funkcije u odnosu na njene WHT vrednosti.

Teorema 2.6.1. *Hemingova težina bilo koje od w-podfunkcija bilo koje Bulove funkcije računa se preko WHT vrednosti na sledeći način:*

$$\mathrm{wt}(f_0, f_1) = \frac{2^n - \hat{F}(0) \pm \hat{F}(w)}{4}$$

Dokaz. Potrebno je izvršiti podelu Bulove funkcije duž nekog proizvoljnog w tako da važi $f(x) = [f_0|f_1]_w$. Jasno je da važi $\mathrm{wt}(f) = \mathrm{wt}(f_0) + \mathrm{wt}(f_1)$. Neka $\#\{\cdot\}$ predstavlja kardinalnost skupa, na osnovu WHT definicije:

$$\begin{aligned}\hat{F}(w) &= \sum_x \hat{f}(x)\hat{L}_w(x) \\ &= \#\{x : \hat{f}(x)\hat{L}_w(x) = 1\} - \#\{x : \hat{f}(x)\hat{L}_w(x) = -1\} \\ &= \#\{x : \hat{f}(x) \oplus \hat{L}_w(x) = 0\} - \#\{x : \hat{f}(x) \oplus \hat{L}_w(x) = 1\} \\ &= 2^n - 2 \times \mathrm{wt}(f \oplus L_w)\end{aligned}$$

iz čega sledi da

$$\mathrm{wt}(f \oplus L_w) = \frac{2^n - \hat{F}(w)}{2}.$$

Konkretno, za $w = 0$

$$\mathrm{wt}(f) = \frac{2^n - \hat{F}(0)}{2}.$$

Sada se razmatra

$$\begin{aligned}f(x) \oplus L_w(x) &= [f_0|f_1]_w \oplus [0|1]_w \\ &= [f_0|f_1 \oplus 1]_w.\end{aligned}$$

Stoga važi

$$\mathrm{wt}(f \oplus L_w) = wt(f_0) + (2^{n-1} - wt(f_1)).$$

Na osnovu prethodnog dobija se

$$\mathrm{wt}(f_0) + \mathrm{wt}(f_1) = \frac{2^n - \hat{F}(0)}{2}$$

i

$$2^{n-1} + \mathrm{wt}(f_0) - \mathrm{wt}(f_1) = \frac{2^n - \hat{F}(w)}{2}.$$

Kombinovanjem ova dva rezultata i preuređivanjem dobija se

$$\mathrm{wt}(f_0) = \frac{2^n - \hat{F}(0) - \hat{F}(w)}{4},$$

i

$$\mathrm{wt}(f_1) = \frac{2^n - \hat{F}(0) + \hat{F}(w)}{4}.$$

čime se dokazuje teorema. □

x_1	x_2	x_3	$f(x)$	$\hat{F}$	$\mathrm{wt}(f_0, f_1), w = (x_1, x_2, x_3)$
0	0	0	0	0	-
0	0	1	1	4	(1,3)
0	1	0	0	0	(2,2)
0	1	1	1	4	(1,3)
1	0	0	0	0	(2,2)
1	0	1	1	4	(1,3)
1	1	0	1	0	(2,2)
1	1	1	0	-4	(1,3)

Tabela 2.5: Primer za Hemingove težine podfunkcija, $n = 3$.

Na primer, ako je $w = (0, 1, 1)$ težine podfunkcija se računaju na sledeći način:

$$\mathrm{wt}(f_0) = \frac{2^n - \hat{F}(0) - \hat{F}(011)}{4} = \frac{8 - 0 - 4}{4} = 1$$

i

$$\mathrm{wt}(f_1) = \frac{2^n - \hat{F}(0) + \hat{F}(011)}{4} = \frac{8 - 0 + 4}{4} = 3$$

2.7 Bent funkcije

Bent funkcije su podskup Bulovih funkcija koje su od posebnog kriptografskog značaja. Iako su prvo prepoznate u kombinatornoj teoriji, tek su 1989. godine identifikovane kao ekvivalent optimalnoj klasi Bulovih funkcija koje zadovoljavaju niz kriptografski značajnih kriterijuma. One poseduju maksimalnu nelinearnost za funkcije sa parnim brojem ulaznih promenljivih. Iako nikada nisu uravnotežene, postaju statistički nerazlučive od uravnoteženih funkcija za dovoljno veliko n. CAST blok šifra predstavlja primer kriptografske primene bent funkcija. Sigurnost CAST je zasnovana na nekoliko 8x32 S-BOX, od kojih se svaki sastoji od 32 bent funkcije sa osam promenljivih.

Definicija 2.7.1. *Neka je $f(x)$ Bulova funkcija od n promenljivih, gde je n paran broj. Funkcija se naziva bent funkcijom ako postiže maksimalnu nelinearnost tako da važi $N(f_{bent}) = 2^{n-1} - 2^{\frac{n}{2}-1}$.*

Bent funkcije poseduju niz specifičnih i vrlo dobro poznatih kriptografskih svojstava.

Teorema 2.7.1. *Bent funkcija od n ulaza ima Hemingovu težinu jednaku $2^{n-1} \pm 2^{\frac{n}{2}-1}$. [8]*

Teorema 2.7.2. *Bent funkcija od n ulaza tako da je $n > 2$ ima stepen manji ili jednak $\frac{n}{2}$. [8]*

Teorema 2.7.3. *Za bent funkciju od n ulaza svi WHT koeficijenti su jednaki $\pm 2^{\frac{n}{2}-1}$. [4]*

Teorema 2.7.4. *Bent funkcija od n ulaza su na maksimalnoj mogućoj udaljenosti 2^{n-2} od bilo koje linearne funkcije.*

Zbog svojih jedinstvenih karakteristika bent funkcije su bile i danas su predmet velikog broja istraživanja. Rani doprinos istraživanju bent funkcija dali su Rothaus u svom radu [6], kao i Dillon u radu [3].

Glava 3

Konstrukcija bent funkcija

Pronalaženje bent funkcija predstavlja težak zadatak. Zbog izuzetno velikog broja Bulovih funkcija neizvodljivo je vršiti nasumična pretraživanja, stoga se javlja potreba za konstrukcijom efikasnijih tehnika. Metodologije koje se koriste su algebarske i heurističke.

Algebarske tehnike su korišćene za razvoj algoritama za konstrukciju kriptograski jakih Bulovih funkcija. Postoji nekoliko osnovnih pristupa algebarskom dizajnu Bulovih funkcija, od kojih je najčešća rekurzivna konstrukcija i njene brojne varijante, a u novije vreme i metoda linearne transformacije. Postoje mnogi specifični algoritmi za konstrukciju Bulovih funkcija koje poseduju određene nivoe kriptografski važnih svojstava uključujući nelinearnost, stepen, otpornost ili specifične kombinacije istih. Algebarske tehnike za konstrukciju obično su dizajnirane tako da postignu određenu osobinu. Međutim, u koncentrisanju na jedno svojstvo efekat drugih značajnih kriptografskih svojstava se često zanemaruje. [8]

Heurističke tehnike predstavljaju glavnu alternativu algebarskim tehnikama. One u proces uključuju iterativno poboljšavanje funkcije u odnosu na jedno ili više svojstava. Optimizacija funkcije u odnosu na više od jednog svojstva pruža mogućnost pronalaženja Bulovih funkcija koje su superiorinije od onih generisanih korišćenjem algebarskih tehnika. Specifične heurističke tehnike uključuju genetske algoritme, algoritme penjanja uzbrdo, algoritmi za simulirano kaljenje ili kombinaciju istih.

Zbog prirode heurističkih tehnika još uvek postoje mnoge mogućnosti za varijacije ovih tehnika kako bi se dalje poboljšale performanse i razmatrala druga kriptografski značajna svojstva ili kombinacije svojstava. [4]

U ovom radu je predstavljen algoritam iz disertacije [4] koji polazeći od jedne bent funkcije konstruiše više drugih bent funkcija većeg stepena. Algoritam je u osnovi heuristički, ali u implementaciji koristi pseudo-slučajnu tehniku zajedno sa algebarskim svojstvima Bulovih funkcija da bi brzo generisao bent funkcije.

Metod

Specifična svojstva koja bent funkcije poseduju čine osnovu algoritma za generisanje koji je predstavljen u nastavku rada. Dobro je poznato da bent funkcije mogu imati stepen od najviše $\frac{n}{2}$. Još jedno korisno svojstvo se može izvesti iz analize Hemingovih težina podfunkcija iz glave 2. Teorema 2.6.1 pokazuje da se Hemingove težine podfunkcija Bulove funkcije f mogu odrediti na osnovu njene WHT. Teorema 2.6.1 se može posebno primeniti na podfunkcije bent funkcije. Kada se bent funkcija podeli duž bilo kog nenula vektora w, tačno jedna od dve proizvedene podfunkcije je uravnotežena, odnosno bent funkcije su poluuravnotežene.

Posledica 3.0.1. *Ako je* $|\hat{F}(0)| = |\hat{F}(w)|$ *za neki nenula vektor* w*, onda je* $f(x)$ *pri podeli* $f(x) = [f_0|f_1]_w$ *poluuravnotežena u pravcu* w*, odnosno važi*

$$\mathrm{wt}(f_0) = 2^{n-2} \ \textit{ili} \ \mathrm{wt}(f_1) = 2^{n-2}$$

Dokaz. Ovo tvrđenje direktno sledi iz Teoreme 2.6.1. □

Posledica 3.0.2. *Ako je* $f(x)$ *bent funkcija od* n *promenljivih, onda za odabrano* w *i podelu* $f(x) = [f_0|f_1]_w$ *važi da je*

$$\mathrm{wt}(f_0) = 2^{n-2} \ \textit{ili} \ \mathrm{wt}(f_1) = 2^{n-2}$$

Slično, Bulova funkcija je bent funkcija ako i samo ako je poluuravnotežena u svakom nenula pravcu.

Dokaz. Uz zapažanje da za bent funkcije posledica 3.0.1 važi za svako $w : \hat{F}(w) = \pm 2^{\frac{n}{2}}$, sledi da su Hemingove težine podfunkcija bent funkcije uvek 2^{n-2} (što je

uravnotežena podfunkcija) i $2^{n-2} \pm 2^{\frac{n}{2}-1}$. Dokaz u suprotnom smeru pokazuje da bilo koja Bulova funkcija koja je poluuravnotežena u bilo kom nenula pravcu mora biti bent funkcija. □

Svojstvo poluuravnoteženosti zajedno sa maksimalnim stepenom za bent funkciju omogućava da se implementira filtrirajući proces. Osnovna ideja je da se počne sa kvadratnom bent funkcijom od n promenljivh, koja se može konstruisati korišćenjem jednostavnih algebarskih tehnika, i da se potom iterativno generišu bent funkcije većeg stepena dodajući njenoj ANF termove koji održavaju maksimalnu nelinearnost funkcije. Bez znanja o specifičnim svojstvima bent funkcija, ova tehnika ne bi bila efikasnija od nasumičnog generisanja funkcije i testiranja da li je funkcija bent. Ako bi se za svaki novi term koji odaberemo izračunavao WHT trenutne funkcije, ovo bi usporilo generisanja bent funckija.

