Diferencijalna kriptoanaliza

Uršan, Josip

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

DIFERENCIJALNA KRIPTOANALIZA

Diplomski rad

Josip Uršan

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 30.08.2021.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime studenta:	Josip Uršan
Studij, smjer:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D-1276, 06.10.2019.
OIB studenta:	86022009716
Mentor:	Izv. prof. dr. sc. Krešimir Grgić
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Prof.dr.sc. Drago Žagar
Član Povjerenstva 1:	Izv. prof. dr. sc. Krešimir Grgić
Član Povjerenstva 2:	Mr.sc. Anđelko Lišnjić
Naslov diplomskog rada:	Diferencijalna kriptoanaliza
Znanstvena grana rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Zadatak diplomskog rada:	Diferencijalna kriptoanaliza je kriptoanalitička metoda koja analizira razlike između parova otvorenog teksta i šifrata (statistička analiza), s ciljem pronalaženja enkripcijskog ključa. Potrebno je analizirati i objasniti različite pristupe problematici diferencijalne kriptoanalize, te ih ilustrirati kroz različite praktične primjere kriptografskih napada tehnikama diferencijalne kriptoanalize. (Student: Josip Uršan)
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	30.08.2021.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 15.09.2021.

Ime i prezime studenta:	Josip Uršan
Studij:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D-1276, 06.10.2019.
Turnitin podudaranje [%]:	8%

Ovom izjavom izjavljujem da je rad pod nazivom**: Diferencijalna kriptoanaliza**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Krešimir Grgić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD

Kriptoanaliza jedno je od temeljnih područja informacijske znanosti i kriptografije, za čiju se pojavu može reći da je paralelna s pojavom kriptografije. Kriptoanaliza jednostavnih, starijih, šifri može se provesti ručno. Moderna povijest, te posljedično i temelji, moderne kriptografije i kriptoanalize postavljeni su u Prvom i Drugom svjetskom ratu. Ova dva vremenska perioda su obilovala raznim uređajima za šifriranje, poput Enigme, Lacida uređaja, SIGABA, te različitim šiframa poput JN-25, Lorentz šifre, Siemens & Halske T52, i mnogim drugim.

Diferencijalna kriptoanaliza danas se razvila u niz podvrsta, poput diferencijalne kriptoanalize višeg reda, no za potrebe diplomskog rada korišteno je nekoliko temeljnih radova diferencijalne kriptoanalize, od kojih valja istaknuti [1][2][4][6].

Cilj diplomskog rada je implementirati temeljni napad diferencijalnom kriptoanalizom koji omogućuje dohvaćanje potključeva. Također je cilj isprobati utjecaj različitih operacija nad S-kutijama na distribuciju diferencija.

U drugom poglavlju predstavljena je temeljna ideja diferencijalne kriptoanalize, te svi potrebni pojmovi, a u trećem poglavlju je kroz nekoliko potpoglavlja predstavljena korištena šifra, te je kroz izdvojeni primjer objašnjena implementacija napada. U četvrtom poglavlju testiran je utjecaj nekoliko različitih operacija na distribuciju diferencija korištene S-kutije.

2. DIFERENCIJALNA KRIPTOANALIZA

Diferencijalna kriptoanaliza predstavlja jedno od najvažnijih dostignuća moderne kriptoanalize. Radi se o metodi napada simetričnih kriptosustava. Iako je ova metoda javnosti otkrivena krajem 80-ih, odnosno početkom 1990-ih, kroz rad [1], nekoliko godina kasnije se kroz [2] članak Dona Coppersmitha, jednog od tvoraca DES kriptosustava, pokazalo da je još u 1970-im godinama IBM-ov kriptografski odjel bio svjestan sposobnosti diferencijalne kriptoanalize, zbog čega su S-kutije DES-a dizajnirane kako bi bile iznimno otporne na napad diferencijalnom kriptoanalizom. Usporedbe radi, FEAL obitelj kriptografskih algoritama razvijena je krajem 1980-ih kao zamjena za DES, te je imala 64-bitni ključ kada je u pitanju FEAL-4 verzija, te 128-bitni ključ za FEAL-N/NX verziju. Kako je FEAL razvijen prije nego što je diferencijalna kriptoanaliza bila poznata, nije bio dovoljno otporan na ovaj napad, te je u neku ruku postao zlatni standard za testiranje kriptoanalitičkih metoda zbog svojih slabosti.

U ovom poglavlju predstavljena je temeljna ideja napada diferencijalnom kriptoanalizom, te je kroz kratak primjer na SPN[1]-u predstavljan tijek napada.

2.1. Supstitucijsko-permutacijska mreža (SPN)

Supstitucijsko-permutacijske mreže su mreže sastavljen od određenog broja rundi, pri čemu se svaka runda sastoji od tri operacije :

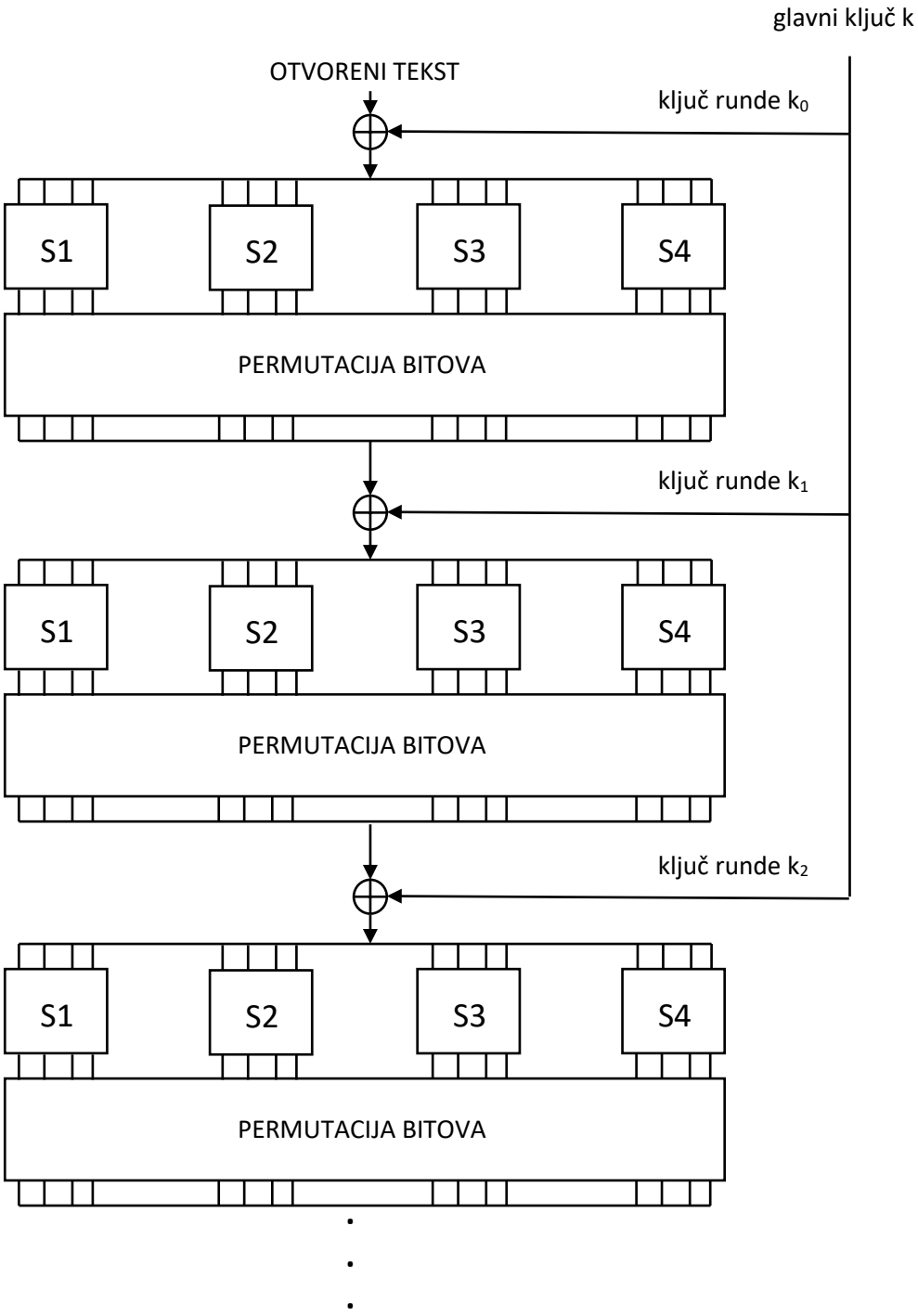
- supstitucija
- permutacija
- miješanje s ključem

Iako supstitucijsko-permutacijske mreže ostavljaju dojam da se zapravo radi o mrežama koje su jednake Feistelovoj mreži, temeljna razlika nastupa u načinu obrade blokova. Supstitucijsko-permutacijske mreže dijeli ulaznih *n* bitova na *s* blokova bitova, pri čemu svaki blok ima $\frac{n}{s}$ bitova. Vrijednost *s* zapravo predstavlja broj S-kutija, od kojih svaka prima $\frac{n}{s}$ bitova. Feistelova mreža ulazni

[1] Substitution Permutation Network – supstitucijsko permutacijska mreža

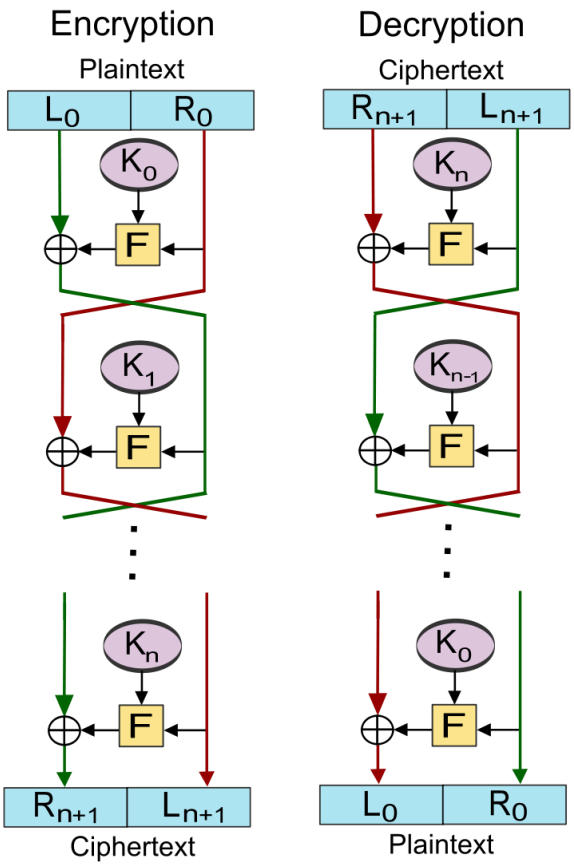
skup bitova dijeli samo na lijevu i desnu polovicu, koje kroz svaku rundu stupaju u interakciju pomoću radnji koje se obavljaju u svakoj rundi.

Primjer supstitucijsko permutacijske mreže nalazi se na slici 2.1.



Slika 2.1. – prikaz supstitucijsko-permutacijske mreže

Na slici 2.2. prikazana je Feistelova mreža.



Slika 2.2. – prikaz Feistelove mreže za proces enkripcije i dekripcije [11]

Supstitucijsko-permutacijska mreža sa slike 2.1., koja će se koristiti za demonstraciju diferencijalne kriptoanalize, prima otvoreni tekst duljine 16 bita, te ga dijeli na 4 skupa od po 4 bita. Svaki skup od 4 bita ulazi u jednu S-kutiju, gdje je izložen procesu supstitucije. Proces supstitucije jednak je za svaku S-kutiju, te je dan tablicom 2.1.

Ulaz	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Izlaz	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

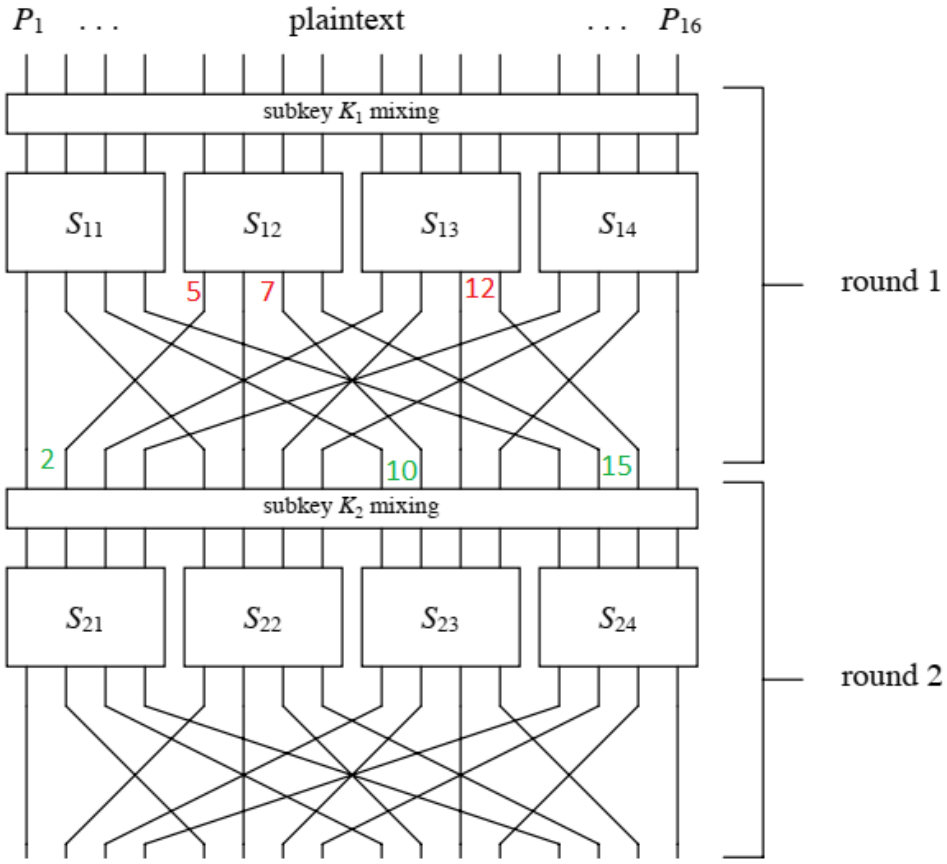
Tablica 2.1. – mapiranje ulaza na izlaze S-kutija supstitucijsko-permutacijske mreže sa slike 2.1.

