SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Antonio Kovačić

DNA KRIPTOGRAFIJA

Diplomski rad

Voditelj rada:
prof. dr. sc. Andrej Dujella

Zagreb, srpanj, 2014.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Ovaj diplomski rad posvećujem svojim roditeljima, sestri, braći, prijateljima, mentoru, profesorima, kolegama s faksa, kao i svim ljudima koji su doprinijeli kako mom intelektualnom rastu, tako i mom rastu kao cjelovite osobe.

Sadržaj

Uvod

U današnje vrijeme svjedoci smo nagloga porasta razmjene podataka. Naglim napretkom današnjih računala, javila se potreba za povećanjem sigurnosti, odnosno zaštite podataka, koji putuju preko komunikacijskog kanala. Današnji kriptosustavi omogućuju siguran prijenos takvih podataka, a ključ njihovog razbijanja zapravo leži u faktoriziranju nekog *velikog* broja (na primjer RSA kriptosustav). Na današnjim računalima, takav problem nije lako riješiv - pa su ti sustavi još uvijek sigurni. Razvojem novih teoretskih modela računala - koji se pokušavaju i u praksi realizirati - uočeno je da problem faktorizacije neće biti više takav problem. Primjer jednog takvog računala je kvantno računalo za kojeg postoji algoritam (*Shorov algoritam*) koji faktorizira broj u polinomnom vremenu. Time se javila potreba za osmišljanjem novih teoretskih modela računala - odnosno kriptosustava - koji bi bili otporni na kvantno izračunavanje - ne bi se mogli probiti uporabom kvantnog računala u nekom razumnom vremenu. Takve kriptosustave ćemo zvati *kvantno rezistentnima*. Tema ovog diplomskog rada biti će DNA kriptografija. Kratko rečeno, radi se o teoretskom modelu kriptografskog sustava koji pomoću DNA izračunavanja šifrira podatke. Prednost takvog sustava jest upravo što je kvantno rezistentan.

U ovom radu najprije ćemo se ukratko upoznati s pojmom DNA računala, odnosno DNA izračunavanja, složenosti DNA računala te algoritmom za enkripciju, odnosno dekripciju podataka pomoću DNA računala.

Poglavlje 1

DNA računalo

1.1 DNA izračunljivost

DNA stroj, kao ni DNA izračunavanje nećemo striktno definirati već će definicija biti opisna - u definiciji ćemo reći koje operacije DNA stroj može izvršavati, i što pri tome mora biti zadovoljeno.
Prije nego definiramo DNA stroj moramo definirati neke pojmove iz logike sudova i kombinatorike.

Definicija 1.1.1. ***Alfabet*** *je proizvoljan konačan skup, čije elemente nazivamo* ***simboli****.*
Neka je $n \in \mathbb{N}$ *proizvoljan te A proizvoljan alfabet, proizvoljni element* $w \in A^n$ *zovemo* ***riječ alfabeta A****. Neka su* $s_1, ..., s_n \in A$*, riječ* $w = (s_1, ..., s_n)$ *alfabeta A još zapisujemo kao* $w = s_1 s_2 ... s_n$*. Smatramo da postoji riječ alfabeta A, koju ćemo označavati s* ε*, koja se ne sastoji ni od jednog simbola i zovemo je* ***prazna riječ****. Po dogovoru smatramo da je* $A^0 = \{\varepsilon\}$*. Skup svih riječi alfabeta A označavamo sa* A^**. Neka su* $a = a_1 ... a_m$*, te,* $b = b_1 ... b_k \in A^*$*, kažemo da je riječ* $c \in A^*$ *nastala* ***konkatenacijom*** *riječi a i b ako vrijedi* $c = ab = a_1 ... a_m b_1 ... b_k$*. Kažemo da je riječ* $c \in A^*$ *podriječ riječi* $a \in A^*$*, ako postoje riječi*

$b, d \in A^*$ *tako da je* $a = bcd$. ***Duljina riječi*** *se definira kao funkcija* $d : A^* \to \mathbb{N}$ *sa:*

$$d(\varepsilon) := 0$$
$$d(wa) := d(w) + 1$$

Definicija 1.1.2. *Neka je S proizvoljan konačan skup, a* $m : S \to \mathbb{N}$ *proizvoljna funkcija.* ***Multiskup M na skupu S*** *je uređeni par* $M = (S, m)$. *Za proizvoljan* $x \in S$, $m(x)$ *zovemo* ***kratnost*** *od x.* ***Kardinalnost multiskupa M*** *(broj elemenata), u oznaci* $|M|$, *se definira kao:*

$$|M| := \sum_{x \in S} m(x)$$

Definicija 1.1.3. ***DNA lanac*** *je proizvoljna riječ alfabeta {A (adenin), G (gvanin), T(timin), C (citozin)}. DNA stroj se sastoji od konstantnog broja konačnih skupova koje nazivamo* ***epruvete****, a čiji su elementi DNA lanci. Za proizvoljnu epruvetu K DNA stroja definiramo multiskup* $MulS(K)$ *kao multiskup svih riječi koje predstavljaju DNA lance sadržane u epruveti K. U DNA stroju su definirane slijedeće instrukcije:*