Ova metoda relativno brzo odbacuje neodgovarajuće termove. Ograničenje bent funkcije da ima maksimalni stepen $\frac{n}{2}$ znači da se mogu uzeti u obzir samo ANF termovi koji ispunjavaju dati kriterijum, čime se smanjuje broj mogućih promena na ANF početne Bulove funkcije. Svojstvo poluuravnoteženosti bent funkcija omogućava dalje filtriranje ANF termova. Tek kada ANF term ispunjava uslove za stepen i Hemingovu težinu bent funkcije, taj term se može uzeti u obzir kao dodatak trenutnoj bent funkciji i nakon toga se određuje nelinearnost modifikovane funkcije. Dakle, potrebno je izvršiti znatno manje izračunavanja WHT nego za potpuno slučajni metod.

Algoritam

Najjednostavniji način da se generiše nova bent funkcija jeste da se krene od bent funkcije od n promenljivih, i da se potom od iste generiše nova bent funkcija. Za implementaciju ovog načina konstrukcije bent funkcija potrebno je uzeti u obzir nekoliko faktora. Korišćenjem gornje granice stepena ANF termova koji se ispituju mogu se generisati bent funkcije određenog stepena. Tehnika može varirati u odnosu na početnu kvadratnu bent funkciju koja se upotrebljava. Polazna kvadratna bent funkcija može biti fiksirana, izabrana tako da ima određenu algebarsku strukturu ili može biti slučajno generisana. Kako bi se testiralo svojstvo poluuravnoteženosti neophodno je da se ispitaju svi pravci podele funkcije. Eksperimentalna analiza je pokazala da je efikasnije izvršiti testiranje podele funkcije u pravcu n jediničnih

vektora [4]. Ovo je neophodno uraditi pre određivanja WHT, odnosno određivanja nelinearnosti funkcije koja predstavlja vremenski najzahtevniji deo algoritma. Ovaj metod omogućuje dobijanje više bent funkcija većeg stepena.

Ovaj algoritam za konstrukciju bent funkcija počinje od fiksirane kvadratne bent funkcije, a zatim pokušava sa dodavanjem novog terma u njenu ANF. Ukoliko funkcija ostaje maksimalno nelinearna dodavanjem tog terma, onda se ona postavlja za tekuću bent funkciju i na osnovu nje se ponovo traži novi term čijim dodavanjem funkcija ostaje bent. Algoritam se završava kada nije moguće dodati više nijedan novi term u funkciju a da ona ostane bent. Algoritam je opisan kodom 1.

Algoritam 1 Generisanje bent funkcija

Ulaz: n paran broj ulaznih promenljivih i $maxDeg$ maksimalni stepen

Izlaz: niz bent funkcija stepena $\leq maxDeg$

1: Kreirati niz A dužine 2^n koji predstavlja ANF funkcije i inicijalizovati ga nulama
2: **for** $i = 0, ..., \frac{n}{2} - 1$ **do**
3: $\quad j = 2^i + 2^{\frac{n}{2}+i}$
4: $\quad A_j = 1$
5: A sada predstavlja ANF kvadratne bent funkcije $\bigoplus_{i=0}^{\frac{n}{2}-1} x_i x_{n/2+i}$
6: Izabrati nasumično r tako da važi $0 \leq r \leq 2^n - 1$
7: **for** $i = 0, ..., 2^n - 1$ **do**
8: $\quad j = (i + r) \mod 2^n$
9: $\quad$ **if** $A_j = 0$ i $3 \leq \mathrm{wt}(j) \leq maxDeg$ **then**
10: $\quad\quad$ Kreirati $A^* = A$ i postaviti vrednost $A^*_j = 1$ (dodavanje j-tog terma)
11: $\quad\quad$ Izračunati istinitosnu tablicu $f(x)$ koja odgovara A^* (inverznom transformacijom ANF)
12: $\quad\quad$ **if** $(\forall w = e_i$ važi da $f = [f_0|f_1]_w$ ima $\mathrm{wt}(f) = 2^{n-1} \pm 2^{\frac{n}{2}-1}$ i važi da je $\mathrm{wt}(f_0)$ ili $\mathrm{wt}(f_1) = 2^{n-2})$ **then**
13: $\quad\quad\quad$ Pronaći N_f, odnosno nelinearnost of funkcije f
14: $\quad\quad\quad$ **if** $N_f = 2^{n-1} - 2^{\frac{n}{2}-1}$ (ako je bent funkcija) **then**
15: $\quad\quad\quad\quad$ Ažurirati $A = A^*$
16: $\quad\quad\quad\quad$ (A sada predstavlja ANF nove bent funkcije)
17: $\quad\quad\quad\quad$ Sačuvati f
18: $\quad\quad\quad\quad$ **goto** 6
19: Prikazati sačuvane bent funkcije

Malom modifikacijom prethodno napisanog algoritma dobijaju se isti rezultati u pogledu pronalaženja bent funkcija, ali se obiđe znatno manji broj ANF pozicija i postiže znatno ubrzanje programa. Modifikacija se ogleda u tome da se u koraku 6 vrednost za j ne uvećava za 1 između svake dve iteracije u petlji, već se uvećava za vrednost brojača i po modulu 2^n, na taj način se ne obilazi svaki term iz ANF.

Algoritam 2 Generisanje bent funkcija

Ulaz: n paran broj ulaznih promenljivih i $maxDeg$ maksimalni stepen

Izlaz: niz bent funkcija stepena $\leq maxDeg$

1: Kreirati niz A dužine 2^n koji predstavlja ANF funkcije i inicijalizovati ga nulama
2: **for** $i = 0, ..., \frac{n}{2} - 1$ **do**
3: $\quad j = 2^i + 2^{\frac{n}{2}+i}$
4: $\quad A_j = 1$
5: A sada predstavlja ANF kvadratne bent funkcije $\bigoplus_{i=0}^{\frac{n}{2}-1} x_i x_{n/2+i}$
6: Izabrati nasumično r tako da važi $0 \leq r \leq 2^n - 1$
7: Inicijalizovati $j = r$
8: **for** $i = 0, ..., 2^n - 1$ **do**
9: $\quad j = (i + j) \mod 2^n$
10: $\quad$ **if** $A_j = 0$ *i* $3 \leq \mathrm{wt}(j) \leq maxDeg$ **then**
11: $\quad\quad$ Kreirati $A^* = A$ i postaviti vrednost $A^*_j = 1$ (dodavanje j-tog terma)
12: $\quad\quad$ Izračunati istinitosnu tablicu $f(x)$ koja odgovara A^* (inverznom transformacijom ANF)
13: $\quad\quad$ **if** ($\forall w = e_i$ važi da $f = [f_0|f_1]_w$ ima $\mathrm{wt}(f) = 2^{n-1} \pm 2^{\frac{n}{2}-1}$ i važi da je $\mathrm{wt}(f_0)$ ili $\mathrm{wt}(f_1) = 2^{n-2}$) **then**
14: $\quad\quad\quad$ Pronaći N_f, odnosno nelinearnost of funkcije f
15: $\quad\quad\quad$ **if** $N_f = 2^{n-1} - 2^{\frac{n}{2}-1}$ (ako je bent funkcija) **then**
16: $\quad\quad\quad\quad$ Ažurirati $A = A^*$
17: $\quad\quad\quad\quad$ (A sada predstavlja ANF nove bent funkcije)
18: $\quad\quad\quad\quad$ Sačuvati f
19: $\quad\quad\quad\quad$ **goto** 6
20: Prikazati sačuvane bent funkcije

Primer 3.0.1. *Primer izvršavanja algoritma za ulaze $n = 4$ i $maxDeg = 2$:*

Niz A, koji predstavlja ANF Bulove funkcije gde element $A[j]$ predstavlja j koeficijent u nizu ANF, se u koraku 1 inicijalizuje nulama.

A:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

U koraku 2 konstruiše se početna bent funkcija $x_1x_3 + x_2x_4$, konkretno elementi A[5] i A[10] postavljaju se na 1.

A:	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0

U koraku 6 bira se nasumično r, konkretno u ovom primeru za vrednost se uzima 8. U koraku 7 pokušava se sa dodavanjem terma x_4 koji nije odgovarajućeg stepena, sledeći element sa kojim se pokušava je $A[9]$. Njemu odgovara kvadratni term x_1x_4 i jeste odgovarajućeg stepena. Konstruiše se novi privremeni niz A^* kao kopija niza A i u njega se dodaje element $A[9]$ i nakon toga se izračunava istinitosna tablica TT procedurom ANFT.

A*:	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0

TT:	0	0	0	0	0	1	0	1	0	1	1	0	0	0	1	1

Određuje se težina same funkcije f_{A^*}, $\mathrm{wt}(f_{A^*}) = 6$ što predstavlja deo uslova nenarušavanja poluuravnoteženosti. Potom se određuju podfunkcije f_0 i f_1 koje nastaju podelom duž jediničnih vektora, videti tabelu 3.1. Pri svakoj podeli bar jedna od dve podfunkcije je uravnotežena.

	f_0	f_1
e_1:	0 0 0 0 0 1 0 1	0 0 1 1 1 0 0 1
e_2:	0 0 0 1 0 1 0 0	0 0 0 1 1 0 1 1
e_3:	0 0 0 0 0 1 1 0	0 1 0 1 0 0 1 1
e_4:	0 0 0 0 0 1 0 1	0 1 1 0 0 0 1 1

Tabela 3.1: Podfunkcije f_0 i f_1.

Pošto su ispunjeni uslovi stepena terma i poluuravnoteženosti, izračunava se WHT ove Bulove funkcije.

WHT:	4	4	4	4	4	-4	-4	4	4	4	-4	-4	4	-4	4	-4

Apsolutne vrednosti svih koeficijanta WHT su jednake, što znači da funkcija postiže maksimalnu nelinearnost. Dodavanjem elementa $A[9]$ funkcija je i dalje bent. Niz A ažurira se ovim termom i program se vraća na pronalaženje novog r (korak 6). Bira se novo nasumično r čija je vrednost 0. Program zatim ispituje elemente sa indeksima 0, 1 i odbacuje ih. Element $A[3]$ je odgovarajućeg stepena, ispunjava algebarske uslove i on se takođe dodaje u ANF, čime se dobija nova bent funkcija. Nakon toga program više ne uspeva da doda ni jedan novi term u ANF. Broj pronađenih bent funkcija je 2.