Permutacija, odnosno miješanje bitova, također se obavlja prema fiksnoj tablici. Redak *ulaz* u tablici 2.2. odnosi se na ulazne pozicije S-kutija iz iduće runde, dok se redak *izlaz* odnosi na izlazne pozicije, odnosno bitove, S-kutija iz prethodne runde.

Ulaz	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Izlaz	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

Tablica 2.2. – mapiranje procesa permutacije supstitucijsko-permutacijske mreže sa slike 2.1.

Na slici 2.3. dan je primjer mapiranja procesa permutacije prema tablici 2.2. Zeleno su označene ulazne vrijednosti, odnosno ulazi u S-kutije iduće runde. Crveno su označeni izlazne vrijednosti, odnosno izlazi S-kutija prethodne runde.



Slika 2.3. – nekoliko primjera mapiranja procesa permutacije [4, str. 4.]

Miješanje s potključevima podrazumijeva proces XOR[2] zbrajanja rezultata supstitucije i permutacije pojedine runde s ključem runde, koji se izračunava iz glavnog ključa. Miješanje s potključevima zapravo i nije bitno kada je u pitanju diferencijalna kriptoanaliza. U idućem potpoglavlju nalazi se dokaz, no radi cjelovitosti rješenja se pretpostavlja da se potključevi generiraju bez međusobne ovisnosti, te da ne postoji veza između njih.

2.2. Općeniti napad diferencijalnom kriptoanalizom

Diferencijalna kriptoanaliza iskorištava statističku povezanost ulazne diferencije s izlaznom diferencijom, tj. diferencijom koja je ulaz u posljednju rundu šifre. Pod pojmom diferencije se podrazumijeva vrijednost koja se dobiva XOR operacijom nad parom ulaznih vrijednosti ili parom izlaznih vrijednosti.

Za nekakav ulazni skup vrijednosti $X = [X_1, X_2, X_3, \dots, X_n]$ se za svaki odabrani par ulaznih vrijednosti definira ulazna diferencija kao isključivo ili operacija između n-bitnih vrijednosti.

$$\Delta X = X' \oplus X'' \qquad (2.1.)$$

U jednadžbi 2.1. oznaka ΔX predstavlja rezultantnu diferenciju, a oznake X' i X'' predstavljaju odabrane ulazne vektore.

Ista logika primjenjuje se za izlazna diferencija, uz razliku da se oznaka Y koristi za izlazne vrijednosti, te da su Y' i Y'' vrijednosti dobivene šifriranjem vrijednosti X' i X''.

Statistička povezanost ulaznih i izlaznih diferencijal iskorištava se jer elementi kriptografskih algoritama zaduženi za sigurnost najčešće nemaju dovoljno visok stupanj nelinearnosti, ili nisu dovoljno dobro konstruirani. Diferencijalna kriptoanaliza iskorištava povećanu vjerojatnost pojave određene diferencije ΔY na izlazu za odabranu diferenciju ΔX na ulazu. Pod povećanom vjerojatnošću pojave se podrazumijeva sve vjerojatnosti pojave znatno veće od $\frac{1}{2^n}$.

Diferencijalna kriptoanaliza je napad izabranim otvorenim tekstom (*chosen plaintext attack*). Radi se o napadu gdje napadač ima mogućnost odabira željenih otvorenih tekstova koje predaje kao ulaz u kriptografski algoritam, kojeg percipira kao crnu kutiju, te na izlazu dobiva šifrate.

[2] XOR – eXclusive OR (isključivo ili operacija)

Neovisno o algoritmu koji se napada, diferencijalna kriptoanaliza ima niz istih koraka koji imaju svoj slijed. Napadač prolazi sve moguće vrijednosti ΔX, te za svaku od njih stvara listu vrijednosti X' i X'' čiji XOR rezultat daje vrijednost ΔX. Za svaku vrijednost X' i X'' se šifriranjem dobivaju odgovarajuće vrijednosti Y' i Y''. Na temelju vrijednosti Y' i Y'', te uočenih ulazno-izlaznih odnosa, napadač pokušava pogoditi ključ posljednje runde. S pretpostavljenim ključem napadač dekriptira posljednju rundu, te provjerava podudara li se dekriptirana vrijednost s vrijednošću koja se očekuje prema prethodno utvrđenim vjerojatnostima. Procesi poput traženja ključa i utvrđivanja ulazno-izlaznih odnosa obavljaju se iterativno, odnosno za sve moguće vrijednosti jer je potrebno pronaći nelinearnosti algoritma [10, str. 73. – 78.].

2.2.1. Analiza elemenata šifre

Prvi i temeljni korak diferencijalne kriptoanalize jest analiza S-kutija. S-kutije su jedan od najvažnijih elemenata svakog kriptografskog algoritma. Cilj S-kutija je učiniti odnos između ključa i šifrata vrlo nejasnim i međusobno nepovezanim jer je šifrat posljedica linearne operacije (XOR) nad otvorenim tekstom i ključem za šifriranje. Zapravo se radi o uvođenju nelinearnosti u linearni rezultat.

Analiza je provedena za S-kutiju opisanu tablicom 2.1.

Neka su nasumično uzete nekakve dvije vrijednosti, X' i X'', te neka je XOR operacija ovih vrijednosti rezultirala vrijednošću $\Delta X = 1011$. Cilj je pronaći sve moguće parove ulaznih vrijednosti čija XOR vrijednost rezultira vrijednošću ΔX. Navedeno se može postići jednostavnim iscrpnom pretragom svih mogućih kombinacija, ili odabirom samo vrijednosti X', te izračunom vrijednosti X'' pomoću izraza $X' \oplus \Delta X = X''$. Ovaj cijeli proces može se izbjeći pomoću nekakvog računalnog programa uporabom upravo iscrpne pretrage, ali ovakav opis koristan je za razumijevanje.

Za svaku odabranu vrijednost X' i X'' izračunavaju se vrijednosti Y' i Y'' na temelju tablice 2.1. U tablici 2.3. dani su rezultati ovog postupka.

Prvi stupac predstavlja XOR vrijednost koju se želi dobiti pomoću X' i X'', drugi stupac predstavlja vrijednost X' koja se slobodno odabire, treći stupac predstavlja vrijednost X'' koja se izračunava na temelju X' i ΔX. Četvrti stupac predstavlja vrijednost Y' koja se izračunava pomoću tablice 2.1., te vrijednosti X'. Peti stupac odnosi se na vrijednost Y'' koja se također izračunava pomoću tablice 2.1., te pomoću vrijednosti X''. Posljednji stupac predstavlja XOR rezultat vrijednosti Y' i Y''.

ΔX	X'	$X'' = X' \oplus \Delta X$	Y'	Y''	ΔY
1011	0000	1011	1110	1100	0010
1011	0001	1010	0100	0110	0010
1011	0010	1001	1101	1010	0111
1011	0011	1000	0001	0011	0010
1011	0100	1111	0010	0111	0101
1011	0101	1110	1111	0000	1111
1011	0110	1101	1011	1001	0010
1011	0111	1100	1000	0101	1101
1011	1000	0011	0011	0001	0010
1011	1001	0010	1010	1101	0111
1011	1010	0001	0110	0100	0010
1011	1011	0000	1100	1110	0010
1011	1100	0111	0101	1000	1101
1011	1101	0110	1001	1011	0010
1011	1110	0101	0000	1111	1111
1011	1111	0100	0111	0010	0101

Tablica 2.3. – popis svih ulaznih parova vrijednosti koji rezultiraju sa ΔX, te svih parova vrijednosti na izlazu iz S-kutije [6, str. 90.]

Sada se na temelju vrijednosti ulaznog diferencijala i izračunatih izlaznih diferencijala može uspostaviti odnos između ulazne i izlazne vrijednosti, odnosno utvrditi učestalost pojave određenih izlaznih vrijednost za određene ulazne vrijednosti.

U tablici 2.4. dana je učestalost pojave pojedinih izlaznih diferencijala.

Provođenjem netom opisanog postupka za sve moguće kombinacije ulaznog diferencijala i izlaznog diferencijala se dobiva tablica 2.5., koja sadrži učestalost pojave svih mogućih izlaznih diferencijala za sve moguće ulazne diferencijale. Vrijednosti ulaznih i izlaznih diferencijala dane su u heksadekadskom zapisu zbog preglednosti.

U tablici 2.5. može se primijetiti kako se itekako radi o S-kutiji koja je nejednolika. Postoji velik broj kombinacija ulaznih i izlaznih diferencijala koje se nikad ne pojavljuju, dok se preostale pojavljuju po 2 i više puta. Specifičan je slučaj $\Delta X = 0000$ i $\Delta Y = 0000$, koji se pojavljuje 16 puta, odnosno svaki puta kada je ulazni diferencijal 0, izlazni diferencijal je također 0. Ovo je očekivano zbog jedan na jedan mapiranja S-kutija [4][5][10, str 73. – 38.].

ΔY	Učestalost pojave
0000	0
0001	0
0010	8
0011	0
0100	0
0101	2
0110	0
0111	2
1000	0
1001	0
1010	0
1011	0
1100	0
1101	2
1110	0
1111	2

Tablica 2.4. – učestalost pojave izlaznih diferencijala za $\Delta X = 1011$

	ΔY															
ΔX	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	2	0	0	0	2	0	2	4	0	4	2	0	0
2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

Tablica 2.5. – tablica distribucije diferencijala

2.2.2. Utjecaj ključa

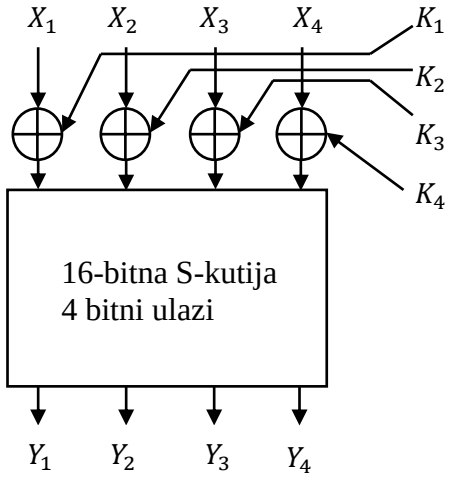
Ključ svake runde se miješa u rezultat permutacije i supstitucije pomoću XOR operacije. Pretpostavimo da je na neku vrijednost dobivenu supstitucijom i permutacijom primijenjen ključ runde, što rezultira vrijednošću $W = [W_1, W_2, W_3, W_4]$. Vrijednost W predstavlja zapravo 16-bitni vektor sastavljen od 4 podskupa po 4 bita, odnosno W_1, W_2, W_3, i W_4. Obzirom da se kombiniranje s ključem runde obavlja za svaki otvoreni tekst te za svaku rundu, imat ćemo cijeli niz W vrijednosti. Tada se može izračunati diferencija šifriranih vrijednosti.

$$\Delta W = [W_1' \oplus W_1'', W_2' \oplus W_2'', \dots] \qquad (2.2.)$$
[4, str 22.]

Raspisivanjem jednadžbe 2.2. dobiva se jednadžba 2.3. iz koje je vidljivo da ključ runde ne igra ulogu, te da se diferencijal šifrirane vrijednosti i diferencijal ulaza mogu izjednačiti.

$$\Delta W_i = W_i' \oplus W_i'' = (X_i' \oplus K_i) \oplus (X_i'' \oplus K_i) = X_i' \oplus X_i'' = \Delta X \qquad (2.3.)$$
[4, str 22.]

Postupak je prikazan na slici 2.4. za 16-bitni ulazni vektor.



Slika 2.4. – shema procesa kombiniranja ključeva rundi i ulaza

2.2.3. Propagacijski odnos

Propagacijski odnos je izraz koji iskazuje vjerojatnost da nekakav ulazni diferencijal daje određeni izlazni diferencijal, odnosno radi se o uvjetnoj vjerojatnosti. Promatrani ulazni i izlazni diferencijali nazivaju se jednostavno diferencijal. Propagacijski odnos za nekakav diferencijal (a', b') matematički se definira jednadžbom 2.4.

$$R_p(a', b') = \frac{N_D(a',\ b')}{2^n} \qquad (2.4.)$$

[6, str. 91]

Funkcija N_D odnosi se na frekvenciju pojavljivanja izlaznog diferencijala b' za ulazni diferencijal a', te se radi o vrijednosti koja se dobiva križanjem odgovarajućeg retka i stupca u tablici 2.5.