- *Kopiraj(*K_1,K_2*)* $\to$ *uz pretpostavku da je* $K_2 = \emptyset$ *, kopira MulS(*K_1*) u MulS(*K_2*) time više* K_2 *nije prazan*
- *Spoji(*K_1, K_2, K*)* $\to$ *uz pretpostavku da* $K=\emptyset$*:*

$$MulS(K) = MulS(K_1) \bigcup MulS(K_2)$$

- *Uoči(K)* $\to$ *ispituje je li* $MulS(K) \neq \emptyset$*, ako je rezultat operacije je* $\top$*, inače* $\bot$*. Također se može pročitati sadržaj epruvete* $MulS(K)$*.*
- *Odvoji(K,w)* $\to$ *za skup K i riječ w (iz* $MulS(K)$*) izbacuje sve riječi iz K koje kao podriječ ne sadrže riječ w*

- *Izvadi(K,w)* $\rightarrow K \backslash Odvoji(K,w)$ *→izbacuje sve riječi iz K koje sadrže w*
- *Odvoji_Pref(K,w)* $\rightarrow$ *izbacuje sve riječi iz K koje ne sadrže w kao prefiks*
- *Odvoji_Suff(K,w)* $\rightarrow$ *izbacuje sve riječi iz K koje ne sadrže w kao sufiks*
- *Proširi(K)* $\rightarrow$ *multiskupu MulS(K) još jednom dodaje elemente od K*
- *Izdvoji_po_duljini(K, l)* $\rightarrow$ *iz K izbacuje sve riječi čija je duljina različita od l*
- *Konkatenacija(K)* $\rightarrow$ *na slučajan način izvodi operaciju konkatenacije nad riječima iz MulS(K) tako da duljina novonastalih riječi ne bude veća od neke konstante, a vraća multiskup koji sadrži sve riječi nastale tom konkatenacijom. Vjerojatnost nastajanja duljih riječi je veća. Ukoliko MulS(K) prije izvođenja ove operacije nad epruvetom K sadrži veliki broj kopija svake od riječi, tada će MulS(K) nakon izvođenja ove operacije nad epruvetom K sadržavati sve moguće kombinacije elemenata iz K.*
  - ***Biološki komplement*** *DNA lanca H definiramo kao DNA lanac koja ima jednako znakova kao i H, ali je svaki znak A zamijenjen znakom T, a svaki znak C znakom G i obratno, i označavamo je s* $\overline{H}$
  - *Neka riječ H ima duljinu* $n \in 2\mathbb{N}$*, tada definiramo* ***biološki prefiks*** *riječi H kao biološki komplement riječi sastavljene od prvih* $\frac{n}{2}$ *znakova iz H, slično definiramo i* ***biološki sufiks*** *riječi H kao biološki komplement riječi sastavljene od zadnjih* $\frac{n}{2}$ *znakova riječi H*
  - *Smatramo da je operacija konkatenacije nad riječima H i J dopuštena ako postoji riječ L takva da je biološki sufiks od H prvih* $\frac{n}{2}$ *znakova od L, a biološki prefiks od J prvih* $\frac{n}{2}$ *znakova od L*
- *Izreži(K)* $\rightarrow$ *na slučajan način "skraćuje" riječi iz MulS(K) do neke fiksne duljine*

- *Izaberi(K)* $\rightarrow$ *na slučajan način iz MulS(K) izabire neku riječ te "generira" novi skup sastavljen od samo te riječi*

Program za DNA stroj *definiramo kao konačan niz gornje navedenih instrukcija. U svakom koraku programa se može izvesti točno jedna instrukcija. Kažemo da program P za DNA stroj* ***izračunava*** *funkciju* $f : S \subseteq N^k \rightarrow \mathbb{N}$ *ako vrijedi:*
$\vec{x} = (x_1, ..., x_k) \in S$ *ako i samo ako program P za DNA stroj s ulazom* $\vec{x}$ *(reprezentiranim pomoću DNA lanaca) u epruveti K završi s izvršavanjem te na kraju izvršavanja vrijedi Uoči(K)=* $\top$ *te je pri tome* $K = \{f(\vec{x})\}$.
Kažemo da je funkcija $f : \mathbb{N} \rightarrow \mathbb{N}$ ***DNA izračunljiva*** *ako postoji program za DNA stroj koji ju izračunava.*