Za Bulove funkcije sa manje od 12 promenljivih teško se uočava razlika između dva algoritma u vidu broja testiranih termova i broja uslova koji su ti termovi ispunili. Za Bulove funkcije sa 12 ili više promenljivih jasno se primećuje razlika između toka izvršavanja ova dva algoritma. Stoga je dat primer rada prvog i drugog algoritma za Bulovu funkciju od 12 promenljivih.

Primer 3.0.2. *Primer rada prvog algoritama za broj promenljivih 12 i maksimalni stepen 3:*

U koraku 1 inicijalizuje se niz A od 2^n elemenata i svi se postavljaju na nulu. Niz A predstavlja ANF Bulove funkcije gde element $A[j]$ predstavlja j term u nizu ANF. U koraku 2 konstruiše se početna kvadratna bent funkcija tako što se biraju indeksi termova po unapred određenoj formuli. Konkretno za ovaj primer indeks j uzima sledeće vrednosti 65, 130, 260, 520, 1040, 2080, a odgovarajuća kvadratna funkcija je $x_1x_6 + x_2x_7 + \ldots + x_6x_{11}$. U koraku 6 bira se nasumično r tako da važi $0 \leq r \leq 2^n - 1$. Izabrano r određuje indeks od kog se počinje pretraga u koraku 7. Ovaj korak služi da se različitim pokretanjem programa generišu različite bent funkcije. U koraku 7 vrši se pretraga za termovima koji će svojim dodavanjem napraviti novu bent funkciju. U ovom primeru pretraga je započeta od elementa $A[3457]$ (vrednost r iz prethodnog koraka). Od narednih 96 termova koji su testirani nijedan nije odgovarajućeg stepena. Element $A[3584]$ ispunjava uslove odgovarajućeg stepena, nenarušavanja poluuravnoteženosti funkcije i maksimalne nelinearnosti. Stoga dodavanjem ovog terma funkcija i dalje ostaje minimalna, odnosno bent, a sadrži jedan term više. Nakon ovog koraka izvršavanje algoritma se nastavlja od koraka 6, gde se bira novo nasumično r i na isti način se traga za novim termovima čijim dodavanjem se pronalaze nove bent funkcije.

Broj testiranih termova	24704
Broj termova koji ispunjavaju uslov odgovarajućeg stepena	1402
Broj termova koji ispunjavaju uslov nenarušavanja uravnoteženosti	193
Broj termova koji ispunjavaju uslov nelinearnosti (br. bent funkcija)	41

Tabela 3.2: Konačan rezultat prve verzije algoritma u primeru 3.0.2

.

Primer 3.0.3. *Primer rada drugog algoritma za broj promenljivih 12 i maksimalni stepen 3:*

Prva tri koraka su ista kao i u prvoj verziji algoritma. Razlika se javlja u koraku 8. U prvoj verziji algoritma u prvoj iteraciji koraka 7 prvi testirani term je izabrano nasumično r, dok je indeks svakog sledećeg terma za jedan veći od indeksa prethodno testiranog terma. U drugoj verziji algoritma ponovo se kreće od nasumično izabranog r, dok je indeks novog terma jednak indeksu prethodnog uvećan za vrednost brojača i iz petlje. Na taj način se izbegava obilaženje svakog terma, a rezultati pokretanja programa pokazali su da se dobija približno isti broj bent funkcija. Konkretno pretraga je započeta od elementa $A[2940]$, indeksi narednih elemenata koji su obiđeni su 2941, 2943, 2946, 2950, 2955, 2961, 2968, 2976, 2985... Prvi sledeći element koji ispunjava sve uslove i čijim dodavanjem se dobija prva bent funkcija jeste element $A[3076]$.

Broj testiranih termova	20590
Broj termova koji ispunjavaju uslov odgovarajućeg stepena	932
Broj termova koji ispunjavaju uslov nenarušavanja uravnoteženosti	148
Broj termova koji ispunjavaju uslov nelinearnosti (br. bent funkcija)	41

Tabela 3.3: Konačan rezultat druge verzije algoritma u primeru 3.0.3

.

U glavi 5 će biti detaljnije razmotrene razlike izvršavanja implementacija ova dva algoritma.

Glava 4

Implementacija

U ovoj glavi prikazane su glavne funkcije koje se koriste u samom algoritmu, a koje su od ključnog značaja za brzinu izvršavanja programa. Glavne funkcije koje se razmatraju su:

- funkcija za izračunavanje ANF od tablice istinitosti;
- funkcija za izračunavanje WHT niza od polarne tablice istinitosti, pri čemu je ova funkcija od velike važnosti za izračunavanje nelinearnosti;
- funkcija split, odnosno funkcija iz uslova 12 algoritma. Ova funkcija vrši filtriranje termova pre određivanja nelinearnosti;

Za potrebe implementacije odabran je programski jezik C.

4.1 Izazovi pri implementaciji

Neophodno je da se razvije efikasna implementacija ovih funkcija kako bi se sam program brže izvršavao. Prvo su razmotreni neki opšti poznati principi optimizacije, a potom su primenjene i neke specifične tehnike nakon analize samih procedura.

Kako bi se optimizovao program, neophodno je smanjiti broj operacija koje procesor izvšava, odnosno smanjiti broj operacija koje zahtevaju duže vreme obrade. Uslovna operacija kao što je IF naredba zahteva više obrade od ADD naredbe ili nekog od bitovskih operatora (npr. XOR, AND, itd.). Skupe operacije

su i množenje i deljenje. Stoga je cilj da se koristi ne samo manji broj operacija već i manji broj skupih operacija.

Neke opšte tehnike optimizacije korišćene prilikom implementacije su:

- Odmotavanje petlje (eng. *loop unrolling*) – For petlja u svakoj iteraciji koristi naredbu if, kao i povećanje promenljive za iteriranje kroz petlju. Glavna prednost korišćenja tehnike odmotavanja je što se korišćenjem dodatnih linija koda smanjuje broj iteracija petlje i na taj način se smanjuje broj izvršenih uslovnih naredbi.

- Smanjenje snage (eng. *strength reduction*) – Ova tehnika podrazumeva zamenu skupih operacija jeftinijim. Mnogi prevodioci automatski će implementirati ovu tehniku, međutim ponekad ova tehnika može da se koristi prilikom pisanja izvornog koda. Na primer operacija množenja ili deljenja stepenom dvojke se može zameniti operacijama levog i desnog šiftovanja (SHIFT).

- Unapred spremljene tabele (eng. *look-up table*) – Unapred spremljena tabela predstavlja niz unapred izračunatih vrednosti i ako se koristi na odgovarajući način često može da donese velike uštede vremena. Korisnost ovih tabela zavisi od dostupnosti radne i keš memorije. Upotreba tabela je efikasna jedino ako je cena kreiranja i pristupa vrednosti iz unapred spremljene tabele manja od cene izvršavanja originalnog izraza.

4.2 Analiza i opis implementacije glavnih funkcija koje se koriste u algoritmu

Kako bi se na jednostavan način vršila obrada i evaluacija Bulove funkcije neophodno je definisati strukture podataka koje bi se koristile za predstavljanje TT i ANF Bulovih funkcija. Struktura podataka koja najbolje predstavlja ove podatke je niz. Najprostije rešenje jeste da to bude niz celobrojnih tipova. Međutim, ovakva implementacija bi dovela do neefikasnog iskorišćenja memorije, jer elementi TT i ANF su iz domena $\{0, 1\}$. Umesto celobrojnog tipa dovoljno je za svaki element rezervisati jedan bit u memoriji. Na taj način struktura podataka i dalje ostaje niz, ali memorijski objekat zauzima minimalnu potrebnu količinu prostora. Iz algoritamskog aspekta mana je što se komplikuje pristup elementima

niza, ali prednost je u korišćenju bitovskih operacija nad procesorskim rečima, što emulira vektorske operacije nad intervalima elemenata.

Za pristupanje i modifikaciju pojedinačnih 32-bitnih elemenata niza potrebne su sledeće funkcije:

```
static inline void set_bit(unsigned* packed_array, long long
                           unsigned bit_index, unsigned value)
{
    if (value) {
        packed_array[bit_index >> 5] |= (1<<(bit_index&31));
    } else {
        packed_array[bit_index >> 5] &= ~(1<<(bit_index&31));
    }
}

static inline unsigned get_bit(unsigned* packed_array, long long
                               unsigned bit_index)
{
    return !!((1<<(bit_index&31)) & (packed_array[bit_index >> 5]));
}
```

Slika 4.1: Funkcije za pristupanje i modifikaciju pojedinačnih elemenata niza.

Svi rezultati vremena izvršavanja koji su prikazani u ovoj glavi za $n \leq 16$ su dobijeni na računaru sa Intel(R) Core(TM) i7-1165G7 procesorom i 16GB RAM memorije sa operativnim sistemom Windows 10. Za $n \geq 18$ rezultati su dobijeni na serveru sa Intel(R) Xeon(R) Gold 6154 CPU procesorom i 120GB RAM memorije sa operativnim sistemom Windows 10.

Izračunavanje algebarske normalne forme

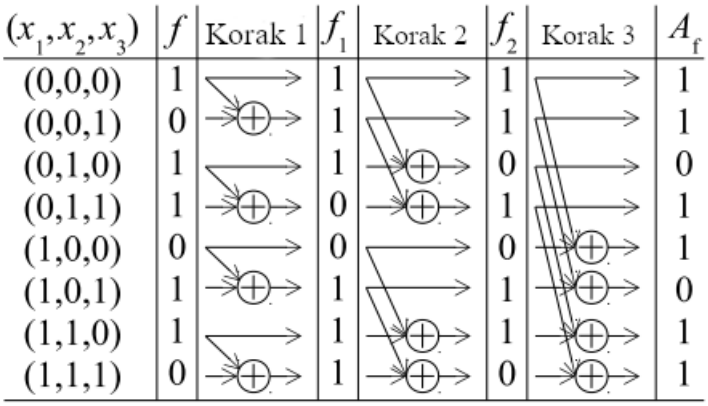
Algebarska normalna forma (ANF) predstavlja jedinstvenu reprezentaciju Bulove funkcije i olakšava izračunavanje stepena i algebarske težine, koji predstavljaju značajna kriptografska svojstva. Od velike važnosti je da se transformacija Bulove funkcije iz formata tabele istinitosti u njen ANF izvrši na efikasan način.