Nazivnik razlomka odnosi se na broj mogućih ulaznih vrijednosti [6, str. 91.-92.].

2.2.4. Diferencijalni trag

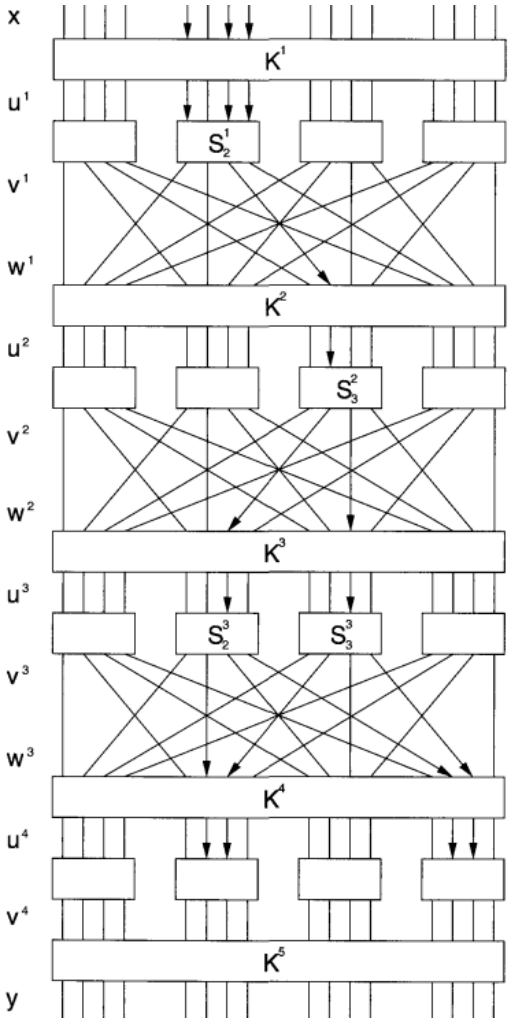
Pretpostavimo da smo pronašli propagacijske odnose diferencijala uzastopnih rundi koji su izlaz runde R, a ulaz runde R+1. Tada se pronađeni diferencijali, odnosno njihovi propagacijski odnosi, mogu kombinirati kako bi se pronašao diferencijalni trag. Diferencijalni trag predstavlja putanju kojom se prolazi kroz mrežu uz određeni ulazni diferencijal.

Za inicijalni ulazni diferencijal, a', tj. ΔX, te uzimajući u obzir supstitucijsku karakteristiku S-kutije svake runde, pronalazi se putanja koja je prikazana na slici 2.5.

Važnost diferencijalnog traga leži u činjenici da, ovisno o karakteristici S-kutija, ulazni diferencijali dovode do aktivacije određenih S-kutija, te određenih bitova svake S-kutije. Svaka S-kutija čiji je ulaz ili izlaz ne-nul naziva se aktivnom.

Za oznaku S-kutija svake runde koristi se notacija S_x^i, pri čemu index *x* predstavlja x-tu S-kutiju u *i*-toj rundi.

Na slici 2.5. vidi se da je ulazni diferencija u S_2^1 vrijednost 1011 ($B_{(16)}$). Iz tablice 2.5. može se zaključiti da je za ulazna diferencija 1011 najvjerojatnija izlazna vrijednost S-kutije 0010 ($2_{(16)}$), što je vidljivo na slici 2.5. Propagacijski odnos za $\Delta X = 1011$ i $\Delta Y = 0010$ je $\frac{8}{16}$, odnosno $\frac{1}{2}$.



Slika 2.5. – diferencijalni trag kroz mrežu

Izlaz prve runde i ulaz druge runde spojeni su tako da 3. bit izlaza S_2^1 ulazi u S_3^2 na mjesto 2. bita. Pozicije bitova promatrane su od najviše značajnog bita.

Za $\Delta X = 0100$ u tablici 2.5. pronalazi se da je najvjerojatnija pojava vrijednosti 0110 na izlazu S_3^2. Propagacijski odnos za $\Delta X = 0100$ i $\Delta Y = 0110$ je $\frac{6}{16}$, odnosno $\frac{3}{8}$.

Isti postupak ponavlja se za preostale runde, te se dolazi do propagacijskih odnosa u tablici 2.6.

		R_p
S_x^i	S_2^1	$R_p(1011, 0010) = \frac{8}{16} = \frac{1}{2}$
	S_3^2	$R_p(0100, 0110) = \frac{6}{16} = \frac{3}{8}$
	S_2^3	$R_p(0010, 0101) = \frac{6}{16} = \frac{3}{8}$
	S_3^3	$R_p(0010, 0101) = \frac{6}{16} = \frac{3}{8}$

Tablica 2.6. – propagacijski odnosi za mrežu sa slike 2.5.

Pretpostavlja se da su propagacijski odnosi međusobni neovisni. Međusobnim množenjem propagacijskih odnosa u tablici 2.6. dobiva se propagacijski odnos za prve tri runde supstitucijsko-permutacijske mreže (jednadžba 2.5.).

Globalno promatrani ulaz je 0000 1011 0000 0000, a izlaz nakon treće runde je 0000 0101 0101 0000.

$$R_p(0000\ 1011\ 0000\ 0000, 0000\ 0101\ 0101\ 0000) = \frac{1}{2} \cdot \frac{3}{8} \cdot \frac{3}{8} \cdot \frac{3}{8} = \frac{27}{1024} \quad \text{(2.5.) [6, str. 92.]}$$

Obzirom da je u potpoglavlju 2.2.2. objašnjeno da ključ ne utječe na vrijednost diferencijala, može se zaključiti da je izlaz treće runde ulaz u četvrtu rundu. Na slici 2.5. možemos povezati izlaznu vrijednost treće runde, O_3, sa ulazom u četvrtu rundu, I_4.

$$O_3 = 0000\ 0101\ 0101\ 0000 \quad \text{(2.6.) [6, str. 92.]}$$

Način povezivanja treće i četvrte runde, odnosno permutacija izlaznih vrijednosti treće runde, daje nam ulaznu vrijednost četvrte runde.

$$I_4 = 0000\ 0110\ 0000\ 0110 \quad \text{(2.7.) [6, str. 92.]}$$

Na temelju iznad opisanog se može zaključiti da se ulazna vrijednost četvrte runde $I_4 = 0000\ 0110\ 0000\ 0110$ javlja s vjerojatnošću $\frac{27}{1024}$ za ulazni diferencijal $\Delta X = 0000\ 1011\ 0000\ 0000$ [4][5][6, str. 92.].

Ovakva analiza obavlja se za sve moguće ulazne i izlazne diferencije.

2.2.5 Napad i dešifriranje šifrata

Radnje opisane u prethodnim poglavljima obavljaju se kako bi se pronašle sakrivene statističke veze između ulaza i izlaza. Nakon što je taj proces obavljen, s obzirom na ulaznu vrijednost u algoritam šifriranja, te ulaznu vrijednost u posljednje S-kutije, te na temelju utvrđenih ulazno-izlaznih odnosa, se pretpostavljaju bitovi ključa. Pretpostavljene vrijednosti ključa se miješaju sa šifratom i guraju unazad kroz S-kutije, te se pokušava pronaći kombinacija bitova koja dovodi do podudaranja s izlaznom diferencijom.

Može se postaviti pitanje zašto diferencijalna i linearna kriptoanaliza idu samo do R-1 runde, kada i posljednja runda obavlja supstituciju, permutaciju i miješanje s potključem. Diferencijalna kriptoanaliza se može promatrati kao metoda koja je sastavljena od dva dijela : *razlikovni*[3] napad i napada otkrivanja ključa. *Razlikovni* dio napada utvrđuje odnos između ulaza i vrijednosti nakon R-1 runde. Napad otkrivanja ključa tada obuhvaća samo pokušaj otkrivanja potključa posljednje runde. Pretpostavljene potključeve posljednje runde se gura unazad kroz posljednju S-kutiju, te se ta vrijednost uspoređuje s onom koja se očekuje na temelju statističke analize algoritma [3].

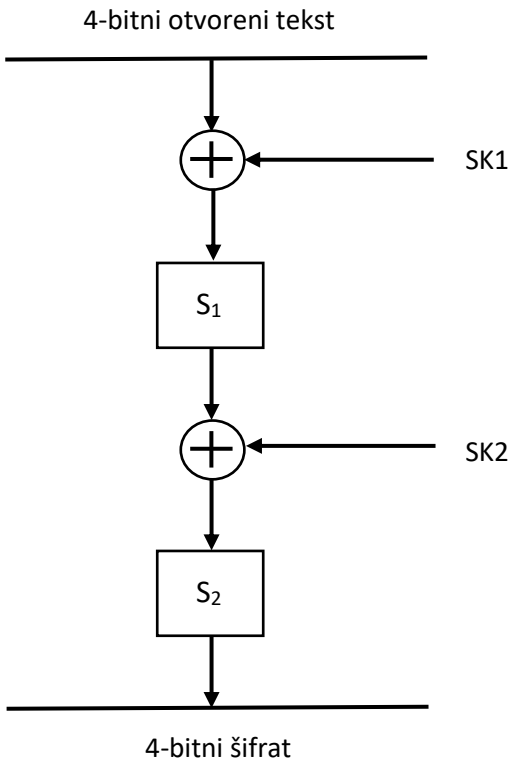
[3] Distinguisher napad

3. IMPLEMENTACIJA NAPADA

U ovom poglavlju opisan je tijek napada na korištenom algoritmu za kriptiranje. Predstavljen je korišteni kriptografski algoritam, svojstva kriptografskog algoritma, te rezultati analize algoritma koji prethode napadu diferencijalnom kriptoanalizom.

3.1. Korišteni kriptografski algoritam – igračka šifra

Za potrebe demonstracije diferencijalne kriptoanalize korišten je algoritam često poznat kao igračka šifra (*toy cipher*). Algoritam je ovaj naziv poprimio zbog svoje jednostavnosti. Zapravo se ne radi o nekakvom specifičnom algoritmu s točno specificiranim brojem rundi i komponenti, već o općenitom nazivu koji pokriva niz jednostavnih šifri koje implementiraju neke, ili sve, mogućnosti supstitucijsko permutacijskih mreža. Igračka šifra implementira se po volji osobe koja ju implementira. Korištena šifra prikazana je na slici 3.1.



Slika 3.1. – korišteni algoritam šifriranja

Šifra sa slike 3.1. prima 4-bitni otvoreni tekst kao ulaz, a kao izlaz daje 4-bitni šifrat. Postoje dvije runde, od kojih se svaka sastoji od jedne XOR operacije s odgovarajućim potključem, te primjene jedne S-kutije u svakoj rundi na rezultat prethodno navedene XOR operacije. Za generiranje potključeva SK_1 i SK_2 koristi se glavni ključ duljine 6 bita. Za generiranje potključeva se koristi shema navedena u izrazu 3.1. za prvi potključ, odnosno u izrazu 3.2. za drugi potključ.

$$SK_1 = [3, 5, 0, 2] \quad (3.1.)$$

$$SK_2 = [4, 2, 3, 1] \quad (3.2.)$$

Elementi navedeni u listama SK_1 i SK_2 predstavljaju indekse elemenata koji se preuzimaju iz glavnog ključa, čime se formira pojedini potključ. Niže je naveden primjer (primjer 3.1.) formiranja potključeva na temelju danog glavnog ključa.

Primjer 3.1. – formiranje potključeva

glavniKljuč	1	0	1	1	1	0
indeksElementa	0.	1.	2.	3.	4.	5.

1. potključ (SK1)	1	0	1	1
indeksElementaIzGlavnogKljuča	3.	5.	0.	2.

2. potključ (SK2)	1	1	1	0
indeksElementaIzGlavnogKljuča	4.	2.	3.	1.

Prvi potključ koristi se za XOR operaciju s ulaznim otvorenim tekstom, dok se drugi potključ korist za XOR operaciju s izlaznom vrijednošću prve S-kutije.

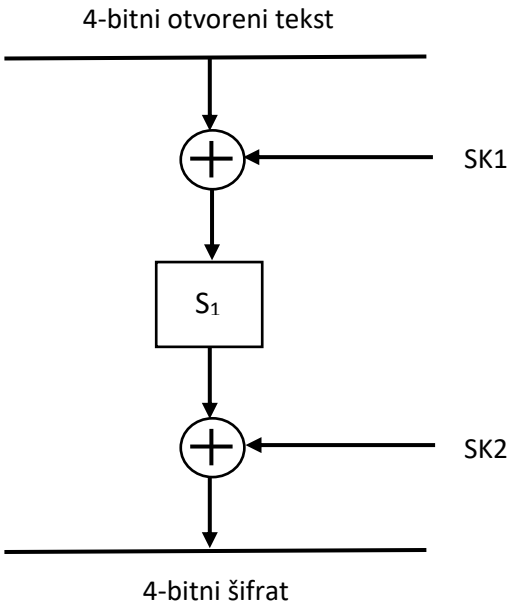
Kako je i prije navedeno, koriste se dvije S-kutije. Jedna se nalazi u svakoj rundi, a njihova konfiguracija, koja je navedena u tablici 3.1., ista je za obje S-kutije. Sve vrijednosti u tablici 3.1. su u dekadskom brojevnom sustavu.