Napomena 1.1.4. *Vidimo da se sve ove operacije izvršavaju nad jednom epruvetom u jednom koraku, odnosno multiskupom* $MulS(K)$*. Što je veća kardinalnost multiskupa* $MulS(K)$*, to se više operacija na riječima izvrši istovremeno, a u stvarnom svijetu sve te operacije imaju svoje "biokemijske analogone" - biokemijske reakcije. Takvo računalo zapravo možemo interpretirati kao superračunalo s izuzetno velikim brojem procesora. U pozadini svega toga se zapravo krije masivni paralelizam. Memoriju DNA računala zapravo predstavljaju epruvete. Jasno je odakle naziv epruvete.*
Uočimoo da je $MulS(K)$ *definiran nad konačnim skupom pa je i on konačan - no vidimo da se on zapravo može proširiti nizom operacija Proširi tako da je njegov kardinalitet izrazito velikog reda (nadeksponencijalnog) , ali u praksi se već sada zaključuje da to neće biti uvijek moguće - naime broj DNA lanaca u epruveti (laboratorijskoj) biti će ograničen volumenom te epruvete.*
Uočimo da operacija konkatenacije uključuje vjerojatnosni efekt - vjerojatnost nastajanja duljih riječi konkatenacijom je veća - odnosno dvije riječi iz skupa K koje će se konkatenirati neće biti izabrane na slučajan način - već tako da se pokuša dobiti riječ maksimalne duljine (maksimalna duljina je određena nekom konstantom). Iz toga očito možemo vidjeti

da sam ishod DNA računanja nije sasvim siguran - no u praksi se pokazuje (pri sintezi DNA lanaca) da je to moguće - u tu svrhu je i uvedena pretpostavka da će se, ukoliko $MulS(K)$ sadrži velik broj kopija od svake riječi iz K, dobiti svaka moguća konkatenacija riječi iz K.

1.2 O složenosti DNA računala

Kako bi nešto rekli o složenosti DNA računala, najprije ćemo navesti nekoliko osnovnih definicija iz teorije složenosti algoritama, odnosno referencirati se na [6].

Definicija 1.2.1. ***Turingov stroj*** *je uređena sedmorka $(Q, \Sigma, \Gamma, \delta, q_0, q_{DA}, q_{NE})$, gdje je redom:*

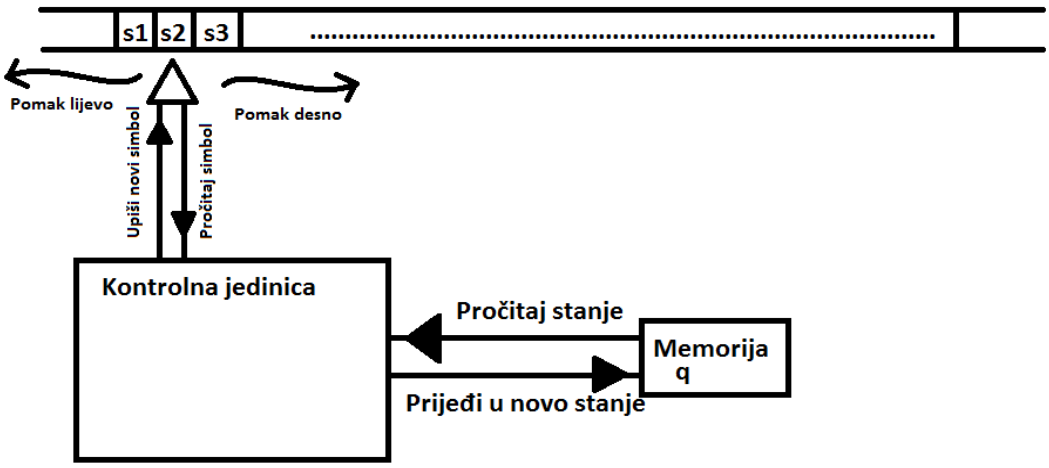
- *Q konačan skup čije elemente nazivamo stanja*
- *Σ je konačan skup, čije elemente nazivamo ulazni simboli, pretpostavljamo da Σ ne sadrži "prazan simbol" kojeg označavamo sa ε*
- *Γ je konačan skup kojeg nazivamo alfabet Turingovog stroja, pretpostavljamo da je $\varepsilon \in \Gamma$, te $\Sigma \subset \Gamma$*
- *$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, D, S\}$ koju nazivamo funkcija prijelaza*
- *$q_0 \in Q$ nazivamo početnim stanjem*
- *$q_{DA} \in Q$ nazivamo stanjem prihvaćanja*
- *$q_{NE} \in Q$ nazivamo stanjem odbijanja, te $q_{NE} \neq q_{DA}$*