Na osnovu teoreme 2.3.1 data je najjednostavnija implementacija ANFT na sledeći način:

```
void anft_naive(unsigned * tt, unsigned* anf, unsigned n) {
    for (int i=0; i<(int)pow(2,n); i++) {
        anf[i]=tt[i];
    }
    for (int i=0; i<n; i++) {
        for (int j=0; j<(int)pow(2,n); j=j+(int)pow(2,i+1)) {
            for (int k=j; k<j+(int)pow(2,i); k++) {
                anf[k+(int)pow(2,i)]=(anf[k]+anf[k+(int)pow(2,i)])%2;
            }
        }
    }
}
```

Slika 4.2: Jednostavna implementacija ANFT.

Algoritam se može na slikovit način predstaviti leptir dijagramom (eng. *butterfly diagram*). Primer za funkciju $f(x_1, x_2, x_3) = (1, 0, 1, 1, 0, 1, 1, 0)$ prikazan je na slici 4.3 [1].



Slika 4.3: Leptir algoritam za $f(x_1, x_2, x_3) = (1, 0, 1, 1, 0, 1, 1, 0)$.

Prethodni algoritam za ANFT može se prevesti u bitovski. U svakom koraku formiraju se bitovske maske koje biraju jednu polovinu operanada, a onda se šiftovanjem poravnavaju operandi koji treba da se saberu za dati korak u dijagramu. U tabeli 4.2 prikazani su koraci za funkciju $f(x_1, x_2, x_3) = (1, 0, 1, 1, 0, 1, 1, 0)$ (SHR označava šiftovanje udesno) [1].

Korak	Operacije	Vrednosti operacija
Ulaz	f	10110110
1.0	m_1	10101010
1.1	$temp = f\ AND\ m_1$	10100010
1.2	$temp = temp\ SHR\ 1$	01010001
1.3	$f = f\ XOR\ temp$	11100111
2.0	m_2	11001100
2.1	$temp = f\ AND\ m_2$	11000100
2.2	$temp = temp\ SHR\ 2$	00110001
2.3	$f = f\ XOR\ temp$	11010110
3.0	m_3	11110000
3.1	$temp = f\ AND\ m_3$	11010000
3.2	$temp = temp\ SHR\ 4$	00001101
3.3	$f = f\ XOR\ temp$	11011011
Izlaz	$A_f = f$	11011011

Tabela 4.1: ANFT za $f(x_1, x_2, x_3) = (1, 0, 1, 1, 0, 1, 1, 0)$.

U prvom koraku koristi se maska $m_1 = 10101010$, čija je heksadecimalna vrednost 0xAA. U promenljivu *temp* smešta se vrednost za $f\ AND\ m_1$, čime se čuvaju operandi za naredna dva koraka. Potom se vrši desno šiftovanje promenljive *temp* za jedan bit, i izvršava se XOR operacija između f i *temp* promenljive i vrednost se smešta u f. U drugom koraku koristi se maska $m_2 = 11001100$ i vrši se desno šiftovanje za 2 bita, i XOR operacija između f i *temp*. U trećem koraku maska treba da izgleda $m_3 = 11110000$, i desno šiftovanje se vrši za 4 bita, i XOR operacija između f i *temp*. Na kraju se za A_f dobija $A_f = (1, 1, 0, 1, 1, 0, 1, 1)$, to jest ANF funkcije f je $f(x_1, x_2, x_3) = 1 \oplus x_3 \oplus x_2x_3 \oplus x_1 \oplus x_1x_2 \oplus x_1x_2x_3$. Za bilo koju funkciju od tri promenljive bitovski ANFT se izvršava na isti način korišćenjem tri operacije u jeziku C kao na slici 4.4. U f se smešta zbir od f i njegovog svakog drugog šiftovanog bita, a onda zbir novog f i njegov svaki drugi šiftovani par bitova i na kraju se na nižih četiri bita od f dodaje viših 4 bita od f.

```
1 f^= ( f & m1) >> 1; f^= ( f & m2) >> 2; f^= f >> 4;
```

Slika 4.4: Primer funkcija za ANFT.

Za funkcije od 4 promenljive bitovski ANFT se može uraditi na sličan način, ali

se maske moraju udvostručiti, tako što se svaka konkatenira sa samom sobom kako bi predstavljale 2 bajta. Maske su redom $m_1 = 1010101010101010$ (0XAAAA), $m_2 = 1100110011001100$ (0XCCCC), $m_3 = 1111000011110000$ (0XF0F0), $m_4 = 1111111100000000$ (0XFF00). Za funkcije od 5 promenljivih koriste se četiri bajta, tako da je neophodno ponovo udvostručiti maske.

Za $f \in F_n$ gde važi $3 \leq n \leq 6$, izvršava se n koraka. Prvih $n-1$ koraka koristi četiri operacije. Poslednji korak može koristiti samo tri operacije, jer se može preskočiti operacija AND sa maskom, s obizorom da su bitovi najveće težine već na odgovarajućim pozicijama, a nakon toga odmah se vrši šiftovanje. Iz ovoga sledi da za A_f zahteva izvršavanje $4n-1$ operacija. Kada se koristi bitovska reprezentacija složenost za zauzetost memorije je konstantna, dok je vremenska složenost linearna, jer je broj koraka $4n-1$.

U drugom slučaju kada $f \in F_n$ gde $n > 6$, koristi se bitovska reprezentacija f u vidu niza 2^{n-6} procesorskih reči veličine $2^6 = 64$ bitova. Na taj način koristi se 64 puta manje memorije. ANFT algoritam izvršava dva glavna koraka:

- Korak 1: Kao što je već opisano, bitovski ANFT za svaku procesorsku reč, s obzirom da imamo 2 dodatna koraka pri izračunavanju, postoji tačno $4 \cdot 6 - 1 + 2 = 25$ operacija za jednu procesorsku reč. Ukupno postoji 2^{n-6} procesorskih reči tako da $25 \cdot 2^{n-6}$ predstavlja broj operacija.
- Korak 2: U ovom koraku izvršava se ANFT sa bitovskim XOR operacijama sa celim procesorskim rečima. Broj operacija je $(n-6) \cdot 2^{n-7}$.

Za $f \in F_n$ gde važi $n > 6$ ukupan broj koraka bitovskog ANFT iznosi:

$$(25 \cdot 2^{n-6}) + ((n-6) \cdot 2^{n-7}) = ((n+44) \cdot 2^{n-7})$$

Algoritam koji se upotrebljava za izračunavanje bitovskog ANFT niza u C kodu predstavljen je na sledeći način:

```
typedef unsigned long long ull;
const ull m1= ~0XAAAAAAAAAAAAAAAA,
          m2= ~0XCCCCCCCCCCCCCCCC,
          m3= ~0XF0F0F0F0F0F0F0F0,
          m4= ~0XFF00FF00FF00FF00,
          m5= ~0XFFFF0000FFFF0000;

void ANF ( ull f[], unsigned num_of_vars ) {
    int num_of_steps = num_of_vars - 6;
    int num_of_comp_words= 1<<num_of_steps ;
// Step 1: bitwise ANF on computer words
    for ( int k= 0; k < num_of_comp_words; k++) {
        ull temp= f[k];
        temp ^= (temp & m1)<<1;
        temp ^= (temp & m2)<<2;
        temp ^= (temp & m3)<<4;
        temp ^= (temp & m4)<<8;
        temp ^= (temp & m5)<<16;
        temp ^= temp<<32;
        f[k]= temp;
    }

    // Step 2
    int blocksize = 1;
    for (int step= 1; step <= num_of_steps ; step++) {
        int source= 0;
        while ( source < num_of_comp_words) {
            int target= source + blocksize ;
            for ( int i= 0; i < blocksize ; i++) {
                f[ target + i ] ^= f[ source + i ] ;
            }
            source += 2 *blocksize;
        }
        blocksize *= 2;
    }
}
```

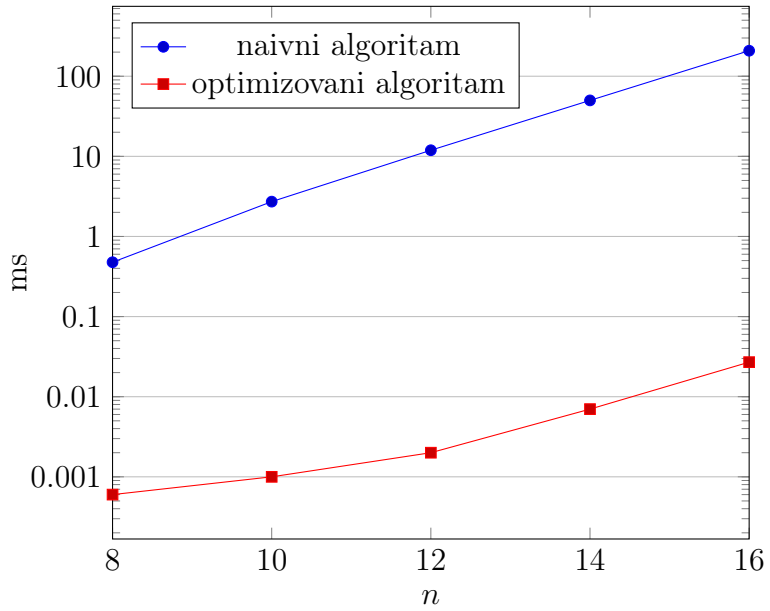
Slika 4.5: Bitovski algoritam ANFT.

Poređenje rezultata za izvršavanje naivnog i optimizovanog algoritma dato je u tabeli 4.2. U tabeli su navedena vremena izvršavanja naivnog i optimizovanog algoritma kao i postignut stepen ubrzanja. Na slici 4.6 prikazan je grafički prikaz zavisnosti

vremena od n za tabelu 4.2.

ANFT algoritam	n				
	8	10	12	14	16
naivni algoritam	0.477	2.721	11.921	49.922	208.144
optimizovani algoritam	0.0006	0.001	0.002	0.007	0.027
postignuto ubrzanje	795.0	2721.0	5960.5	7131.7	7709.0

Tabela 4.2: Pregled vremena izvršavanja algoritma ANF izraženo u ms za deset funkcija.