S_1/S_2 konfiguracija	
Ulazna vrijednost	Izlazna vrijednost
0	3
1	14
2	1
3	10
4	4
5	9
6	5
7	6
8	8
9	11
10	15
11	2
12	13
13	12
14	0
15	7

Tablica 3.1. – konfiguracija S_1 i S_2 S-kutija

3.2. Analiza elemenata algoritma šifriranja

Prvi korak diferencijalne kriptoanalize jest analiza svojstava šifre, odnosno elemenata šifre kod kojih se može utvrditi nekakav uzorak. Analiza se obavlja na S-kutijama, odnosno u ovom slučaju samo nad S_1 kutijom obzirom da je cilj utvrditi ulaz u posljednji element posljednje runde. Obzirom da je napadnuti algoritam poznat napadaču prilikom primjene diferencijalne kriptoanalize, S_2 kutija postaje nevažna jer je napadač svjestan njezinog mapiranja, te jednostavno može obrnuti izlaze iz S_2 kutije kako bi utvrdio točne ulazne vrijednosti u S_2 kutiju. Ovim pojednostavljenjem shema šifre poprima strukturu prikazanu na slici 3.2.



Slika 3.2. – pojednostavljenje korištenog algoritma šifriranja uklanjanjem S_2

Analiza ulazno-izlaznih svojstava, odnosno vjerojatnost pojave određene izlazne diferencije za promatranu ulaznu diferenciju dana je u tablici 3.2. za S-kutiju S_1.

		ΔY															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ΔX	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	2	0	4	0	0	0	2	0	0	0	2	0	6	0	0
	2	0	2	2	0	2	0	0	2	0	2	0	2	0	2	0	2
	3	0	0	2	0	2	0	0	0	0	2	4	0	4	0	0	2
	4	0	0	0	0	2	4	0	6	0	0	0	0	2	0	0	2
	5	0	0	2	0	2	0	2	2	2	0	4	0	0	0	0	2
	6	0	0	2	2	0	2	2	0	4	0	0	0	2	0	2	0
	7	0	0	0	2	0	2	0	0	2	0	0	4	0	0	2	4
	8	0	2	0	0	0	6	0	0	2	2	0	2	0	0	2	0
	9	0	0	2	2	2	2	4	0	4	0	0	0	0	0	0	0
	10	0	2	0	0	2	0	0	0	2	2	2	0	4	0	2	0
	11	0	4	2	2	0	0	0	0	0	4	2	2	0	0	0	0
	12	0	2	4	0	2	0	0	0	0	0	2	0	2	2	2	0
	13	0	2	0	2	0	0	2	2	0	2	2	0	0	0	0	4
	14	0	0	0	2	0	0	2	0	0	2	0	4	2	4	0	0
	15	0	0	0	0	2	0	4	2	0	0	0	0	0	2	6	0

Tablica 3.2. – distribucija diferencija S-kutije

Sve vrijednosti u tablici 3.2. su u dekadskom brojevnom sustavu. Krajnji lijevi stupac predstavlja vrijednosti ulaznih diferencija za svaki redak, dok najgornji redak predstavlja vrijednosti izlaznih diferencija za svaki stupac.

Rezultati u tablici 3.2. posljedica su guranja svih mogućih parova ulaznih vrijednosti, za svaku moguću ulaznu diferenciju, kroz S_1, te bilježenja pojave svakog izlaznog para koji ima određenu izlaznu diferenciju za promatrani ulazni par vrijednosti.

Može se primijetiti kako diferencijalni trag $0 \rightarrow 0$ ima najveću frekvenciju, no ovaj diferencijalni trag nema veliku važnost jer jednostavno ukazuje na činjenicu da izostanak promjena na ulazu uzrokuje izostanak promjena na izlazu.

Napadača zanimaju crveno označene vrijednosti u tablici 3.2.. Za danu S_1 konfiguraciju pronađena su četiri jaka diferencijalna traga. Ovi diferencijalni tragovi izdvojeni su u tablici 3.3.

$\Delta X \rightarrow \Delta Y$
$1 \rightarrow 13$
$4 \rightarrow 7$
$8 \rightarrow 5$
$15 \rightarrow 14$

Tablica 3.3. – diferencijalni tragovi s najvećom frekvencijom pojave

Za svaki izdvojeni diferencijalni trag vrijedi da će se za odabranu ulaznu diferenciju (npr. 1) odgovarajuća izlazna diferencija (13) pojaviti 6 puta. Vrijednosti u tablici distribucije diferencija mogu se promatrati i kao indikatori broja ulaznih vrijednosti koji, u odgovarajućim međusobnim kombinacijama, dovode do pojave odgovarajuće ulazne i izlazne diferencije. Ovo svojstvo doći će do izražaja upravo prilikom pronalaženja potključeva.

Uz izuzetak prvog reda, u prosjeku se za svaku ulaznu diferenciju ostvaruje 6.267 različitih izlaznih diferencija. Od 256 mogućih kombinacija ulazno-izlaznih diferencija javlja se njih 94, odnosno 36.72%.

Za navedenu S_1 konfiguraciju može se reći kako je relativno slaba jer je mogući prostor kombinacija iskorišten nešto više od trećine. U određenoj mjeri se ovaj nedostatak može pripisati mogućnosti da

korištena S_1 konfiguracija vjerojatno nije optimalna, no problem također leži i u činjenici da se radi o šifri koja koristi 4-bitne vrijednosti, što samo po sebi ne daje širok raspon mogućnosti. U kasnijim poglavljima stavljen je fokus na optimizaciju distribucije diferencija.

3.3. Napad

Napad je implementiran računalno pomoću programskog jezika python, te niza pomoćnih biblioteka. Sav računalni kod, kao i popis verzija svih korištenih programa, nalazi se u prilogu.

Tijek napada predstavljen je slikom u prilogu P.3.3.. Pojedinosti, poput reorganizacije učitanih podataka, pretvaranja iz jednog u drugi tip podatka, i sličnih radnji, nisu detaljno prikazane na slici u prilogu P.3.3. jer se radi o općenitim radnjama koje nisu direktno vezane uz algoritam napada.

Obzirom da su frekvencije diferencijalnih tragova jednake (tablica 3.3.), algoritam odabire nasumično jedan od diferencijalnih tragova, no za potrebe demonstracije algoritma ručno je odabran diferencijalni trag $8 \rightarrow 5$. Korišteni glavni ključ je 101110, a konfiguracije potključeva, te S_1 i S_2 kutija su jednake onima navedenim u potpoglavlju 3.1.. Na temelju glavnog potključa algoritam daje potključeve $SK_1 = 1011$ i $SK_2 = 1110$.

Za diferencijalni trag $8 \rightarrow 5$ algoritam pronalazi dobre parove prikazane u tablici 3.4..

Dobri parovi	
Ulazi	Izlazi
[10, 2]	[0, 5]
[13, 5]	[11, 14]
[14, 6]	[7, 2]
[2, 10]	[5, 0]
[5, 13]	[14, 11]
[6, 14]	[2, 7]

Tablica 3.4. – dobri parovi diferencijalnog traga $8 \rightarrow 5$

Ulazne vrijednosti svih parova mapiraju se tako da prva ulazna vrijednost prvog para daje prvu izlaznu vrijednost za taj par. Na primjer, ako je ulazni par [10, 2], ulazna vrijednost 10 daje izlaz 0, a ulazna vrijednost 2 daje izlaz 5.

Preostali parovi su loši parovi jer za danu ulaznu diferenciju ne daju očekivanu izlaznu diferenciju.

Može se primijetiti da se broj jedinstvenih ulaznih vrijednosti, te broj parova koje one daju, podudara s frekvencijom iz tablice distribucije diferencija, kao i broj izlaza. Iz ulaznih parova izdvajaju se sve jedinstvene vrijednosti za koje se pretpostavlja da se mogu javiti nakon prve XOR operacije. To su vrijednosti 10, 2, 13, 5, 14 i 6, koje će se zbog jednostavnosti označavati grčkim slovom α. Ove vrijednosti kao izlaz iz S_1 daju vrijednosti 15, 13, 0, 1, 9 i 5, koje su zbog jednostavnosti označene grčkim slovom β. Pretpostavljene ulazne vrijednosti u S_1, te odgovarajuće izlazne iz S_1 dane su u tablici 3.5..

Ulaz u S_1 (α)	Izlaz iz S_1 (β)
10	15
2	13
13	0
5	1
14	9
6	5

Tablica 3.5. – pretpostavljene vrijednosti na ulazu u S_1, te njihovi izlazi

Neka je ulazni par u šifru [13, 5]. Odabrani ulazni par rezultira izlazom [11, 14]. Nasumično je odabrana α vrijednost 6. XOR operacija vrijednosti 13 i 6 daje za pretpostavljenu vrijednost prvog potključa vrijednost 1011, odnosno 11. Slučajno se upravo radi o stvarnom prvom potključu.

Obzirom da je α vrijednost 6 (ulaz u S_1), β vrijednost je 5 (izlaz iz S_1). Primjenom XOR operacije između β vrijednosti te izlazne vrijednosti 11 (vrijednost koja je posljedica ulazne vrijednosti 13), dobiva se pretpostavljenu vrijednost drugog potključa iznosa 1110, odnosno 14.

U ovom izdvojenom slučaju ispravno je odabrana α vrijednost, a posljedično i β vrijednost. Kada je pronađen par potključeva za koji se pretpostavlja da bi mogao biti ispravan, oni se testiraju nad svim lošim parovima tako da se par otvorenog teksta svakog lošeg para šifrira s pretpostavljenim potključevima, te se rezultat šifriranja uspoređuje sa stvarnim šifratima. Ako se šifrati postignuti pretpostavljenim potključevima podudaraju, brojač ispravnih pojavljivanja se inkrementira korištenom paru potključeva. Za svaki potključ postoji zasebna lista u kojoj indeks svakog mjesta u listi predstavlja pretpostavljeni potključ, a vrijednost na pojedinom indeksu predstavlja frekvenciju pojave tog potključa. Rezultati frekvencija za diferencijalni trag $8 \rightarrow 5$ nalaze se na slici 3.3.. Radi se o snimci ispisa zaslona konzole.

```
SK1 frequencies : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 40, 0, 0, 0, 0]
SK2 frequencies : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 40, 0]
```

Slika 3.3. – frekvencije pojedinog ključa

Iz slike 3.3. vidljivo je da se element na 11. indeksu u prvoj listi, te element na 14. indeksu u drugoj listi, javljaju 40 puta. Jedanaesti indeks u prvoj listi predstavlja $SK_1 = 1011$, dok četrnaesti indeks u drugoj listi predstavlja $SK_2 = 1110$.

Na prvi pogled ovaj rezultat može djelovati neintuitivno. Razumno je očekivati određeni broj pojava i drugih potključeva, a ne samo onih koji su ispravni. Ovaj rezultat posljedica je upravo usporedbe šifrata šifriranog s pretpostavljenim parom potključeva sa šifratom koji je inicijalno provučen kroz šifru prilikom kreiranja dobri i loših parova. Bez ovog uvjeta dobivaju se rezultati prikazani na slici 3.4..

```
SK1 frequencies : [60, 0, 0, 40, 40, 0, 0, 40, 60, 0, 0, 40, 40, 0, 0, 40]
SK2 frequencies : [20, 20, 40, 10, 20, 20, 10, 40, 10, 20, 20, 40, 20, 10, 40, 20]
```

Slika 3.4. – frekvencije ključeva bez usporedbe šifrata

Iz slike 3.4. vidljivo je da postoji niz frekvencija pojava obaju potključeva koje bi svojim iznosom navele napadača na krivi zaključak, ili nedoumicu između nekoliko potključeva s istim brojem pojava. Prethodno navedeni uvjet osigurava da se to ne dogodi.

Broj pojava SK_1 i SK_2 na slici 3.4. posljedica je pojavljivanja ispravnog para potključeva više puta prilikom ispitivanja svih dobrih parova.

Za vrijednosti koje su šifrirane glavnim ključem 101110, odnosno potključevima 1011 i 1110, te pomoću diferencijalnog traga $8 \rightarrow 5$, potključevi su uspješno otkriveni.

Važno je istaknuti da je diferencijalna kriptoanaliza vjerojatnosni napad, što znači da jaka karakteristika diferencije ne mora dovesti do uspješnog otkrivanja potključeva. Jaka karakteristika samo indicira da postoji ulazno-izlazni odnos koji se može iskoristiti.

4. KONFIGURACIJA S-KUTIJA

S-kutije i brojnost rundi su dva temeljna elementa koji znatno doprinose sigurnosti šifre. Moderne šifre ne koriste S-kutije samostalno, već se oslanjaju na Shannonov princip konfuzije i difuzije. S-kutije skrivaju odnos između ulaznih i izlaznih bitova, te odnos između šifrata i ključa, odnosno omogućuju konfuziju. Permutacijske kutije rasprostiru utjecaj promjene svakog pojedinog bita [7][8, str. 68.]. Šifra predstavljena u 3. poglavlju ne koristi permutacijske kutije.