Napomena 1.2.2. *(Opis rada Turingovog stroja)*
Turingov stroj zapravo ima četiri glavna dijela: kontrolnu jedinicu (koja zapravo oponaša dijelovanje funkcije δ), beskonačnu traku, neograničenu s lijeve i desne strane, takvu da se

u svakom trenutku rada stroja na jednom registru trake nalazi točno jedan simbol, memoriju u kojoj se pamti trenutačno stanje stroja te glavu za čitanje koja se u jednom koraku rada stroja može pomicati za točno jedno mjesto na traci: desno, lijevo ili ostati na istom simbolu. Glava se na početku nalazi na nekom mjestu na traci (unaprijed definiranom), zatim čita simbol. Pročitani simbol, u paru s trenutnim stanjem stroja "se šalje" u kontrolnu jedinicu. Glava nakon toga, najprije zamijeni pročitani simbol nekim drugim simbolom, stroj prelazi u novo stanje, a glava se pomiče na drugi registar (L (lijevo), D (desno)) ili ostaje na istom mjestu (S).

Vidimo da opisani Turingov stroj može stati u dva završna stanja q_{DA}, odnosno q_{NE}, takav Turingov stroj se naziva još odlučitelj. Uočimo da Turingov stroj ne mora nužno uvijek stati. Shematski prikaz Turingovog stroja možete vidjeti na slici 1.1.

Nedeterministički Turingov stroj se definira na analogan način, samo što je funkcija prijelaza definirana sa: $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, D, S\})$.



Slika 1.1: Shematski prikaz Turingovog stroja

Definicija 1.2.3. *Neka su* $f, g : \mathbb{N} \to \mathbb{R}^+$ *dvije funkcije. Kažemo da je funkcija g* ***asimp-***

totska gornja međa *za funkciju f ako postoje $c > 0$ i $n_0 \in \mathbb{N}$ tako da za svaki $n \geq n_0$ vrijedi*

$$f(n) \leq cg(n)$$

Činjenicu da je g asimptotska međa od f označavamo sa $f(n) = O(g(n)$.

Osnovne definicije (što je alfabet logike sudova, interpretacija, ispunjivost formule, konjunktivna, odnosno, disjunktivna normalna forma i tako dalje) se mogu naći u [7, s. 12-25].
Više o Turingovom stroju te nekim pojmovima na koje se pozivamo u idućim rezultatima se mogu naći u:

- Turing prepoznatiljivost, Turing odlučivost [6, s. 141-142]
- Vremenska složenost Turingovog stroja determinističkog se može naći u [6, s. 248], a nedeterminističkog u [6, s. 255]
- Klase vremenske složenosti:
  - $TIME(f(n))$ u [6, s. 251]
  - P u [6, s. 258]
  - Vezano uz klasu NP u [6, s. 265-267]
- Polinomna reducibilnost u [6, s. 272]
- NP-potpunost u [6, s. 276]

Označimo sa SAT skup definiran na idući način:

$$SAT = \{F : F \text{ je ispunjiva formula logike sudova } \}$$

Formulacija *problema SAT* glasi:

Za danu formulu logike sudova F koja je u konjunktivnoj normalnoj formi odrediti vrijedi li $F \in SAT$.

Konjunktivnu normalnu formu koja u svakoj svojoj elementarnoj disjunkciji sadrži točno $k \in \mathbb{N}\backslash\{0\}$ literala nazivamo k-knf. Formulacija problema $k - SAT$ glasi:

Za proizvoljnu formulu F koja je k-knf odrediti je li F ispunjiva.

U [6, s. 276-283] se može vidjeti da je problem SAT *NP-potpun* problem, kao i $3 - SAT$. Sljedeći teorem govori zapravo o tome da DNA računala, u pogledu vremenske složenosti, imaju bolja svojstva nego deterministički Turnigovi strojevi:

Teorem 1.2.4. *(Lipton) Za svaku konjunktivnu normalnu formu F u kojoj se pojavljuje n propozicionalih varijabli i m klauzula, u $O(m+1)$ separacija i $O(m)$ spajanja te jednim uočavanjem možemo odlučiti vrijedi li $F \in SAT$*

Dokaz navedenog teorema se može naći u [5].
S pogleda odlučivosti jezika ipak nemamo takav rezultat, odnosno postoji slutnja koja kaže:

Slutnja 1.2.5. *(Kvantna i biološka slutnja) Problem je odlučiv na DNA računalu ili kvantnom računalu ako i samo ako je Turing-odlučiv.*

Postavlja se prirodno pitanje kako izračunati složenost DNA računala. Odgovor je jednostavan: složenost DNA računala procijenjujemo brojem instrukcija koje DNA stroj izvrši, te s kardinalosti skupa $MulS(K)$. Zbog toga što kardinalnost skupa $MulS(K)$ može izrazito brzo rasti (samo jedna operacija *Proširi(K)*, za pripadnu funkciju kratnosti m multiskupa $MulS(K)$ vrijedi da je: $m_{nova}(x) = 2 \cdot m_{stara}(x)$, gdje je $m_{nova}(x)$ kratnost od x nakon

izvršenja operacije *Proširi*,a $m_{stara}(x)$ kratnost od x prije izvršenja te iste operacije) prostorna složenost nekog programa za DNA stroj obično doseže nadeksponencijalnu veličinu (vidjet ćemo u idućem podpoglavlju takav slučaj).
U sljedećem potpoglavlju ćemo procijeniti složenost jednog programa za DNA stroj.