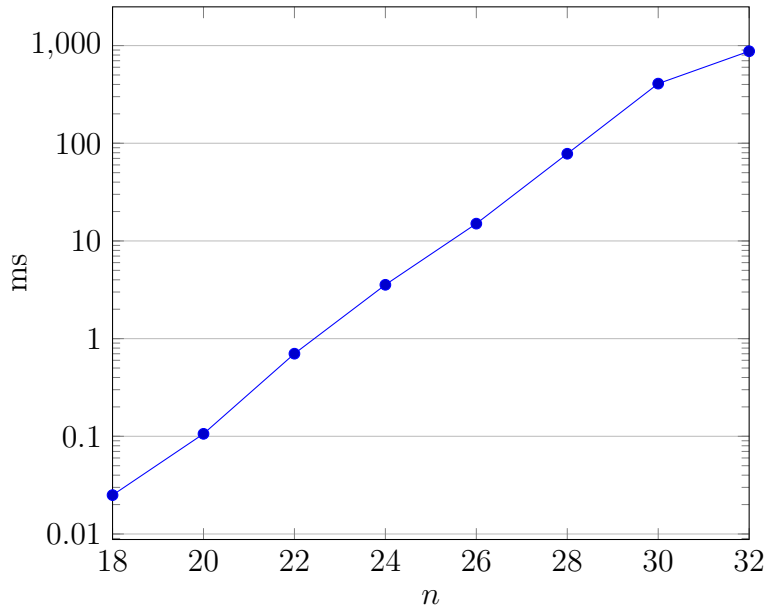
Slika 4.6: Grafički prikaz zavisnosti vremena od n za tabelu 4.2.

Najmanji očekivani stepen ubrazanja je 64 puta brže izvršavanje, ali se vidi da je razlika u brzini veća od toga, čak i da raste sa brojem promenljivih Bulove funkcije. Prvi razlog je što za malo n broj koraka ANFT preskočenih u prvom delu algoritma predstavlja veći udeo ukupnog posla, a za veće n razlika se objašnjava tako što je kod prvog algoritma loša iskorišćenost memorijskog keša naspram drugog, jer je potrebno 64 puta više prostora da se ceo ANF predstavi u memoriji.

Za jako velike brojeve $n > 16$ takođe je dato prosečno vreme izvršavanja algoritma ANF za jednu funkciju u tabeli 4.3. Vreme je izraženo u ms. Na slici 4.7 prikazan je grafički prikaz zavisnosti vremena od n za tabelu 4.3.

ANFT algoritam	n							
	18	20	22	24	26	28	30	32
optimizovani algoritam	0.025	0.106	0.701	3.557	15.033	78.101	407.882	875.229

Tabela 4.3: Pregled vremena izvršavanja algoritma ANF izraženo u ms za 1 funkciju.



Slika 4.7: Grafički prikaz zavisnosti vremena od n za tabelu 4.3.

Izračunavanje WHT

Drugi način da se jedna Bulova funkcija jedinstveno predstavi je njena Volš-Adamardova transformacija (WHT). Elementi WHT se nazivaju koeficijentima WHT Bulove funkcije. Na osnovu najveće apsolutne vrednosti među koeficijentima se određuje nelinearnost Bulove funkcije, a samim tim da li je funkcija bent.

Definicija 4.2.1. *Relacija između WHT i TT Bulove funkcije može se izraziti na sledeći način:* $\hat{F} = H_n \cdot \hat{f}$*, gde je* $H_n \pm 1$ *matrica veličine* $2^n \cdot 2^n$ *koja se rekurzivno definiše na sledeći način*

$$H_n = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes H_{n-1} \text{ gde je } H_0 = [1]$$

Na osnovu prethodne definicije prikazan je program za određivanje WHT koji pretpostavlja da TT ne koristi bitovsku reprezentaciju.

```
1  void wht(unsigned* tt, int* wht, int* temp, unsigned n){
2      for (unsigned i=0; i<(int)pow(2,n); i++){
3          wht[i]=(1 - 2*tt[i]);
4      }
5
6      for (unsigned i=0; i<n; i++) {
7          for (unsigned j=0; j<(int)pow(2,n); j=j+(int)pow(2,i+1)) {
8              for (unsigned k=j; k<j+(int)pow(2,i); k++) {
9                  int a=wht[k]-wht[k+(int)pow(2,i)];
10                 wht[k]=wht[k]+wht[k+(int)pow(2,i)];
11                 wht[k+(int)pow(2,i)]=a;
12             }
13         }
14     }
15 }
```

Slika 4.8: Naivni algoritam WHT.

Algoritam WHT može se na slikovit način predstaviti leptir dijagramom. Jedan primer izvršavanja algoritma WHT za vektor t, dužine 2^n u n koraka prikazan je na slici 4.9. Na samom početku $t = t^{(0)} = PTT(f)$. Da bi se izvršio prvi korak, neophodno je particionisati $t^{(0)}$ u 2^{n-1} parova na sledeći način

$(t_0, t_1), (t_2, t_3), ..., (t_{2^n-2}, t_{2^n-1})$.

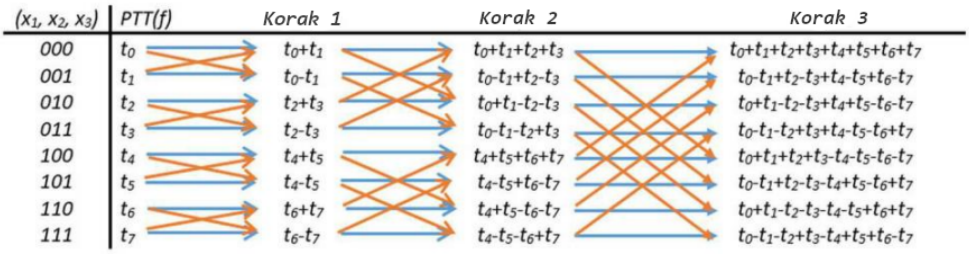
Novi vektor se može prikazati na sledeći način:

$$t^{(1)} = (t_0 + t_1, t_0 - t_1, t_2 + t_3, t_2 - t_3, ..., t_{2^n-2} + t_{2^n-1}, t_{2^n-2} - t_{2^n-1}).$$

U koraku s gde $1 \leq s \leq n$, vektor (odnosno tabela) se particioniše u intervale dužine 2^s i primenjuju se odgovarajući proračuni na sledeći način:

$$t_i^{(s)} = t_i^{(s-1)} + t_{i+j}^{(s-1)},\ t_{i+j}^{(s)} = t_i^{(s-1)} - t_{i+j}^{(s-1)}$$

za svako i gde $0 \leq i \leq 2^n$, tako da važi $i \equiv 0, 1, ..., 2^{s-1} - 1 \pmod{2^s}$, $j = 2^{s-1}$.



Slika 4.9: Algoritam WHT za funkciju od tri promenljive.

Tabelu TT koja je u bitovskoj reprezentaciji treba prevesti u WHT koji je u celobrojnoj reprezentaciji. Pod pretpostavkom da su elementi memorijskog objekta TT 32-bitni celobrojni tipovi, to bi značilo da se za svaki element memorijskog objekta TT treba izvršiti 5 koraka neoptimizovanog algoritma WHT, kako bi se dobilo po 32 elementa niza WHT.

Prelaz iz bitovskog TT u celobrojni WHT može se optimizovati sledećim tehnikama:

- unapred spremljenom tabelom
- odmotavanjem petlje

Pre izvršavanja algoritma WHT treba da se izračuna za svaku moguću TT od osam elemenata njen odgovarajući WHT od osam koeficijenata i staviti u unapred spremljenu tabelu, tako da je indeks tog WHT njen odgovarajući TT. Korišćenjem te tabele izbegavaju se prva tri od pet koraka algoritma WHT potrebnih da se

32-bitni elementi memorijskog objekta TT pretvore u WHT od 32 koeficijenta. Nakon korišćenja unapred izračunatih WHT nizova, primenjeno je odmotavanje petlje na preostala dva koraka algoritma WHT i na taj način od četiri niza od po 8 WHT koeficijenata izračunava se niz od 32 koeficijenta WHT u jednom koraku.

```
int W[256][8];

/*Generisanje svih WHT nizova od 8 koeficijenata*/
void Matrix_W() {

int i, j, p, q, fn=0, w[4][2]={{2,0},{0,-2},{0,2},{-2,0}};
for (i=0; i<4; i++)
    for (j=0; j<4; j++)
        for (p=0; p<4; p++)
            for (q=0; q<4; q++) {
                W[fn][0]=w[q][0]+w[p][0]+w[j][0]+w[i][0];
                W[fn][1]=w[q][1]+w[p][1]+w[j][1]+w[i][1];
                W[fn][2]=w[q][0]-w[p][0]+w[j][0]-w[i][0];
                W[fn][3]=w[q][1]-w[p][1]+w[j][1]-w[i][1];
                W[fn][4]=w[q][0]+w[p][0]-w[j][0]-w[i][0];
                W[fn][5]=w[q][1]+w[p][1]-w[j][1]-w[i][1];
                W[fn][6]=w[q][0]-w[p][0]-w[j][0]+w[i][0];
                W[fn][7]=w[q][1]-w[p][1]-w[j][1]+w[i][1];
                fn++;
            }

};

void WHT(int *tt, int *wht, int n) {

    int i, j, k, a, b, l, size=(1<<n), N=size>>5;
    /* U jednom koraku ove petlje koriscenjem unapred spremljene
    tabele i odmotavanjem petlji se od svakog 32-bitnog broja
    izracunava odgovarajuci WHT od 32 koeficijenta*/
    for (i=0; i<N; i++) {
        l=i*32;
        a=tt[i]&0xFF;
        b=(tt[i]&0xFF00)>>8;
        j=(tt[i]&0xFF0000)>>16;
        k=(tt[i]&0xFF000000)>>24;
        wht[l]=W[a][0]+W[b][0]+W[j][0]+W[k][0];
        wht[l+1]=W[a][1]+W[b][1]+W[j][1]+W[k][1];
```

```
38          wht[l+2]=W[a][2]+W[b][2]+W[j][2]+W[k][2];
39          wht[l+3]=W[a][3]+W[b][3]+W[j][3]+W[k][3];
40          wht[l+4]=W[a][4]+W[b][4]+W[j][4]+W[k][4];
41          wht[l+5]=W[a][5]+W[b][5]+W[j][5]+W[k][5];
42          wht[l+6]=W[a][6]+W[b][6]+W[j][6]+W[k][6];
43          wht[l+7]=W[a][7]+W[b][7]+W[j][7]+W[k][7];
44          wht[l+8]=W[a][0]-W[b][0]+W[j][0]-W[k][0];
45          wht[l+9]=W[a][1]-W[b][1]+W[j][1]-W[k][1];
46          wht[l+10]=W[a][2]-W[b][2]+W[j][2]-W[k][2];
47          wht[l+11]=W[a][3]-W[b][3]+W[j][3]-W[k][3];
48          wht[l+12]=W[a][4]-W[b][4]+W[j][4]-W[k][4];
49          wht[l+13]=W[a][5]-W[b][5]+W[j][5]-W[k][5];
50          wht[l+14]=W[a][6]-W[b][6]+W[j][6]-W[k][6];
51          wht[l+15]=W[a][7]-W[b][7]+W[j][7]-W[k][7];
52          wht[l+16]=W[a][0]+W[b][0]-W[j][0]-W[k][0];
53          wht[l+17]=W[a][1]+W[b][1]-W[j][1]-W[k][1];
54          wht[l+18]=W[a][2]+W[b][2]-W[j][2]-W[k][2];
55          wht[l+19]=W[a][3]+W[b][3]-W[j][3]-W[k][3];
56          wht[l+20]=W[a][4]+W[b][4]-W[j][4]-W[k][4];
57          wht[l+21]=W[a][5]+W[b][5]-W[j][5]-W[k][5];
58          wht[l+22]=W[a][6]+W[b][6]-W[j][6]-W[k][6];
59          wht[l+23]=W[a][7]+W[b][7]-W[j][7]-W[k][7];
60          wht[l+24]=W[a][0]-W[b][0]-W[j][0]+W[k][0];
61          wht[l+25]=W[a][1]-W[b][1]-W[j][1]+W[k][1];
62          wht[l+26]=W[a][2]-W[b][2]-W[j][2]+W[k][2];
63          wht[l+27]=W[a][3]-W[b][3]-W[j][3]+W[k][3];
64          wht[l+28]=W[a][4]-W[b][4]-W[j][4]+W[k][4];
65          wht[l+29]=W[a][5]-W[b][5]-W[j][5]+W[k][5];
66          wht[l+30]=W[a][6]-W[b][6]-W[j][6]+W[k][6];
67          wht[l+31]=W[a][7]-W[b][7]-W[j][7]+W[k][7];
68      }
69
70      for (i=32; i<size; i=i<<1)
71          for (j=0; j<size; j=j+(i<<1))
72              for (k=j; k<j+i; k++) {
73                  a=wht[k]-wht[k+i];
74                  wht[k]=wht[k]+wht[k+i];
75                  wht[k+i]=a;
76              }
77  };
```