U ovom poglavlju ispitan je utjecaj različitih konfiguracija S_1 kutije na tablicu distribucije diferencija s ciljem postizanja što je moguće uniformnije raspodjele. Također je demonstriran utjecaj jednostavnih logičkih operacija s ulazom i izlazom S_1 kutije na distribuciju diferencija.

4.1. Utjecaj različitih S_1 konfiguracija

Kako bi se testirali utjecaji različitih konfiguracija S_1 kutije uvedene su potrebne promjene u vidu automatskog generiranja drugačije S_1 konfiguracije prilikom svakog izvođenja skripte.

Prva S_1 konfiguracija je S_1 = [8, 9, 1, 0, 6, 7, 12, 11, 5, 3, 15, 13, 10, 4, 2, 14]. Ova konfiguracija rezultira tablicom distribucije diferencija 4.1..

Druga S_1 konfiguracija je S_1 = [6, 4, 0, 5, 9, 13, 10, 12, 7, 11, 2, 14, 1, 15, 8, 3]. Ova konfiguracija daje tablicu distribucije diferencija 4.2..

Treća S_1 konfiguracija je S_1 = [1, 15, 13, 5, 8, 11, 3, 10, 12, 0, 9, 4, 7, 14, 2, 6]. Ova konfiguracija daje tablicu distribucije diferencija 4.3..

U svakoj tablici su najveće frekvencije označene crvenom bojom.

Broj prosječnih različitih vrijednosti po redu tablice (uvijek uz izuzetak prvog reda), ukupan broj iskorištenih ulazno-izlaznih kombinacija, te broj visoko vjerojatnih tragova diferencija naveden je uz svaku tablicu, a zbog preglednosti su svi parametri svake S_1 konfiguracije navedeni u tablici 4.4..

		ΔY															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ΔX	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	6	2	0	0	0	2	2	0	0	0	0	2	0	2	0
	2	0	0	0	0	0	0	0	0	2	4	6	0	2	0	2	0
	3	0	0	0	0	2	0	2	0	6	0	0	2	2	2	0	0
	4	0	0	0	2	0	0	0	2	0	0	0	2	0	4	4	2
	5	0	4	0	0	0	0	0	0	0	2	2	0	2	0	0	6
	6	0	0	2	0	2	2	0	6	0	2	0	0	0	2	0	0
	7	0	2	0	2	0	2	4	2	0	0	0	4	0	0	0	0
	8	0	0	0	2	0	2	0	0	0	0	2	0	2	4	4	0
	9	0	0	4	0	0	0	0	0	0	2	0	2	4	2	0	2
	10	0	0	0	2	6	0	2	2	0	2	0	0	0	0	0	2
	11	0	2	2	0	0	6	2	0	4	0	0	0	0	0	0	0
	12	0	0	2	6	2	0	2	0	0	0	0	0	0	2	2	0
	13	0	2	4	2	2	2	0	0	0	0	0	0	2	0	0	2
	14	0	0	0	0	2	0	0	2	2	4	4	2	0	0	0	0
	15	0	0	0	0	0	2	2	0	2	0	2	4	0	0	2	2

Tablica 4.1. – distribucija diferencija za prvu S_1 konfiguraciju

Uz izuzetak prvog reda, prosječan broj različitih vrijednosti po redu je $5.9\dot{3}$. Od 256 mogućih kombinacija iskorišteno je njih 89, odnosno 34.76% svih mogućih vrijednosti. Ova testirana S_1 konfiguracija daje šest visoko vjerojatnih tragova diferencije jednake vjerojatnosti pojave, koji su označeni crvenom bojom u tablici.

ΔX \ ΔY	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	2	0	2	2	2	0	0	0	0	2	4	0	2	0
2	0	4	0	2	0	4	2	0	0	2	0	0	2	0	0	0
3	0	0	2	2	2	2	0	4	0	4	0	0	0	0	0	0
4	0	0	0	0	2	0	2	0	0	4	4	0	0	2	0	2
5	0	2	0	0	0	0	2	0	2	0	2	2	2	2	0	2
6	0	2	0	2	0	0	0	0	6	2	0	0	2	0	0	2
7	0	0	0	2	2	0	0	0	0	0	2	0	2	4	2	2
8	0	2	6	0	0	0	0	0	2	0	0	2	0	0	0	4
9	0	0	0	2	2	0	2	2	0	2	0	0	2	2	2	0
10	0	2	0	2	2	0	0	2	0	0	2	2	0	0	4	0
11	0	0	2	0	0	4	2	0	2	0	2	2	0	2	0	0
12	0	0	2	0	0	0	4	2	4	0	0	2	0	0	2	0
13	0	0	2	2	2	2	0	0	0	2	2	0	0	2	2	0
14	0	2	0	2	0	0	0	4	0	0	2	2	0	2	2	0
15	0	2	0	0	2	2	0	2	0	0	0	2	2	0	0	4

Tablica 4.2. – distribucija diferencija druge S_1 konfiguracije

Druga S_1 konfiguracija daje prosječno $6.7\dot{3}$ različitih vrijednosti po redu, te ukupno daje 101 ulazno-izlazno kombinaciju od 256 mogućih, odnosno 39.45%. Testirana S_1 kombinacija daje dva visoko vjerojatna traga diferencije jednakog iznosa vjerojatnosti, koje su u tablici istaknute crvenom bojom.

		ΔY															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ΔX	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	2	2	0	0	0	2	4	0	0	2	2	2	0
	2	0	2	0	0	2	4	0	0	2	0	2	2	2	0	0	0
	3	0	2	4	0	2	0	0	0	4	2	0	0	2	0	0	0
	4	0	0	2	0	2	0	0	0	0	2	0	4	0	0	4	2
	5	0	0	2	0	0	0	4	6	0	0	2	0	0	0	0	2
	6	0	0	2	0	0	4	2	0	0	0	2	0	0	0	6	0
	7	0	0	2	2	0	0	2	2	0	0	2	2	2	2	0	0
	8	0	4	0	0	2	2	0	0	0	0	0	0	2	2	0	4
	9	0	2	0	2	0	2	2	0	2	2	0	0	4	0	0	0
	10	0	2	0	0	4	2	0	0	2	0	2	2	0	2	0	0
	11	0	0	0	0	0	2	2	0	0	4	0	0	0	6	2	0
	12	0	2	0	2	2	0	2	0	0	0	2	2	0	0	2	2
	13	0	0	0	2	0	0	0	6	4	0	0	2	0	0	0	2
	14	0	2	0	2	0	0	0	0	0	2	4	2	0	0	0	4
	15	0	0	4	4	0	0	2	2	0	0	0	0	2	2	0	0

Tablica 4.3. – distribucija diferencija za treću S_1 konfiguraciju

Treća S_2 konfiguracija daje prosječno 6.267 različitih vrijednosti po redu, te od ukupno 256 mogućih ulazno-izlazni kombinacija realizira njih 94, odnosno 36.72%. Četiri visoko vjerojatna traga diferencije označena su crvenom bojom u tablici.

S1 konfiguracija	Prosječan broj različitih vrijednosti po redu	Realiziran broj ulazno-izlaznih vrijednosti (od 256)	Broj visoko vjerojatnih tragova diferencija
1.	$5.9\dot{3}$	89 (34.76%)	6
2.	$6.7\dot{3}$	101 (39.45%)	2
3.	6.267	94 (36.72%)	4

Tablica 4.4. – evaluacijski parametri tablica distribucija diferencija svih S_1 konfiguracija

Druga S_1 konfiguracija rezultira iskorištavanjem blizu 40% svih mogućih ulazno-izlaznih vrijednosti, te daje samo dva visoko vjerojatna traga diferencije. Po ovim parametrima se ova konfiguracija može izdvojiti kao najbolja od tri predstavljene. Ipak, iscrpna pretraga svih mogućih S_1 konfiguracija vjerojatno bi rezultirala određenim brojem konfiguracija koje imaju znatno uniformniju raspodjelu, te nizak broj visoko vjerojatnih tragova diferencija. Ovakva pretraga zahtijevala bi pretraživanje $\sim 2.092^{13}$ mogućnosti.

4.2. ILI operacija nad izlazom S_1

Logička operacija ili primijenjena je na izlaze iz S_1 kako bi se ispitao učinak na tablicu distribucije diferencija. Rezultati za konfiguraciju S_1 = [8, 9, 1, 0, 6, 7, 12, 11, 5, 3, 15, 13, 10, 4, 2, 14] prikazani su u tablici 4.5.. Uz izuzetak prvog reda se po redu javlja prosječno 3.2 različitih vrijednosti, te je ukupno iskorišteno samo 48 od 256 mogućih ulazno-izlaznih kombinacija, odnosno 18.75%. Javlja se jedanaest tragova diferencije visoke frekvencije, od kojih se jedan ulazno-izlazni par javlja dvanaest puta. Prostor mogućih vrijednosti je smanjen provođenjem ILI operacije nad izlazom S_1, što je i vidljivo prema iskorištenosti vrijednosti u tablici 4.5.

Preostali rezultati nisu navedeni jer su skladu s izdvojenim primjerom.

		ΔY															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ΔX	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	4	8	0	0	0	0	0	0	4	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	12	4	0	0	0	0	0	0
	3	4	0	0	0	0	0	0	0	8	4	0	0	0	0	0	0
	4	0	4	0	0	0	0	0	0	4	8	0	0	0	0	0	0
	5	0	4	0	0	0	0	0	0	4	8	0	0	0	0	0	0
	6	4	8	0	0	0	0	0	0	0	4	0	0	0	0	0	0
	7	4	8	0	0	0	0	0	0	0	4	0	0	0	0	0	0
	8	0	4	0	0	0	0	0	0	8	4	0	0	0	0	0	0
	9	4	0	0	0	0	0	0	0	4	8	0	0	0	0	0	0
	10	8	4	0	0	0	0	0	0	0	4	0	0	0	0	0	0
	11	4	8	0	0	0	0	0	0	4	0	0	0	0	0	0	0
	12	6	6	0	0	0	0	0	0	2	2	0	0	0	0	0	0
	13	6	6	0	0	0	0	0	0	2	2	0	0	0	0	0	0
	14	2	2	0	0	0	0	0	0	6	6	0	0	0	0	0	0
	15	2	2	0	0	0	0	0	0	6	6	0	0	0	0	0	0

Tablica 4.5. – tablica distribucija diferencija S_1 konfiguracije uz ILI operaciju na S_1

4.3. ILI operacija nad izlazom S_1 s pomičnom vrijednošću

Logička operacija ILI primijenjena je na izlazne vrijednosti iz S_1 tako da se bitovi vrijednosti s kojom izlaz S_1 obavlja ILI operaciju rotiraju za jedno mjesto ulijevo prilikom primjene ILI operacije nad svakim bitom. Tablica distribucije diferencija 4.6. prikazuje ostvarene rezultate za S_1 = [8, 9, 1, 0, 6, 7, 12, 11, 5, 3, 15, 13, 10, 4, 2, 14].

Uz izuzetak prvog reda, prosječno se po redu javlja 3.467 različitih vrijednosti, te je ukupno iskorišteno 52 od mogućih 256 ulazno-izlaznih kombinacija, odnosno 20.31%. Primjena pomične vrijednosti rezultira s deset jakih tragova diferencije, koji su istaknuti crvenom bojom u tablici 4.6..

Preostali primjeri nisu navedeni jer su u skladu s primjerom prikazanim u tablici 4.6..

		ΔY															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ΔX	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	6	0	6	0	0	0	0	0	2	0	2	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	8	0	8	0	0	0	0	0
	3	2	0	2	0	0	0	0	0	10	0	2	0	0	0	0	0
	4	0	0	4	0	0	0	0	0	4	0	8	0	0	0	0	0
	5	4	0	0	0	0	0	0	0	4	0	8	0	0	0	0	0
	6	4	0	8	0	0	0	0	0	4	0	0	0	0	0	0	0
	7	4	0	8	0	0	0	0	0	0	0	4	0	0	0	0	0
	8	2	0	2	0	0	0	0	0	6	0	6	0	0	0	0	0
	9	0	0	4	0	0	0	0	0	8	0	4	0	0	0	0	0
	10	6	0	6	0	0	0	0	0	2	0	2	0	0	0	0	0
	11	8	0	4	0	0	0	0	0	4	0	0	0	0	0	0	0
	12	2	0	10	0	0	0	0	0	2	0	2	0	0	0	0	0
	13	6	0	6	0	0	0	0	0	2	0	2	0	0	0	0	0
	14	2	0	2	0	0	0	0	0	6	0	6	0	0	0	0	0
	15	2	0	2	0	0	0	0	0	2	0	10	0	0	0	0	0

Tablica 4.6. – distribucija diferencija za ILI operaciju sa izlazom iz S_1 s pomičnom vrijednošću

4.4. ILI operacija nad ulazom u S_1 s pomičnom vrijednošću

Logička operacija ILI primijenjena je na ulazne vrijednosti u S_1 tako da se bitovi vrijednosti s kojom izlaz S_1 obavlja ILI operaciju rotiraju za jedno mjesto ulijevo prilikom primjene ILI operacije nad svakim bitom. Tablica distribucije diferencija 4.7. prikazuje ostvarene rezultate za S_1 = [8, 9, 1, 0, 6, 7, 12, 11, 5, 3, 15, 13, 10, 4, 2, 14].