Problem Hamiltonovog puta

Definicija 1.2.6. ***Konačan usmjereni graf*** *je uređeni par $G = (V, E)$ gdje je V proizvoljan konačan skup čije elemente nazivamo* ***vrhovi****, a $E \subseteq V \times V$ skup čije elemente nazivamo* ***bridovi****. Ako je $E = V \times V$ kažemo da je usmjereni graf G* ***potpuni graf****. Kažemo da je brid e* ***petlja*** *ako vrijedi: $(\exists x \in V) : e = (x, x)$.* ***Šetnja u grafu*** *G je $2n + 1$-torka $(v_0, e_0, v_1, e_1, ..., v_{n-1}, e_{n-1}, v_n)$, pri čemu vrijedi:*

- $(\forall i \in \{0, ..., n\}) \quad v_i \in V$
- $(\forall i \in \{0, ..., n-1\}) \quad e_i \in E$
- $e_i = (v_i, v_{i+1}), \forall i \in \{0, ..., n-1\}$

Kažemo da šetnja $(v_0, e_0, v_1, e_1, ..., v_{n-1}, e_{n-1}, v_n)$ ***prolazi kroz vrh*** *$x \in V$ ako postoji $i \in \{0, ..., n\}$ takav da je $x = v_i$, te da šetnja* ***počinje*** *s vrhom v_0 i* ***završava*** *s vrhom v_n. Duljina šetnje se definira kao broj bridova koji se pojavljuju u njoj.*
Staza *u grafu je šetnja $(v_0, e_0, v_1, e_1, ..., v_{n-1}, e_{n-1}, v_n)$ za koju vrijedi*

$$(\forall i, j \in \{0, ..., n-1\})\,(i \neq j) \rightarrow e_i \neq e_j$$

Put *u grafu je šetnja $(v_0, e_0, v_1, e_1, ..., v_{n-1}, e_{n-1}, v_n)$ za koju vrijedi:*

$$(\forall i, j \in \{0, ..., n\})\,(i \neq j) \rightarrow v_i \neq v_j$$

Hamiltonov put *je put koji prolazi kroz sve vrhove grafa G.*

Napomena 1.2.7. *Neusmjereni graf se definira analogno, ali se skup bridova definira kao:*

$$E \subseteq \{\{x, y\} : x, y \in V\}$$

Također, radi jednostavnosti, pretpostavili smo da između dva vrha $x, y \in V$ može biti najviše dva usmjerena brida i u tom slučaju vrijedi: $(x, y) \in E$ *i* $(y, x) \in E$.

Problem Hamiltonovog puta glasi:

Postoji li u proizvoljnom konačnom grafu $G = (V, E)$ za vrhove $x, y \in V$ Hamiltonov put koji počinje s x, a završava s y.

U [6, s. 286-291] se može vidjeti da je problem Hamiltonovog puta NP-potpun problem. U ovom poglavlju analiziramo *Adlemanov algoritam* koji rješava problem u $O(n \log(n))$ operacija. U kasnijim poglavljima ćemo obrazložiti reprezentaciju podataka pomoću DNA lanaca, za sada ćemo samo reći da su naši podaci reprezentirani lancima parne duljine l. Sada ćemo prezentirat Adlemanov algoriatam za traženje Hamiltonovog puta koji počinje s vrhom v_{in}, a završava s v_{out} u usmjerenom označenom grafu $G = (V, E)$. Ali prije toga ćemo reći nešto o vezi bridova i vrhova. Ako su dani vrhovi A i B i reprezentirani riječima H i J, tada je brid (A, B) reprezentiran riječju koja je nastala konkatenacijom (u smislu operacije nad riječima) biološkog sufiksa riječi H i biološkog prefiksa riječi J. Kako bi mogli razlikovati koji brid povezuje koje vrhove, uviđamo da svaki vrh mora imati jedinstveni prefiks i sufiks, a ne samo jedinstven prikaz jednom riječju. Nakon što smo objasnili vezu između bridova i vrhova moramo najprije "pripremiti" epruvetu za algoritam.

$$K = V \cup E$$

U početku je upravo $MulS(K) = K$.