Slika 4.10: Optimizovani algoritam WHT.

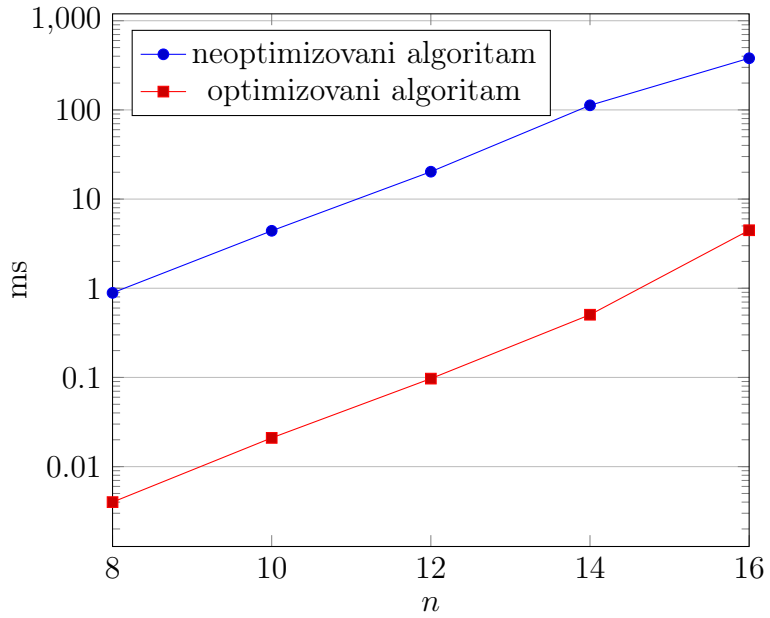
S obzirom da se prilikom transformacije TT u WHT, a i u ostatku algoritma WHT, vrednost koeficijenta WHT menja jedanput po koraku, moguće je postići dalje ubrzanje primenom višenitnog izvršavanja. Prilikom transformacije TT u WHT računanja koeficijenata WHT su međusobno nezavisna, pa je moguće podeliti ceo WHT na podintervale koje će svaka nit nezavisno da izračunava. Isti princip se može primeniti na najugnježdeniju petlju algoritma WHT: računanje WHT koeficijenata od iteracije do iteracije su nezavisna pa se iteracije petlje mogu raspodeliti na niti. Ovoj tehnici se mora pažljivo pristupiti, jer ako je količina posla koja se deli na niti premala može čak da se dovede do usporenja algoritma.

U tabeli 4.4 je prikazano vreme izvršavanja neoptimizovanog algoritma i optimizovanog bez višenitnog izvršavanja, kao i postignuta stopa ubrazanja. Na slici 4.11 prikazan je grafički prikaz zavisnosti vremena od n.

WHT algoritam	n				
	8	10	12	14	16
neoptimizovani algoritam	0.888	4.409	20.275	112.592	379.766
optimizovani algoritam	0.004	0.021	0.097	0.505	4.473
postignuto ubrzanje	222.0	209.9	209.0	222.9	84.9

Tabela 4.4: Pregled izvršavanja algoritma WHT u ms za 10 funkcija.

S obzirom da memorijski objekti u naivnoj i optimizovanoj implementaciji WHT koriste istu količinu memorijskog prostora, jedino ubrzanje koje može da se očekuje dolazi iz računskih optimizacija. Pošto se odmotavanje petlji i korišćenje unapred izračunatih vrednosti koristi da se ubrza izračunavanje samo prvih pet koraka WHT, stepen ubrzanja dobijen sa ove dve tehnike će opadati sa porastom ukupnog broja koraka. Broj koraka WHT je jednak broju promenljivih Bulove funkcije i zato se može videti da stepen ubrzanja opada već od $n = 16$. Ubrzanja za $n > 20$ su većim delom postignuta višenitnim izračunavanjem.

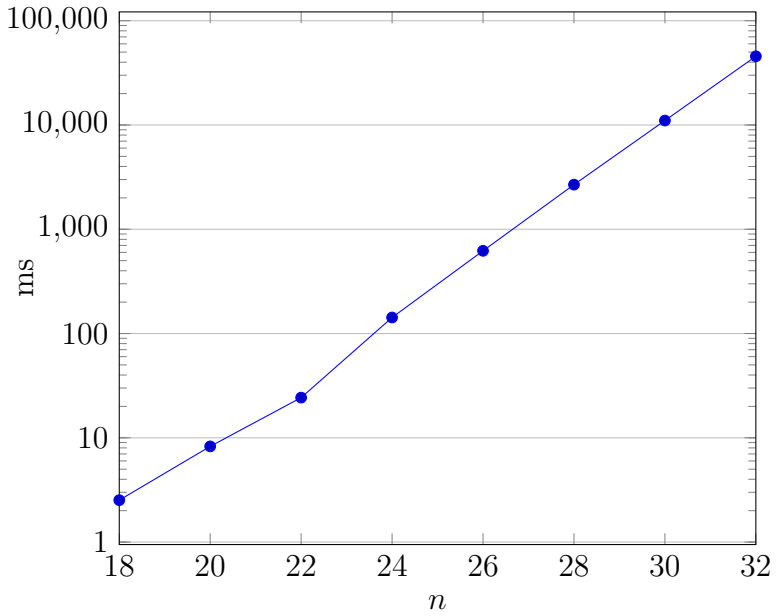


Slika 4.11: Grafički prikaz zavisnosti vremena od n za tabelu 4.4.

S obzirom da za malo n višenitno izvršavanje ne ubrzava algoritam, tabela optimizovanog algoritma za višenitno izvršavanje prikazana je u tabeli 4.5, dok je grafički prikaz zavisnosti vremena od n prikazano na slici 4.12.

WHT	n							
alg.	18	20	22	24	26	28	30	32
opt. alg.	2.522	8.272	24.283	142.487	621.192	2678.005	11028.173	45587.443

Tabela 4.5: Pregled izvršenja algoritma WHT za veliko n.



Slika 4.12: Grafički prikaz zavisnosti vremena od n za tabelu 4.5.

Funkcija split

Funkcija split iz algoritma 2 u koraku 13 vrši filtriranje termova pre određivanja nelinearnosti, na osnovu posledice 3.0.2, odnosno činjenice da je Bulova funkcija bent funkcija ako i samo ako je poluuravnotežena u svakom nenula pravcu.

Prema posledici 3.0.2, funkcija split vrši filtriranje termova tako što za svaki jedinični vektor od funkcije f generiše podfunkcije f_0 i f_1, i potom određuje Hemingovu težinu za funkcije f, f_0 i f_1. Ukoliko važi da je $\mathrm{wt}(f) = 2^{n-1} \pm 2^{\frac{n}{2}-1}$ i važi da je $\mathrm{wt}(f_0)$ ili $\mathrm{wt}(f_1) = 2^{n-2}$ za svako $w = e_i$ term se uzima u obzir i određuje se nelinearnost kako bi se proverilo da li je funkcija koja uključuje novi term bent funkcija.

Naivna implementacija funkcije split data je na slici 4.10.

```
unsigned scalar_product_unit_vector(long long unsigned x,
long long unsigned unit_vector) {
    return x & unit_vector;
}
void split_along_direction_e_n(unsigned * f, unsigned n,
                               unsigned* f_wt, unsigned ei)
{
    unsigned f0_counter = 0, f1_counter = 0;
    for (long long unsigned x = 0; x < (unsigned)pow(2,n); x++) {
        if (scalar_product_unit_vector(x, ei) == 0) {
            f0_counter += f[x];
        } else {
            f1_counter += f[x];
        }
        *f_wt += f[x];
    }
}
```

Slika 4.13: Funkcija split.

Bitovskom reprezentacijom TT Bulove funkcije moguće je koristiti ugrađene instrukcije x86 procesora koje prebrojavaju sve nenula bitove u jednoj procesorskoj reči.