Uz izuzetak prvog reda, prosječno se po redu javlja 1.8. različitih vrijednosti, te je ukupno iskorišteno 27 od mogućih 256 ulazno-izlaznih kombinacija, odnosno 10.55%.

		ΔY															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ΔX	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	8	0	8	0	0	0
	3	0	0	0	0	0	0	0	0	0	0	8	0	8	0	0	0
	4	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	0	0	0	0	0	0	0	8	0	8	0	0	0
	7	0	0	0	0	0	0	0	0	0	0	8	0	8	0	0	0
	8	0	0	0	8	0	8	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	8	0	8	0	0	0	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	8
	11	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	8
	12	0	0	0	8	0	8	0	0	0	0	0	0	0	0	0	0
	13	0	0	0	8	0	8	0	0	0	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	8
	15	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	8

Tablica 4.7. – distribucija diferencija za ILI operaciju s ulaznom vrijednošću u S_1 s pomičnom vrijednošću

5. ZAKLJUČAK

U radu je predstavljena temeljna ideja diferencijalne kriptoanalize na općenitom primjeru. Napad diferencijalnom kriptoanalizom implementiran je u python programskom jeziku za proizvoljno definiranu šifru, te je za šifru i odabrani diferencijalni trag prikazan način dohvaćanja potključeva.

Pretposljednje poglavlje sadrži prikaz nekoliko različitih konfiguracija S_1 kutije, te utjecaj dodavanja ILI logičke operacije na distribuciju diferencija. Demonstrirano je kako permutacije konfiguracija S_1 kutije mogu dovesti do određenih poboljšanja u uniformnosti tablice distribucije diferencija, no ove permutacije neće rezultirati značajnim poboljšanjima jer korištena šifra ne poštuje niz različitih kriptografskih principa.

Uz S-kutije nužno je koristiti permutacijske kutije koje značajno pomažu u dekorelaciji otvorenog teksta i šifrata. Principi poput vektora različitih duljina na ulazu i izlazu S-kutija značajno pomažu, uz poštivanje uvjeta da ulazni vektor treba biti veće duljine od izlaznog.

Ipak, mnoge dobre S-kutije nastaju kriptografskom analizom svih mogućih S-kutija te odabirom onih koje pružaju najbolje karakteristike. Prilikom poboljšavanja postojećih, ili implementiranja novih S-kutija svakako bi se trebalo osvrnuti i na rješenja i uvide pružene u [9]. AES S-kutije dizajnirane su imajući u vidu upravo [9].

Pruženo programsko rješenje kao takvo je funkcionalno. Primarna namjena ovog programskog rješenja je testiranje različitih potencijalnih poboljšanja i shvaćanje diferencijalne kriptoanalize. Poboljšanja programskog rješenja se mogu ostvariti u nizu područja:

- refaktoriranje programskog koda
- povezivanje rješenja u jednu cielinu
- poopćenje rješenja na ulaze i izlaze proizvoljne duljine
- vađenje konfigurabilnih parametara u konfiguracijsku datoteku

Refaktoriranjem koda postigla bi se značajno bolja čitljivost i razumljivost osobama koje nisu implementirale rješenje, te bi se mnogi elementi koda koji postoje samo zbog dodatnih provjera ispravnosti uklonili. Također bi se šifra mogla implementirati kao jedna runda za koju bi korisnik u konfiguracijskoj datoteci definirao koliko ju puta želi ponoviti, čime bi se dobila karakteristika n rundi, gdje je n definirani broj rundi.

Trenutačno je rješenje sastavljeno od nekoliko zasebnih skripti koje se izvode u određenom redoslijedu, te se pozivaju ručno, jedna nakon druge. Za osobu koja nije upoznata s rješenjem, za koje ne postoji formalna dokumentacija, postojao bi određeni period koji bi zahtijevao upoznavanje sa svim funkcionalnostima. Ovaj problem može se otkloniti međusobnim vezivanjem skripti, te pozivanjem samo jedne, glavne, skripte koja izvodi sve potrebne radnje i izradom dokumentacije.

Poopćenjem rješenja na ulaze i izlaze proizvoljne duljine omogućilo bi se provođenje različitih testiranja s ulaznim i izlaznim vektorima željene duljine. Ovo poboljšanje usko je vezano s vađenjem svih konfigurabilnih parametara u konfiguracijsku datoteku. Izvlačenje konfigurabilnih parametara podrazumijeva parametere poput duljine ulaznih i izlaznih vektora, vrijednost otvorenog teksta za iduće šifriranje, vrijednost glavnog ključa, konfiguracija potključeva, konfiguracija S-kutija i broj rundi.

LITERATURA

[1] : E., Biham, A., Shamir, *Differential cryptanalysis of DES-like cryptosystems*, Journal of Cryptology 4**,** 3–72, 1991

[2] : D., Coppersmith, *The Data Encryption Standard (DES) and its strength against attacks*, IBM Journal of Research and Development. 38 (3), svibanj 1994.

[3] : Y., Wang, W., Wu, Z., Guo, X., Y, *Differential Cryptanalysis and Linear Distinguisher of Full-Round Zorro,*: International Conference on Applied Cryptography and Network Security, lipanj 2014.

[4] : H., M., Heys, *A Tutorial on Linear and Differential Cryptanalysis*, Cryptologia 26(3), srpanj 2002.

[5] : D., Mukhopadhyay, *lecture : Differential cryptanalysis*, Department of Computer Science and Engineering, Indian Institute of Tehchonology Kharagpur

[6] : D., R., Stinson, *Cryptography : Theory and Practice, 3rd edition,* Chapman & Hall/CRC, 2006.

[7] : https://cryptography.fandom.com/wiki/Confusion_and_diffusion [12.8.2021.]

[8] : W., Stallings, Cryptography and Network Security, 4. izdanje, Prentice Hall, 2005.

[9] : K., Nyberg, *Perfect nonlinear S-boxes*, Davies D.W., Advances in Cryptology — EUROCRYPT ’91. EUROCRYPT 1991. Lecture Notes in Computer Science, broj 547. Springer, Berlin, Heidelberg, (1991)

[10] : A., Dujella, M., Maretić, *Kriptografija*, Element, Zagreb, 2007.

[11] : https://en.wikipedia.org/wiki/Feistel_cipher#/media/File:Feistel_cipher_diagram_en.svg (15.8.2021.)

SAŽETAK

U ovom diplomskom radu opisana je temeljna ideja diferencijalne kriptoanalize na općenitom primjeru. Implementirana je jednostavna, proizvoljna, 4-bitna šifra sa dvije runde, jednom S-kutijom u svakoj rundu, te jednim potključem u svakoj rundi. Za ovu šifru provedena je potrebna analiza koju zahtijeva diferencijalna kriptoanaliza, a uključuje analizu karakteristike šifre do R-1 rundi i analizu frekvencija ulaza i izlaza šifre za karakteristiku do R-1 rundi. Nakon obavljene analize implementiran je napad kojeg opisuje diferencijalna kriptoanaliza na 4-bitnu šifru. Potključevi su uspješno dohvaćeni, a nad S-kutijom su provedeni daljnji testovi kako bi se uočio utjecaj konfiguracije S-kutije na ulazno-izlazne odnose. Sav programski kod realiziran je u python programskom jeziku.

Ključne riječi : diferencijalna kriptoanaliza, S-kutije, supstitucijsko-permutacijske mreže

ABSTRACT

The fundamental idea of differential cryptanalysis is described in this thesis on a general example. A simple, arbitrary, 4-bit cipher has been implemented. This cipher consists of two rounds, one S-box in each round, and one subkey in each round. Necessary analysis dictated by differential cryptanalysis has been conducted for this 4-bit cipher, which includes analysing cipher characteristic to R-1 rounds and analysing input and output frequencies for cipher's R-1 round characteristic. After analysis, differential cryptanalysis attack on the 4-bit cipher has been implemented. Subkeys were successfully retrieved, and further analyses of S-boxes have been conducted to test out the S-box configuration influence on the input and output relations. All code was implemented using python programming language.

Key words : differential cryptanalysis, S-box, supstitution-permutation networks

ŽIVOTOPIS

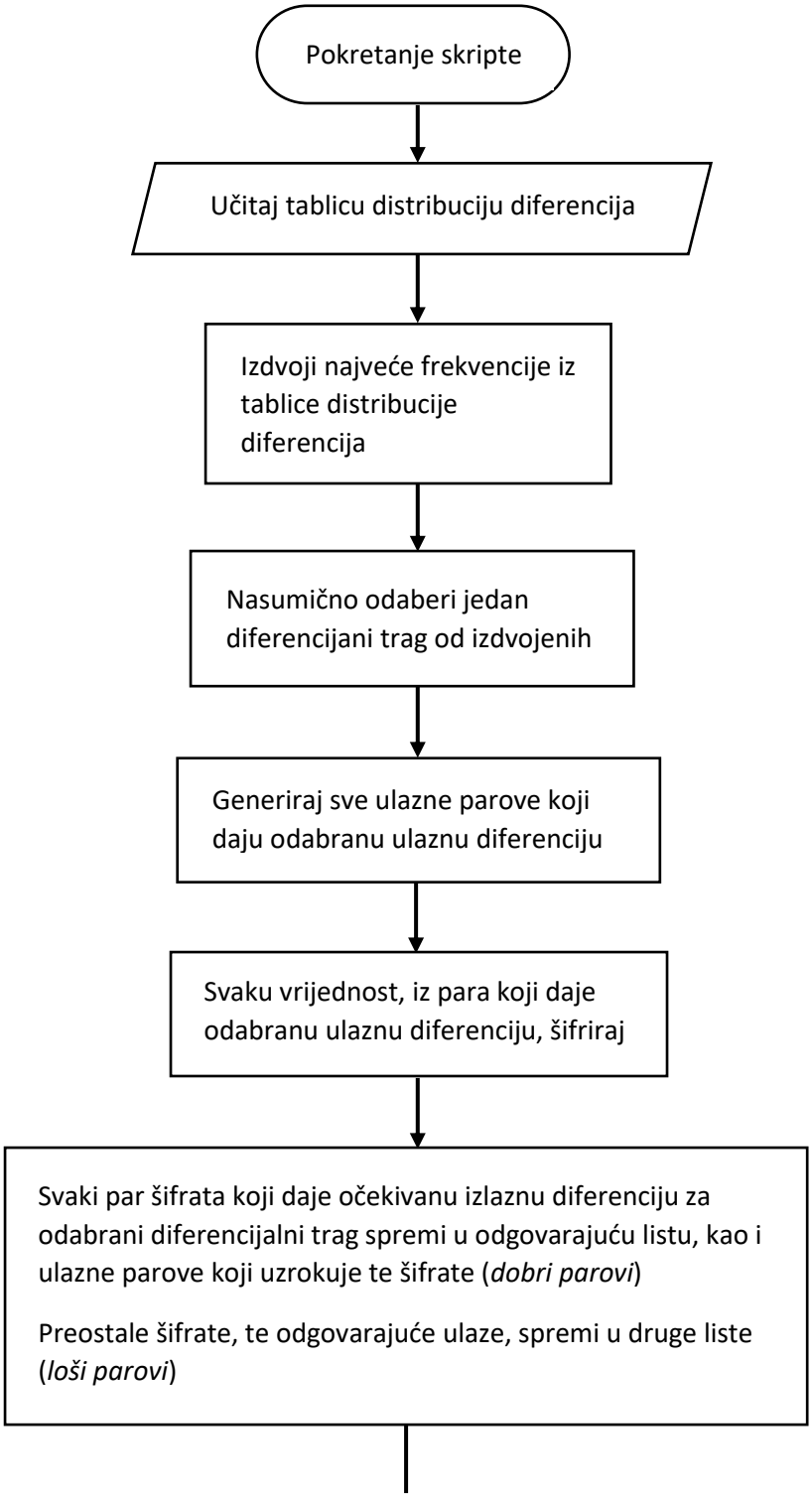
Josip Uršan rođen je 11. veljače 1997. godine u Zagrebu. Nakon pohađanja osnovne škole u Osijeku, upisao je I. gimnaziju u Osijeku. Tijekom osnovnoškolskog i srednjoškolskog obrazovanja sudjelovao je na dva županijska natjecanja iz povijesti, te jednom županijskom natjecanju iz engleskog jezika. Tijekom osnovnoškolskog obrazovanja pohađao je tri napredna tečaja engleskog jezika u školi stranih jezika Lanico, dok je tijekom srednjoškolskog obrazovanja položio dva ispita znanja njemačkog jezika Konferencije ministara kulture Savezne Republike Njemačke za učenike u inozemstvu (DSD I i DSD II).