Adlemanov algoritam:

1. Ulaz: Graf $G = (V, E)$, $|V| = n$, $v_1 = v_{in}$ vrh iz kojeg krećemo, $v_n = v_{out}$ vrh u kojem završavamo, stavi vrhove i bridove u K

2. $\lceil 2n \log_2(n) \rceil$ puta primjeni operaciju *Proširi(K)* tako da dobiješ barem $2^{2n \log_2(n)} = n^{2n}$ kopija svake riječi u $MulS(K)$

3. Primjeni operaciju *Konkatenacija(K)* da dobiješ šetnju u G, tako da duljina šetnje bude manja od n - broj bridova u šetnji može biti manji ili jednak n

4. Primjeni *Odvoji_Pref(K, v_{in})*: izbacujemo one šetnje koje ne počinju vrhom v_{in}

5. *Odvoji_Suff(K, v_{out})* : izbacujemo one šetnje koje ne završavaju s v_{out}

6. Primjeni operaciju *Izdvoji_po _duljini(K, $l \cdot n + l \cdot (n-1)$)* da iz $K(MulS(K))$ izbaciš sve one riječi koje u sebi ne sadrže točno n vrhova i $n-1$ bridova (šetnje čija je duljina točno $n-1$)

7. na v_i primjeni operaciju *Odvoji(K, v_i)* , $\forall i \in \{2, 3, ..., n-1\}$: iz $MulS(K)$ ukloni sve one šetnje u kojima se neki od vrhova ne pojavljuje

8. *Uoči(K)*: postoji li Hamiltonov put

Nama zapravo bridovi u ovom algoritmu, na ovaj način konstruirani daju mogućnost povezivanja dva vrha koja su povezana nekim birdom (u smislu biokemije, igraju ulogu enzima inače se vrhovi ne bi mogli povezati).
Brojimo korake algoritma:

- 2: $\lceil 2n \log_2(n) \rceil$ koraka
- 3-6: Po jedan korak svaka operacija

- 7: $n-2$ koraka
- 8: jedan korak

Ukupno: $\lceil 2n\log_2(n)\rceil + n - 2 + 5 = \lceil 2n\log_2(n)\rceil + n + 3 = O(n\log(n))$ operacija. No, rekli smo da se složenost DNA stroja mjeri i kardinalnošću skupa $MulS(K)$ koji u jednom trenutku sadrži i n^{2n} elemenata. Još je preostalo dokazati da algoritam radi:

Teorem 1.2.8. *Neka je G=(V,E) usmjeren označen graf, te v_{in} i v_{out} elementi iz V, tada Adlemanov algoritam odlučuje postoji li u usmjerenom grafu G=(V,E) Hamiltonov put od v_{in} do v_{out}.*

Dokaz.

- $|V| = n$, $K = V \cup E$ gdje smatramo da je svakom vrhu dodijeljen jedinstven prefiks i sufiks. Neka je minimalni DNA lanac duljine l.
- Definiramo rekurzivno skupove $MulS(K_n)$ odakle ćemo zapravo izvući kako izgleda naš skup $MulS(K)$ nakon primjene operacije *Proširi(K)* $\lceil 2n\log_2(n)\rceil$ puta

$$K_0 = K \rightarrow MulS(K_0) = K_0$$

$$MulS(K_{n+1}) = MulS(K_n) \cup MulS(K_n), n \in \mathbb{N}$$

Nakon ovog koraka, redefiniramo skup MulS(K)

$$MulS(K) = MulS(K_{\lceil 2n\log_2(n)\rceil})$$

Zapravo sada trebamo dokazati da je $2^{\lceil 2n\log_2(n)\rceil}$ dovoljan broj kopija skupa K za kreiranje svih šetnji u grafu:

$$2^{\lceil 2n\log_2(n)\rceil} \geq 2^{2n\log_2(n)} = n^{2n}$$

pa je dovoljno pokazati da je n^{2n} dovoljan broj kopija skupa K od kojih možemo kreirati sve šetnje u grafu. Bez smanjenja općenitosti u tu svrhu možemo pretpostaviti da je G potpuno povezan usmjeren graf (dakle svaki brid je povezan sa svakim u oba smjera). Zašto? Jer ako G nije potpuno povezan onda ima manji broj šetnji od potpuno povezanog usmjerenog grafa.
U tu svrhu definiramo skup A^k:

$$A^k = \{(b_1, ..., b_k) : b_i \in V\}$$

Uvidimo da smo u skupu A^k dozvolili i petlje! Dakle može postojati brid (v_i, v_i). Kada to ne bi dozvolili, na uređenu k-torku bi samo još stavili uvjet da je $b_i \neq b_{i+1}, \quad \forall i \in \{1, ..., k-1\}$. Sada, jer između svaka 2 vrha ima točno 1 brid za svaki smjer koji povezuje te vrhove, vidimo da su sve šetnje duljine $k-1$ jedinstveno određene skupom A^k. Pa su sve šetnje do duljine $n-1$ (jer ćemo tako izabrati operaciju konkatenacije da kreira šetnje duljine ne duže od $n-1$) reprezentirane idućim skupom:

$$\bigcup_{k=1}^{n} A^k$$

Preostalo je dokazati da kardinalnost gornjeg skupa nije veća od n^{2n}. Kardinalnost skupa A^k je lako odrediti. Naime za prvu komponentu uređene k-torke ima n mogućnosti, za 2 isto n, općenito za i-tu komponentu ima n mogućnosti.