Jedinični vektori određuju koji bitovi tabele TT pripadaju kojoj podfunkciji, i to

tako što dele TT na smenjujuće podintervale gde bitovi pripadaju ili jednoj ili drugoj podfunkciji. Ako bar 64 bita uzastopno pripadaju jednoj podfunkciji, moguće je koristiti ugrađenu procesorsku instrukciju *popcount*. U suprotnom koristi se naivni pristup i mora bit po bit da se iterira kroz TT i da se računaju težine podfunkcija.

```
1
2  unsigned count(const unsigned a, long long unsigned length)
3  {
4      unsigned n = 0;
5      typedef unsigned long long __attribute__((may_alias)) ull_a;
6      const ull_a *p;
7
8      for (p = (const ull_a *)a; p != (const ull_a *)
9      ((long long unsigned)a + length); p++) {
10         n += __builtin_popcountll(*p);
11     }
12     return n;
13 }
14
15 unsigned popcount_split(unsigned* TT, long long unsigned ei,
16                         unsigned n, long long unsigned length_in_bits){
17
18     unsigned subfunction_condition = 1;
19     unsigned subs[2] = {0,0};
20     unsigned num_bytes = length_in_bits / 8 ;
21     unsigned num_chunks = length_in_bits / ei;
22     unsigned length_of_chunk = num_bytes / num_chunks;
23     unsigned balanced_sub_wt = pow(2,n-2);
24
25     for (int i = 0; i < num_chunks; i++) {
26         subs[i%2] += count(TT + i * length_of_chunk,
27                            length_of_chunk);
28     }
29     subfunction_condition = (subs[0] == balanced_sub_wt) ||
30     (subs[1] == balanced_sub_wt);
31
32     return subfunction_condition;
33 }
```

Slika 4.14: Optimizovana funkcija split.

U tabeli 4.6 prikazani su zbirovi vremena izvršavanja optimizovanih i neoptimizovanih funkcija split za svaki jedinični vektor e_i, za svako n, kao i postignuto ubrzanje.

Split funkcija	n				
	8	10	12	14	16
neoptimizovana funkcija	0.295	1.261	9.917	44.502	131.161
optimizovana funkcija	0.019	0.081	0.325	1.215	6.124
postignuto ubrzanje	15.5	15.5	30.5	36.6	21.4

Tabela 4.6: Pregled vremena izvršenja funkcije Split u ms za 10 funkcija.

.

Višenitno izvršavanje se lakše uvodi za funkciju split, s obzirom da se ne vrši nikakva modifikacija originalnog TT. Za svaki jedinični vektor e_i pokreće se odvojena nit koja izračunava težine podfunkcija određenih deljenjem TT duž pravca tog jediničnog vektora. Pamti se rezultat poređenja dobijenih težina sa težinama koje bi ispunile uslov poluuravnoteženosti. Funkcija nije poluuravnotežena ako je bilo koja nit vratila negativan rezultat. U tabeli 4.7 prikazani su zbirovi vremena izvršavanja višenitnih funkcija split za svaki jedinični vektor e_i, za svako n.

Split funkcija	n							
	18	20	22	24	26	28	30	32
opt. alg.	0.001	0.005	0.017	0.046	0.128	0.456	1.869	7.846

Tabela 4.7: Pregled izvršavanja višenitnog split koraka algoritma u s.

Glava 5

Eksperimentalni rezultati

U glavi 3 predstavljen je algoritam za generisanje bent funkcija, ali i njegova modifikacija. Što je veći broj promenljivih u Bulovoj funkciji i što je veći maksimalni stepen, razlike u vremenu izvršavanja ova dva algoritma postaju sve izraženije. U odeljku 5.1 prikazuju se rezultati dobijeni osnovnim algoritmom za $n \leq 16$, a u odeljku 5.2 rezultati dobijeni optimizovanim algoritmom za $18 \leq n \leq 32$.

5.1 Eksperimentalni rezultati za malo n

U ovom odeljku prikazuju se rezultati koji su postignuti pokretanjem ovih algoritama za $8 \leq n \leq 16$ i sa stepenom $3 \leq maxDeg \leq \frac{n}{2}$. Ovi algoritmi generišu različite bent funkcije. Iz samih rezultata se može uvideti da drugi algoritam postiže značajno ubrzanje. Zbog toga se u odeljku 5.2 koristi drugi algoritam za pronalaženje bent funkcija za $18 \leq n \leq 32$ i sa stepenom $3 \leq maxDeg \leq \frac{n}{2}$.
Ovaj metod za generisanje bent funkcija koji je opisan prethodnim algoritmom je postepeno unapređivan, kako bi se analizirale sve neophodne funkcije koje su potrebne za izvršavanje algoritma, ispitala sposobnost algoritma da generiše različite funkcije i procenila efikasnost u odnosu na vreme izvršavanja. Algoritam je pokrenut 10 puta za svaki od ulaza i srednja vrednost vremena izvršavanja je prikaza u tabeli 5.1. Program je izvršavan na računaru sa procesorom Intel(R) Core(TM) i7-1165G7 i 16GB RAM memorije. Naredna tabela predstavlja rezime podataka prikupljenih tokom izvršavanja algoritma. U tabeli su redom prikazani:

- **n** – broj ulaznih promenljivih;
- **maxDeg** – maksimalni stepen;

- **BAPU-10** (br. ANF pozicija iz uslova 10) – broj ANF pozicija koji ispunjavaju uslov 10 iz algoritma: za term A_j važi $3 \leq \text{degree}(A_j) \leq maxDeg$ i $A_j \neq 1$;
- **BAPU-13** (br. ANF pozicija iz uslova 13) – broj ANF pozicija koji ispunjavaju uslov 13 iz algoritma, odnosno nov term ne narušava uslov poluuravnoteženosti funkcije;
- **BBF** (br. bent funkcija) – ukupan broj generisanih bent funkcija koji ispunjavaju poslednji i najbitniji uslov 15 iz algoritma;
- **PVGBF** (prosečno vreme generisanja bent funk. (ms)) – prosečno potrebno vreme za generisanje jedne bent funkcije;

Za sve ulazne vrednosti i za sve stepene iz tabele pokretanjem programa generisano je više bent funkcija.

Za funkcije sa većim n kao ulaznim parametrom generiše se veći broj bent funkcija. Broj bent funkcija se takođe uvećava sa stepenom $maxDeg$. Jedino odstupanje se dešava za ulaze gde je $n \geq 12$. Najveći $maxDeg$ tada proizvodi manje bent funkcija od $maxDeg - 1$. Ovo se dešava zbog manjeg broja pojavljivanja bent funkcija stepena $\frac{n}{2}$.
Od svih ANF termova koji se razmatraju, u proseku je odbačeno oko 66% termova, zahvaljujući uslovu 13. Ovaj algoritam daje relativno veliki broj bent funkcija u relativno kratkom vremenskom roku za ulaz veličine $n \leq 16$.

n	maxDeg	BAPU-10	BAPU-13	BBF	PVGBF
8	3	177	28	12	17.42
8	4	584	92	17	22.06
10	3	517	56	22	16.72
10	4	2259	461	32	40.41
10	5	3751	514	37	54.70
12	3	1345	168	41	18.80
12	4	5088	991	70	40.35
12	5	21 194	4782	83	175.11
12	6	19 315	5218	69	224.71
14	3	3995	633	68	44.25
14	4	19 646	4893	114	150.80
14	5	59 127	14 561	147	368.30
14	6	112 107	34 067	163	978.96
14	7	140 197	33 778	155	1159.94
16	3	5762	1274	103	66.01
16	4	60 458	16 739	215	389.47
16	5	322 105	95 664	294	1143.75
16	6	208 573	64 715	339	551.25
16	7	2 324 901	769 920	346	3313.36
16	8	1 771 815	451 544	340	4917.76

Tabela 5.1: Rezultati izvršavanja prvog originalnog algoritma.

n	maxDeg	BAPU-10	BAPU-13	BBF	PVGBF
8	3	104	18	12	2.83
8	4	351	48	17	4.47
10	3	351	52	20	4.65
10	4	1785	228	33	10.18
10	5	7661	973	37	43.78
12	3	746	129	41	4.53
12	4	5541	1052	70	17.00
12	5	29 658	8103	85	85.94
12	6	28 518	8586	77	98.84
14	3	3501	470	67	12.20
14	4	45 852	14 118	116	132.25
14	5	70 267	17 666	155	112.44
14	6	103 431	25 877	166	174.43
14	7	166 657	47 830	165	305.24
16	3	3370	644	106	11.65
16	4	57 444	18 543	218	110.27
16	5	38 629	15 186	296	68.01
16	6	105 797	31 397	332	162.07
16	7	220 229	77 796	345	308.21
16	8	253 899	83 166	339	353.31

Tabela 5.2: Rezultati izvršavanja drugog modifikovanog algoritma.

5.2 Eksperimentalni rezultati za veliko n

U narednim tabelama prikazani su rezultati pokretanja programa za $18 \leq n \leq 32$ i $3 \leq maxDeg \leq \frac{n}{2}$. U tabeli 5.3 prikazani su rezultati izvršavanja algoritma u slučajevima kada je postignut uslov zaustavljanja, odnosno nije bilo moguće generisati više nijednu novu bent funkciju dodavanjem bilo kog terma na poslednju pronađenu bent funkciju. Algoritam je pokrenut deset puta za svaku kombinaciju ulaza. U tabeli 5.3 je prikazan primer jednog izvršavanja algoritma, koje je predstavljalo prosek po ukupnom broju ANF pozicija koje zadovoljavaju uslov 10, ukupnom broju ANF pozicija koje zadovoljavaju uslov 13 kao i ukupnom broju pronađenih bent funkcija. Za izabrano izvršavanje algoritma na osnovu ukupnog vremena izvršavanja i broja pronađenih bent funkcija izračunato je prosečno vreme generisanja jedne bent funkcije u ms. U tabeli 5.3 su redom prikazane kolone kao u tabelama 5.1 i 5.2.