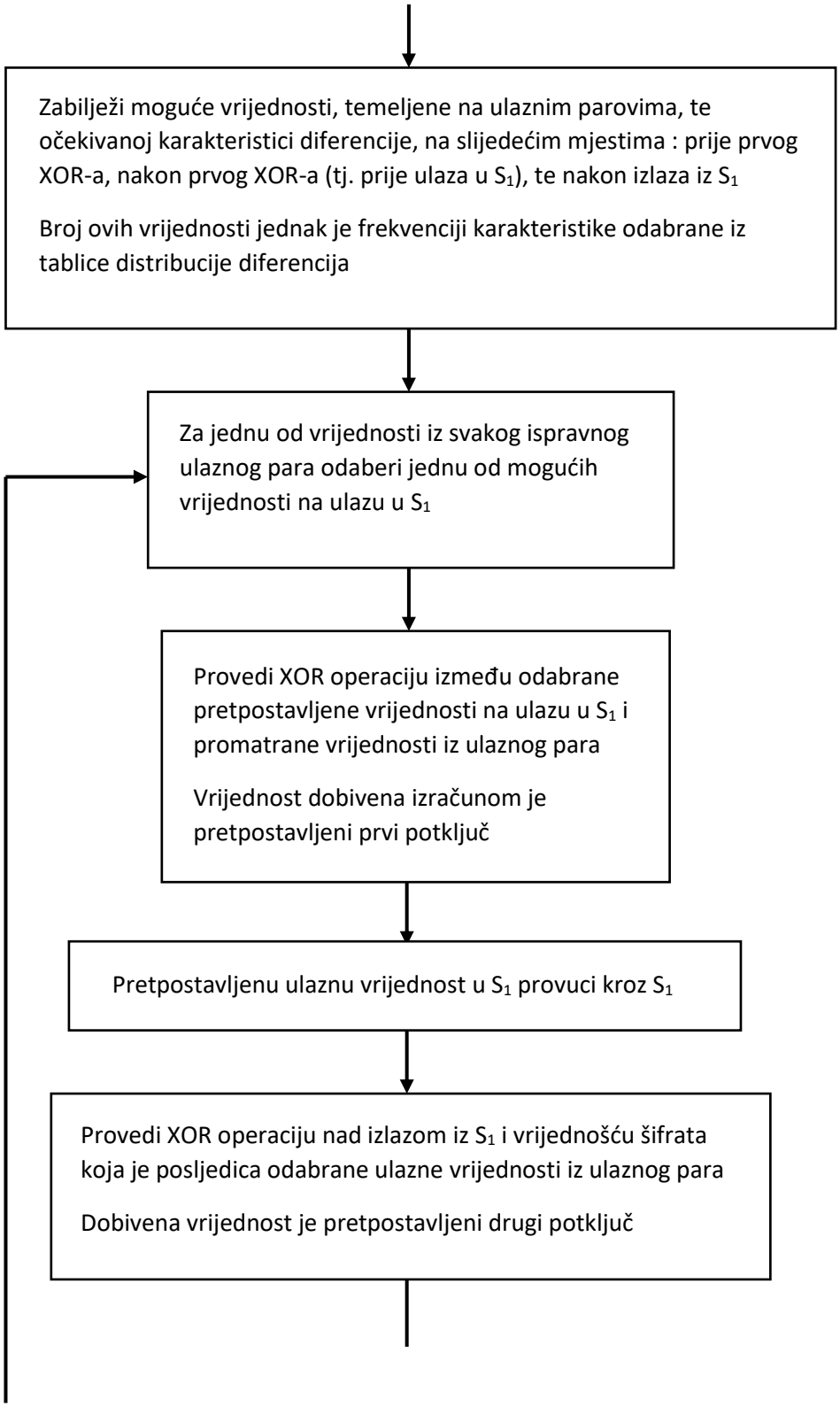
Nakon položene mature, 2016. godine upisuje prvu godinu preddiplomskog studija elektrotehnike na FERIT-u. Finalist je EWOB (Entrepreneurs without borders) natjecanja 2018. godine, koje organiziraju studenti Ekonomskog fakulteta u Osijeku, te sudionik Ericsson Summer Camp-a 2019. godine.

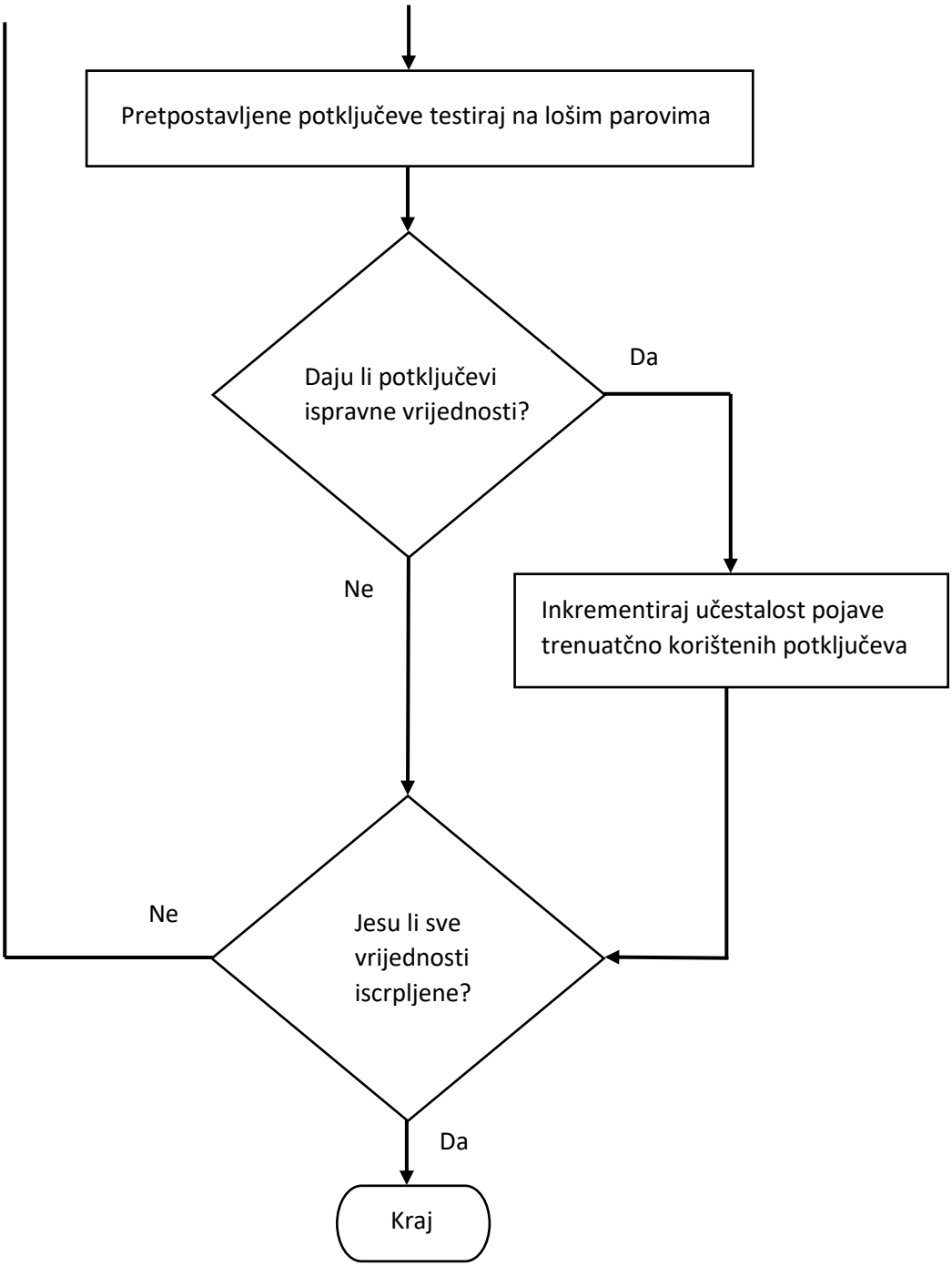
Godine 2019. upisuje diplomski studij elektrotehnike, smjer komunikacije i informatika, izborni blok mrežne tehnologije (DKB), na FERIT-u. Obaveznu praksu odrađuje u Ericsson Nikola Tesla-i, gdje nastavlja raditi preko studentskog ugovora od veljače 2021. godine do srpnja 2021. godine. Dobitnik je priznanja za uspješnost u studiranju na diplomskom studiju.

PRILOZI

P.3.3. – tok programa




Zabilježi moguće vrijednosti, temeljene na ulaznim parovima, te očekivanoj karakteristici diferencije, na slijedećim mjestima : prije prvog XOR-a, nakon prvog XOR-a (tj. prije ulaza u S_1), te nakon izlaza iz S_1
Broj ovih vrijednosti jednak je frekvenciji karakteristike odabrane iz tablice distribucije diferencija
Za jednu od vrijednosti iz svakog ispravnog ulaznog para odaberi jednu od mogućih vrijednosti na ulazu u S_1
Provedi XOR operaciju između odabrane pretpostavljene vrijednosti na ulazu u S_1 i promatrane vrijednosti iz ulaznog para
Vrijednost dobivena izračunom je pretpostavljeni prvi potključ
Pretpostavljenu ulaznu vrijednost u S_1 provuci kroz S_1
Provedi XOR operaciju nad izlazom iz S_1 i vrijednošću šifrata koja je posljedica odabrane ulazne vrijednosti iz ulaznog para
Dobivena vrijednost je pretpostavljeni drugi potključ


Pretpostavljene potključeve testiraj na lošim parovima
Daju li potključevi ispravne vrijednosti?
Da
Ne
Inkrementiraj učestalost pojave trenuatčno korištenih potključeva
Jesu li sve vrijednosti iscrpljene?
Ne
Da
Kraj

Korišteni softver s verzijama

Softver	Verzija
Python	3.9.1.
json	*
numpy	1.19.5
itertools	*
ast	*
pandas	1.2.1
Visual Studio Code	1.59.0

Oznaka zvjezdice (*) u stupcu verzije odnosi se na pakete čije su verzije vezane uz korištenu verziju pythona.

Specifikacije korištenog računala

CPU	Intel G2020
RAM	8GB (frekvencija : 1333MHz)
GPU	integrirana Intel grafika treće generacije
Chipset	Intel
Pohrana podataka	256GB SSD, 1TB HDD
OS	Windows 10 Pro, Build : 19041.1110

S1BlackBox.py

```
import json
import numpy as np

# S box mapping
#S1 = [[3,0], [1,2]]
#S2 = [[1,0], [3,2]]

S1 = [3, 14, 1, 10, 4, 9, 5, 6, 8, 11, 15, 2, 13, 12, 0, 7]
S2 = [3, 14, 1, 10, 4, 9, 5, 6, 8, 11, 15, 2, 13, 12, 0, 7]

with open("../outputs/allPossibleInputs.json", "r") as allOutputsForAllInputsFile:
    readData = json.load(allOutputsForAllInputsFile)

readDataPairs = readData.items()

SBoxInputs = []
outputs = []

for key, value in readDataPairs:
    SBoxInputs.append(list(str("".join(str(x) for x in value))))

for i in range(0, len(SBoxInputs)):
    for j in range(0, len(SBoxInputs[i])):
        SBoxInputs[i][j] = int(SBoxInputs[i][j])

endJsonStructure = {}
outputS1 = []

for i in range(0, len(SBoxInputs)):
    inputToInt = str(''.join(str(x) for x in SBoxInputs[i]))
    inputToInt = '0b' + inputToInt
    inputToInt = int(inputToInt, 2)

    #print("S1 output FIRST VALUE {} : {}".format(inputToInt, S1[inputToInt]))
    S1Out_first = np.binary_repr(S1[inputToInt], width = 4)
    S1_outputFirstValue = (list(map(int,(S1Out_first))))

    for j in range(0, len(SBoxInputs)):
        inputPairXOR = []
        outputPairXOR = []

        inputToInt_second = str(''.join(str(x) for x in SBoxInputs[j]))
        inputToInt_second = '0b' + inputToInt_second
```

```
        inputToInt_second = int(inputToInt_second, 2)
        #print("S1 output SECOND VALUE {} : {}".format(inputToInt_second, S1[inputT
oInt_second]))
        S1Out_second = np.binary_repr(S1[inputToInt_second], width = 4)
        S1_outputSecondValue = list(map(int, S1Out_second))

        for k in range(0, len(SBoxInputs[i])):
            inputPairXOR.append(SBoxInputs[i][k] ^ SBoxInputs[j][k])
            outputPairXOR.append(S1_outputFirstValue[k] ^ S1_outputSecondValue[k])

        # struktura : X', X'', X' XOR X'' ---> ulazni elementi
        jsonKey = str(''.join(str(x) for x in SBoxInputs[i])) + ',' + str(''.join(s
tr(x) for x in SBoxInputs[j])) + ',' + str(''.join(str(x) for x in inputPairXOR))

        # struktura : S1(X'), X1(X''), S1(X') XOR S1(X'')
        endJsonStructure[jsonKey] = [str("".join(str(x) for x in S1_outputFirstValu
e)), str("".join(str(x) for x in S1_outputSecondValue)), str("".join(str(x) for x i
n outputPairXOR))]

jsonString = json.dumps(endJsonStructure)
jsonFile = open("blackBoxIODifferentials.json", "w")
jsonFile.write(jsonString)
jsonFile.close()
```

blackBox_frequencyAnalysisDDT.py

```
import json
import itertools

#Format JSON strukture u blackBoxIODifferentials.json :
#   kljuc : X', X'', X' XOR X''
#   vrijednost : S1(X'), S1(X''), S1(X') XOR S1(X'')

with open("blackBoxIODifferentials.json", "r") as blackBoxResultsFile:
    readData = json.load(blackBoxResultsFile)

readDataPairs = readData.items()

startList = [0, 1]
blackBoxInputXORs = list(itertools.product(startList, repeat = 4))

test = []
for element in blackBoxInputXORs:
    test.append(str("".join(str(x) for x in element)))

for element in test:
    DDT_RowFrequencies = [0]*16
    for elements in readDataPairs:
        innerSplit = elements[0].split(',')
        if element == innerSplit[2]:
            indexAndXORValue = '0b' + elements[1][2]
            indexAndXORValue = int(indexAndXORValue, 2)
            DDT_RowFrequencies[indexAndXORValue] += 1

    with open("DDT.txt", 'a') as saveFile:
        saveFile.write(str(DDT_RowFrequencies))
        saveFile.write('\n')
```