$$|A^k| = n^k \rightarrow \left|\bigcup_{k=1}^{n} A^k\right| = \sum_{k=1}^{n} |A^k| = \sum_{k=1}^{n} n^k = \frac{n(n^n-1)}{n-1}$$

$$\frac{n(n^n-1)}{n-1} \leq \frac{n \cdot n^n}{n-1} \leq n \cdot n^n = n^{n+1} \leq n^{2n}$$

- Primjenom operacije *Konkatenacija(K)* dobili smo sve moguće šetnje u grafu G

(spremljene u $MulS(K)$)

- Operacijama *Odvoji_Pref(K, v_{in})* i *Odvoji_Suff(K, v_{out})* iz skupa $MulS(K)$ izbacujemo sve one šetnje koje ne počinju vrhovima v_{in} i v_{out}

- operacijom *Izdvoji_po_duljini(K, $l \cdot n + l \cdot (n-1)$)* uklanjamo sve preostale bridove i one šetnje čija je duljina strogo manja od $n-1$. Sada su ostale šetnje duljine $n-1$, ali to još nisu putevi (a onda ni Hamiltonovi putevi). Kako ima n vrhova, a šetnja je duljine $n-1$, to znači da je u šetnji točno n vrhova kroz koje šetnja prolazi, ako se neki vrh ne nalazi u šetnji, to znači da se neki drugi vrh pojavljuje dva puta. A kako smo već uklonili one šetnje koje ne počinju s v_{in} i ne završavaju s v_{out} jedino je preostalo ukloniti sve one šetnje koje ne sadrže neki $v_i \in V \backslash \{v_{in}, v_{out}\}$.

- Za svaki $x \in V \backslash \{v_{in}, v_{out}\}$ čini:

$$Odvoji(K, x)$$

- Ovim su korakom zapravo u $MulS(K)$ ostali samo Hamiltonovi putevi koji počinju s v_{in} i završavaju s v_{out}, ako takvih ima, operacijom *Uoči(K)* dobivamo rješenje.