Za neke kombinacije broja ulaza i maksimalnog stepena termova koji ulaze u bent funkciju zbog predugačkog izvršavanja algoritama nije se čekalo na uslov zaustavljanja algoritma. Stoga je za ovakve kombinacije prikazano vreme potrebno da se izgeneriše unapred zadati broj bent funkcija (početna kvadratna bent funkcija ne ulazi u zadati broj). Za ovakve instance algoritma, gde se nije čekalo na uslov zaustavljanja, pokrenut je modifikovan algoritam koji se zaustavlja nakon što pronađe unapred zadat broj bent funkcija. U tabeli 5.4 predstavljeni su podaci tih izvršavanja:

- **n** – broj ulaznih promenljivih;
- **maxDeg** – maksimalni stepen;
- **BTBF** (br. traženih bent funkcija) – unapred zadat broj traženih bent funkcija;
- **UBTKNIU-10** (ukupan br. termova koji nisu ispunili uslov 10);
- **UBTKSIU-10** (ukupan br. termova koji su ispunili uslov 10);
- **PVGBF** (prosečno vreme generisanja 1 bent funk. (s)) – algoritam je pokrenut deset puta za svaku kombinaciju ulaza i unapred zadat broj traženih bent funkcija. U tabeli 5.4 je prikazan primer jednog izvršavanja algoritma, koje

je predstavljalo prosek po ukupnom broju termova koji nisu ispunili uslov 10, ukupnom broju termova koji su ispunili uslov 10. Za izabrano izvršavanje algoritma na osnovu ukupnog vremena izvršavanja i broja pronađenih bent funkcija izračunato je prosečno vreme generisanja jedne bent funkcije u s;

n	maxDeg	BAPU-10	BAPU-13	BBF	PVGBF
18	3	4114	904	167	0.04
18	4	24 455	9692	413	0.11
18	5	86 246	30 339	638	0.25
18	6	229 776	87 834	777	0.56
18	7	562 676	189 154	837	1.22
18	8	820 169	316 767	612	2.57
18	9	1 055 213	366 210	528	3.66
20	3	4754	1195	241	0.13
20	4	37 787	13 606	619	0.44
20	5	162 175	61 929	1082	1.11
20	6	445 370	189 726	1446	1.98
20	7	1 197 005	509 253	1367	6.59
20	8	2 468 415	913 805	1778	9.95
20	9	3 903 697	1 554 752	1384	19.96
20	10	4 332 428	1 449 913	1116	27.78
22	3	8134	1458	293	0.56
22	4	58 351	20 056	802	1.84
22	5	255 554	102 742	1891	3.51
22	6	910 756	433 812	2781	9.47
22	7	2 717 122	1 160 935	3208	24.65
22	8	8 605 125	3 927 239	3129	71.77
24	3	10 985	2914	435	3.08
24	4	89 307	38 404	1312	6.82
24	5	440 473	170 820	2980	14.09
26	3	15 958	4268	475	10.91
26	4	113 196	49 102	1982	21.93
28	3	17 541	5004	734	34.45

Tabela 5.3: Rezultati generisanja bent funkcija.

n	maxDeg	BTBF	UBTKNIU-10	UBTKSIU-10	PVGBF
22	9	5	1086	272	5.5
22	10	5	1373	426	4.0
22	11	5	1202	656	6.4
24	6	5	3341	9	0.4
24	7	5	3597	39	1.4
24	8	5	3437	165	5.6
24	9	5	2491	253	9.2
24	10	5	2713	510	18.6
24	11	5	11 491	3474	138.0
24	12	5	3148	1760	64.8
26	5	5	24 936	24	3.2
26	6	5	6060	19	2.6
26	7	5	7330	35	4.8
26	8	5	8385	86	12.2
26	9	5	9193	411	67.2
26	10	5	3138	383	62.0
26	11	5	12 096	2558	380.2
26	12	5	4141	1461	203.4
26	13	5	5701	3481	2433.0
28	3	5	309 848	7	3.4
28	4	5	76 617	8	4.4
28	5	5	33 544	12	7.6
28	6	5	7426	21	12.6
28	7	5	18 629	74	43.4
28	8	5	25 824	243	139.0
28	9	5	8064	444	260.4
28	10	5	13 543	440	255.8
28	11	5	13 720	1367	828.0
28	12	5	3700	1008	584.0
28	13	5	20 788	10 462	6283.0
28	14	5	18 491	11 068	6638.0

Tabela 5.4: Rezultati generisanja bent funkcija.

n	maxDeg	BTBF	UBTKNIU-10	UBTKSIU-10	PVGBF
30	3	2	84 625	2	16.5
30	4	2	119 128	3	22.5
30	5	2	5076	2	17.0
30	6	2	24 894	2	16.5
30	7	2	16 213	10	61.5
30	8	2	11 984	4	26.0
30	9	2	684	7	49.0
30	10	2	1799	137	840.5
30	11	2	11 488	578	3616.0
30	12	2	1993	99	612.0
30	13	2	1107	419	2613.0
30	14	2	424	131	820.0
30	15	2	677	380	2419.5
32	3	1	691 460	1	61.0
32	4	1	126 872	1	69.0
32	5	1	7576	1	64.0
32	6	1	42 899	3	156.0
32	7	1	1832	1	66.0
32	8	1	24 005	9	445.0
32	9	1	3014	4	207.0
32	10	1	234	1	66.0
32	11	1	344	35	1649.0
32	12	1	497	40	2037.0
32	13	1	146	76	3435.0
32	14	1	2	2	111.0
32	15	1	162	31	1346.0
32	16	1	144	72	3339.0

Tabela 5.4: Rezultati generisanja bent funkcija.

U prethodnim tabelama se vidi da se sa povećanjem maksimalnog stepena termova koji mogu biti dodati u funkciju opada broj odbačenih termova u uslovu 10. Razlog tome je što u razmatranje ulazi veći broj termova, ali odatle se ne vidi odmah zbog čega zajedno sa maksimalnim stepenom raste i broj pregledanih termova koji su prošli uslov 10 pre nego što se izgeneriše unapred zadati broj bent funkcija.

Ako grupišemo na osnovu stepena termove koji su prošli uslov 10, ali nisu proizveli novu bent funkciju, možemo jasnije videti da termovi većeg stepena, iako su zastupljeniji u ANF Bulovih funkcija, imaju manju verovatnoću da njihovim dodavanjem proizvedu novu bent funkciju.

degree	**Br. termova dodatih u ANF**	**Br. termova koji nisu ušli u ANF**
3	0	0
4	0	1
5	1	5
6	1	25
7	1	64
8	1	212
9	0	500
10	1	882
11	0	1574
12	0	2354
13	0	2680
14	0	2766

Tabela 5.5: Analiza prihvaćenih i odbačenih termova prilikom traženja 5 bent funkcija za $n = 28$ $maxDeg = 14$.

Glava 6

Zaključak i dalji razvoj

Bent funkcije su funkcije Bulove algebre koje poseduju svojstvo maksimalne nelinearnosti. Takve funkcije su od velike važnosti u domenu kriptografije. Svojstvo nelinearnosti proverava se pomoću WHT, koji je računarski zahtevna procedura za Bulove funkcije sa dovoljno velikim brojem ulaza. Algoritam koji je implementiran u ovom radu koristi i druga bitna svojstva Bulovih funkcija, kako bi smanjio broj funkcija na koje se primenjuje WHT prilikom traganja za bent funkcijama.

U ovom radu se istražuje primena optimizacija na procedure iz algoritma predstavljenog u [4] i korišćenje modernih računarskih sistema za ubrzavanje generisanja bent funkcija za iste ulazne parametre kao u disertaciji. Demonstrirano je primetno ubrzanje, a onda se na osnovu njega istražuje kako ono utiče na proširenje domena pretrage za bent funkcijama u odnosu na domen pretrage iz disertacije.

Verifikovana je implementacija postizanjem rezultata iz izvorne disertacije [4] i generisane su bent funkcije za sve ulaze od $n = 8$ do $n = 16$. Zatim su glavne procedure optimizovane da bi moglo da se pristupi generisanju bent funkcija sa više od 16 promenljivih. Transformacija ANF je unapređena tako što se koriste procesorske reči od 64 bita, vektorske operacije nad 64 elementa u kombinaciji sa odmotavanjem petlji. Određivanje težine funkcije i njenih podfunkcija je unapređeno korišćenjem specifične mikroinstrukcije procesorske arhitekture x86 i višenitnim izvršavanjem. WHT je takođe unapređena višenitnim izvršavanjem. Korišćenjem ovih unapređenja uspešno su generisane bent funkcije za sve ulaze $16 \leq n \leq 32$. Za generisanje bent funkcija malog stepena izvršavanje algoritma ne traje dugo. Prosečno vreme potrebno da se generiše jedna bent funkcija stepena 3 se kreće od 0.04 sekunde do 61 sekunde u zavisnosti od broja ulaza. Ako se traže bent funkcije

većeg stepena ili bent funkcije veće algebarske težine, onda izvršavanje algoritma zahteva znatno više vremena. To se može ilustrovati primerima tabele 5.4 gde je za $n = 28$, $maxDeg = 14$ u proseku trebalo 6638.0 sekundi da bi se generisala jedna bent funkcija.

Treba napomenuti da se ne zna da li među bent funkcijama koje generiše ovaj algoritam postoje funkcije koje pripadaju istoj klasi ekvivalencije. Za dve funkcije se kaže da pripadaju istoj klasi ekvivalencije ako postoji linearna transformacija, definisana kao u [4], tako da se jedna funkcija može izraziti linearnom transformacijom druge. Zato bi bilo korisno sa stanovišta kriptografije da se takođe izvrši i analiza generisanih funkcija i utvrdi pripadnost klasama ekvivalencije.

Dalji razvoj ovog heurističkog algoritma ima matematičku i računarsku perspektivu. Razvoj iz matematičke je uslovljen pronalaženjem nekog svojstva bent funkcija kojim bi se dalje umanjio broj potrebnih izvršavanja WHT da se pronađe nova bent funkcija. Iz računarske perspektive razvoj zavisi od daljeg ubrzanja postojećih procedura. Pod istim uslovima pod kojima je primenjeno višenitno izvršavanje, moglo bi se primeniti paralelno izračunavanje na procesorima koji podržavaju SIMD. Primer za to bi bilo korišćenje grafičkih karti uz pomoć platformi OpenCL i CUDA. Jedan primer paralelizacije se sreće u radu [2], gde je primenjena na proceduru WHT, ali za Bulove funkcije sa manje od 20 promenljivih. Stoga ostaje otvoreno pitanje uspešne primene paralelizacije na procedure opisane u ovom radu.

Bibliografija

[1] Valentin Bakoev. Fast computing the algebraic degree of boolean functions. In *International Conference on Algebraic Informatics*, pages 50–63. Springer, 2019.

[2] Dusan Bikov and Ilija Bouyukliev. Parallel fast walsh transform algorithm and its implementation with cuda on gpus. *Cybernetics and Information Technologies*, 18(5):21–43, 2018.

[3] John Francis Dillon. *Elementary Hadamard difference-sets.* University of Maryland, College Park, 1974.

[4] Joanne Elizabeth Fuller. *Analysis of affine equivalent boolean functions for cryptography.* PhD thesis, Queensland University of Technology, 2003.

[5] Willi Meier and Othmar Staffelbach. Nonlinearity criteria for cryptographic functions. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 549–562. Springer, 1989.

[6] Oscar S Rothaus. On "bent" functions. *Journal of Combinatorial Theory, Series A*, 20(3):300–305, 1976.

[7] Kathrin Schacke. On the kronecker product. *Master's thesis, University of Waterloo*, 2004.

[8] Natalia Tokareva et al. *Bent Functions.* Elsevier, 2007.