DDT_to_csvFile.py

```
import ast
import pandas as pd

readLines = []

file1 = open('DDT.txt', 'r')
Lines = file1.readlines()

for line in Lines:
    readLines.append(ast.literal_eval(line.strip()))

sortByColumn = [[] for _ in range(16)]

for i in range(0, 16):
    for j in range(0, 16):
        sortByColumn[i].append(readLines[j][i])

cols = ['0', '1',  '2',  '3',  '4',  '5',  '6',  '7',  '8',  '9',  '10',  '11',  '1
2',  '13', '14', '15']
data = {'0':[], '1':[], '2':[], '3':[], '4':[], '5':[], '6':[], '7':[], '8':[], '9'
:[], '10':[], '11':[], '12':[], '13':[], '14':[], '15':[]}

for i in range(0, 16):
    for j in range(0, 16):
        data[str(j)].append(sortByColumn[j][i])

dataFrame = pd.DataFrame(data)
dataFrame.to_csv("DDT_csvFormat.csv", index = True)
```

attack.py

```
import ast
import itertools
import numpy as np

masterKey = "101110"

subkey0_generator = [3, 5, 0, 2]
subkey1_generator = [4, 2, 3, 1]

S1 = [3, 14, 1, 10, 4, 9, 5, 6, 8, 11, 15, 2, 13, 12, 0, 7]
S2 = [3, 14, 1, 10, 4, 9, 5, 6, 8, 11, 15, 2, 13, 12, 0, 7]

def stringToListOfInts(inputValue):
    inputValue = list(inputValue)
    returnValue = []
    for k in range(0, len(inputValue)):
        returnValue.append(int(inputValue[k]))

    return returnValue

def toyCipher_ciphering(valueForCiphering):
    subkey0 = []
    subkey1 = []

    for indexElement in subkey0_generator:
        subkey0.append(masterKey[indexElement])

    for indexElement in subkey1_generator:
        subkey1.append(masterKey[indexElement])

    print("\nSK1 : {}".format(subkey0))
    print("SK2 : {}\n".format(subkey1))

    for i in range(0, len(subkey0)):
        subkey0[i] = int(subkey0[i])
        subkey1[i] = int(subkey1[i])

    firstXOR = []
    for i in range(0, len(valueForCiphering)):
        firstXOR.append(valueForCiphering[i] ^ subkey0[i])

    inputToInt = str(''.join(str(x) for x in firstXOR))
    inputToInt = '0b' + inputToInt
```

```
    inputToInt = int(inputToInt, 2)

    newS1 = np.binary_repr(S1[inputToInt], width = 4)
    outputS1_split = stringToListOfInts(newS1)

    secondXOR = []
    for i in range(0, len(outputS1_split)):
        secondXOR.append(outputS1_split[i] ^ subkey1[i])

    return secondXOR

def encrypt_checkSubkeyValues(valueForCiphering, subkey1, subkey2):

    for i in range(0, len(subkey1)):
        subkey1[i] = int(subkey1[i])
        subkey2[i] = int(subkey2[i])

    firstXOR = []
    for i in range(0, len(valueForCiphering)):
        firstXOR.append(valueForCiphering[i] ^ subkey1[i])

    inputToInt = str(''.join(str(x) for x in firstXOR))
    inputToInt = '0b' + inputToInt
    inputToInt = int(inputToInt, 2)

    newS1 = np.binary_repr(S1[inputToInt], width = 4)
    outputS1_split = stringToListOfInts(newS1)

    cipheredValueForCheck = []
    for i in range(0, len(outputS1_split)):
        cipheredValueForCheck.append(outputS1_split[i] ^ subkey2[i])

    return cipheredValueForCheck

def applyS1(valueForS1):
    inputToInt = str(''.join(str(x) for x in valueForS1))
    inputToInt = '0b' + inputToInt
    inputToInt = int(inputToInt, 2)

    S1Output = np.binary_repr(S1[inputToInt], width = 4)
    S1_boxedValue = stringToListOfInts(S1Output)

    return S1_boxedValue
```

```
startingList = [0, 1]
allPossible4BitValues = list(itertools.product(startingList, repeat=4))
readLines = []

file1 = open('DDT.txt', 'r')
Lines = file1.readlines()

print(Lines)

for line in Lines:
    readLines.append(ast.literal_eval(line))

for element in readLines:
    print(element)
    print('\n')

rowColumnValues = []
foundProbabilities = []

for element in readLines:
    foundProbabilities.append(max(element))
    maxIndex = element.index(max(element))

    temp = []
    temp.append(readLines.index(element))   #index reda, tj deltaX
    temp.append(maxIndex)   #index stupca, tj. deltaY
    rowColumnValues.append(temp)


absoluteMaxProbabilites = []
absoluteMax_rowColumnProbabilities = []

del foundProbabilities[0]   #ukloni frekvenciju 16 od 0->0
del rowColumnValues[0]
print("FOUND PROBABILITIES : {}".format(foundProbabilities))
print("ROW COLUMN VALUES : {}".format(rowColumnValues))
indicesOfMaxValues = [index for index, value in enumerate(foundProbabilities) if va
lue == max(foundProbabilities)]
print(indicesOfMaxValues)

for indexValue in indicesOfMaxValues:
    absoluteMaxProbabilites.append(foundProbabilities[indexValue])
    absoluteMax_rowColumnProbabilities.append(rowColumnValues[indexValue])

print("\nABSOLUTE MAX PROBABILITIES : {}".format(absoluteMaxProbabilites))
```

```
print("THEIR ROW COL VALUES : {}".format(absoluteMax_rowColumnProbabilities))

print("\nMost probable differentials : ")
for i in range(0, len(absoluteMaxProbabilites)):
    print("{}/16 : {} -
> {}".format(absoluteMaxProbabilites[i], absoluteMax_rowColumnProbabilities[i][0],
absoluteMax_rowColumnProbabilities[i][1]))

index_chooseRandomDifferentialCharacteristic = 2#random.randrange(len(absoluteMaxPr
obabilites))
inputXOR = list(map(int, np.binary_repr(absoluteMax_rowColumnProbabilities[index_ch
ooseRandomDifferentialCharacteristic][0], width=4)) )
outputXOR = list(map(int, np.binary_repr(absoluteMax_rowColumnProbabilities[index_c
hooseRandomDifferentialCharacteristic][1], width=4)))
print("\n\nChosen characteristic : {}/16 : {} -
> {}".format(absoluteMaxProbabilites[index_chooseRandomDifferentialCharacteristic],
 absoluteMax_rowColumnProbabilities[index_chooseRandomDifferentialCharacteristic][0
], absoluteMax_rowColumnProbabilities[index_chooseRandomDifferentialCharacteristic]
[1]))
print("Chosen index : {}\nInput XOR : {}\nOutput XOR : {}".format(index_chooseRando
mDifferentialCharacteristic, inputXOR, outputXOR))


inputPairsYieldingInputXOR = []
for element in allPossible4BitValues:
    temp = []
    temp2 = []
    aux = []
    for i in range(0, len(inputXOR)):
        temp.append(inputXOR[i] ^ element[i])
        temp2.append(element[i])
    aux.extend([temp, temp2])
    inputPairsYieldingInputXOR.append(aux)

print("\nInput pairs yielding input XOR : {}".format(inputPairsYieldingInputXOR))


allOutputsYieldingWantedOutputXOR = []
remainingOutputsThatAreResultOfInputXOR = []
inputPairThatCausesDeltaY = []
remainingInputs = []

for inputPairs in inputPairsYieldingInputXOR:
    auxiliaryList = []
    auxList2 = []
```

```
    auxList3 = []
    auxRightInputs = []

    firstCipherText = toyCipher_ciphering(inputPairs[0])
    secondCipherText = toyCipher_ciphering(inputPairs[1])

    print("FCP : {}".format(firstCipherText))
    print("SCP : {}".format(secondCipherText))


    print("OT Xdash : {}  CT Ydash: {}".format(inputPairs[0], firstCipherText))
    print("OT Xdashdash : {}   CT Ydashdash : {}".format(inputPairs[1], secondCiphe
rText))

    firstSecondOutputXOR = []
    for k in range(0, len(firstCipherText)):
        firstSecondOutputXOR.append(firstCipherText[k] ^ secondCipherText[k])
    print("XOR : {}\n".format(firstSecondOutputXOR))

    if firstSecondOutputXOR == outputXOR:
        print("THIS PAIR MATCHES WANTED OUTPUT XOR : {} | {}".format(firstCipherTex
t, secondCipherText))
        auxiliaryList.append(firstCipherText)
        auxiliaryList.append(secondCipherText)
        allOutputsYieldingWantedOutputXOR.append(auxiliaryList)

        auxRightInputs.append(inputPairs[0])
        auxRightInputs.append(inputPairs[1])
        inputPairThatCausesDeltaY.append(auxRightInputs)
    else:
        auxList2.append(firstCipherText)
        auxList2.append(secondCipherText)
        remainingOutputsThatAreResultOfInputXOR.append(auxList2)
        auxList3.append(inputPairs[0])
        auxList3.append(inputPairs[1])
        remainingInputs.append(auxList3)

print("All outputs yielding wanted output XOR : {}".format(allOutputsYieldingWanted
OutputXOR))
print("LEN right outputs : {}".format(len(allOutputsYieldingWantedOutputXOR)))
print("Right inputs : {}".format(inputPairThatCausesDeltaY))
print("\nRemaining list of outputs that don't yield appropriate output XOR, but are
 a consequence of input values : {}".format(remainingOutputsThatAreResultOfInputXOR
))
```

```
print("LEN wrong outputs: {}\n\n\n".format(len(remainingOutputsThatAreResultOfInput
XOR)))


sixPossibleValues_inputToS1 = []
sixPossibleValues_outputFromS1 = []

for element in inputPairThatCausesDeltaY:
    sixPossibleValues_inputToS1.append(element[0])
    sixPossibleValues_outputFromS1.append(applyS1(element[0]))

print("Six possible values INPUT to S1 : {}".format(sixPossibleValues_inputToS1))
print("Six possible values OUTPUT from S1 : {}".format(sixPossibleValues_outputFrom
S1))

plaintexts_left = []
plaintexts_right = []
ciphertexts_left = []
ciphertexts_right = []

for element in allOutputsYieldingWantedOutputXOR:
    ciphertexts_left.append(element[0])
    ciphertexts_right.append(element[1])
for element in inputPairThatCausesDeltaY:
    plaintexts_left.append(element[0])
    plaintexts_right.append(element[1])

print("plaintexts left : {}".format(plaintexts_left))
print("plaintexts right : {}\n".format(plaintexts_right))
print("ciphertexts left : {}".format(ciphertexts_left))
print("ciphertexts right : {}".format(ciphertexts_right))

sk1freqs = [0]*16
sk2freqs = [0]*16

for leftPT in plaintexts_left:
    for element in sixPossibleValues_inputToS1:
        print("ELEMENT : {}".format(element))
        tempSK1 = []
        tempS1Output = []
        tempSK2 = []
        for k in range(0, 4):
            tempSK1.append(leftPT[k] ^ element[k])
```

```
        tempS1Output = (sixPossibleValues_outputFromS1[sixPossibleValues_inputToS1.
index(element)])
        temp_ciphertextLeft = ciphertexts_left[plaintexts_left.index(leftPT)]

        for l in range(0, 4):
            tempSK2.append(tempS1Output[l] ^ temp_ciphertextLeft[l])

        leftCT_test = encrypt_checkSubkeyValues(leftPT, tempSK1, tempSK2)
        rightCT_test = encrypt_checkSubkeyValues(plaintexts_right[plaintexts_left.i
ndex(leftPT)], tempSK1, tempSK2)

        outXORCheck = []
        for n in range(0, 4):
            outXORCheck.append(leftCT_test[n] ^ rightCT_test[n])

        for i in range(0, len(remainingInputs)):
            remLeftCT_test = encrypt_checkSubkeyValues(remainingInputs[i][0], tempS
K1, tempSK2)
            remRightCT_test = encrypt_checkSubkeyValues(remainingInputs[i][1], temp
SK1, tempSK2)

            if remLeftCT_test == remainingOutputsThatAreResultOfInputXOR[i][0] and
remRightCT_test == remainingOutputsThatAreResultOfInputXOR[i][1]:
                SK1_int = str(''.join(str(x) for x in tempSK1))
                SK1_int = '0b' + SK1_int
                SK1_int = int(SK1_int, 2)
                sk1freqs[SK1_int] += 1

                SK2_int = str(''.join(str(x) for x in tempSK2))
                SK2_int = '0b' + SK2_int
                SK2_int = int(SK2_int, 2)
                sk2freqs[SK2_int] += 1
        print("Current SK1 guess: {}".format(tempSK1))
        print("Current SK2 guess : {}".format(tempSK2))

print("\n\nSK1 frequencies : {}".format(sk1freqs))
print("SK2 frequencies : {}".format(sk2freqs))
```