□

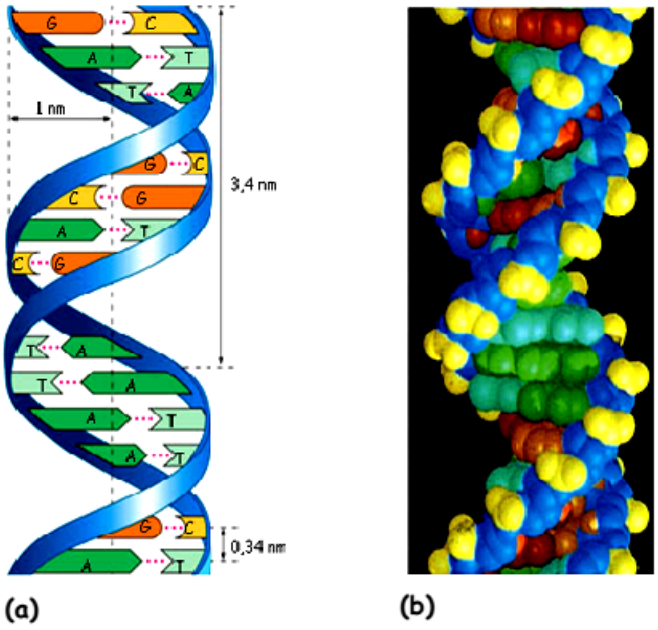
Poglavlje 2

DNA kriptografija

2.1 Osnovni biološki koncepti deoksiribonukleinske kiseline

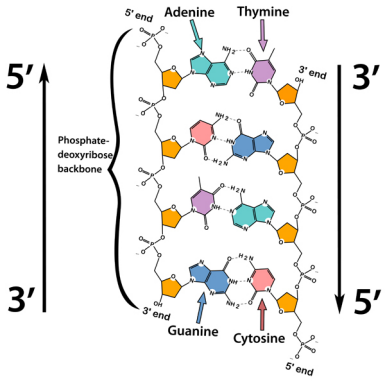
Deoksiribonukleinska kiselina (DNA) sastavljena je od dva duga lanca nukleotida (polinukleotidni lanci) koji su omotani jedan oko drugoga, odnosno ima strukturu *dvostruke uzvojnice* (engleski **double helix**) (slika 2.1). **Nukleotid** je osnovna građevna jedinica DNA, a gradi ju jedna od četiri dušićne baze: *adenina (A), gvanina (G), timina (T) i citozina (C)*, šećera pentoze te fosfatne skupine. Sa $1' - 5'$ označavamo ugljikove atome u molekuli šećera, a na slici 2.4 se može vidjeti atomska struktura nukleotida. Vidimo da ako se na $2'$ veže hidroksilna skupina - govorimo o **ribonukleinskoj kiselini**, u kojoj se timin zamjenjuje *uracinom (U)*, inače, ako se na $2'$ veže vodik, govorimo o DNA (pripadna pentoza je deoksiriboza). Veza između dva nukleotida je kovalentna, a nastaje tako da se hidroksilna skupina na $3'$ ugljikovom atomu pentoze veže s fosfatnom skupinom drugog nukleotida koja se nalazi na $5'$ ugljikovom atomu pripadne pentoze. Pripadna dva lanca koja sastavljaju jednu molekulu DNA su *antiparalelni* što znači da moraju biti suprotne orijentacije - jedan lanac dvostruke uzvojnice mora završavati s 5' ugljikovim atomom kao

krajem, a druga zavravati s 3' ugljikovim atomom kao kraje (slika 2.3). Pri povezivanju pripadnih dušićnih baza uvijek se adenin povezuje s timinom, a gvanin s citozinom. Sada kada smo osbjasnili neke osnovne principe DNA molekule, komentirat ćemo neka tehnološka postignuća koja su biokemijski analogoni operacija DNA stroja navedenih u poglavlju 1.1. *DNA sintetizator* je stroj koji kemijski sintetizira DNA lance. Umjetne jednolančane DNA (signle stranded DNA) koje ćemo kasnije označavati sa *ssDNA* nazivamo oligonukleotide. Dvostruke uzvojnice ćemo nadalje označavati sa *dsDNA*. U određenim uvjetima, komplementarne ssDNA mogu oformiti dsDNA, a taj proces se zove *hibridizacija*. (Navesti ostale biokemijske operacije koje se mogu upotrebljavati).

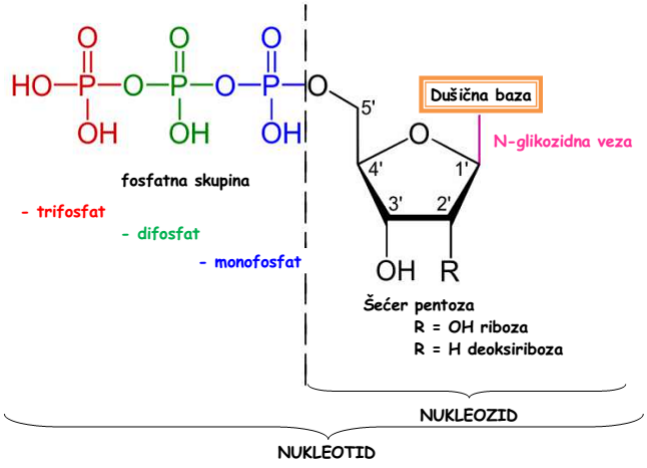


Slika 2.1: Struktura DNA: a) 2D model, b) 3D model

Slika 2.2: Hibridizacija



Slika 2.3: Antiparalelnost DNA



Slika 2.4: Nukleotid

2.2 Reprezentacija podataka DNA lancima

Reprezentacija podataka DNA lancima se vrši na idući način:

1. Ulaz: Tekst T

2. Tekst $T \rightarrow$ ASCII T_A (Decimalni ili Hex)

3. ASCII $T_A \rightarrow$ 8-bitni binarni prikaz T_B

4. Svaka 2 bita zamijeni pripadnom dušićnom bazom prema 2.1

$\mathbf{x} \in \{0,1\}^2$	**Dušićna baza**
00	A
01	C
10	G
11	T

Tablica 2.1: Reprezentacija uređenog para bitova sa dušićnim bazama

Primjer 2.2.1. *Reprezentirajmo "DNA" sa DNA lancem:*

Tekst	ASCII kod	Binarni prikaz	DNA prikaz
D	44	01 00 01 00	CACA
N	4E	01 00 11 10	CATG
A	41	01 00 00 01	CAAC

Tablica 2.2: Reprezentacija riječi "DNA" DNA lancem

Iz 2.2 slijedi: $"DNA" = (CACACATGCAAC)_{DNA}$

Naravno, vidimo da bi ovakvu "šifru" bilo iznimno lako probiti pa reprezentaciju podataka nećemo razmatrati kao potencijalni kriptosustav jer bi njegova sigurnost bila izuzetno mala.

2.3 Osnove kriptografije

Kriptografija je znanstvena disciplina koja proučava metode koje u nesigurnom komunikacijskom kanalu čuvaju integritet i sigurnost podataka i poruka te skrivaju njihov sadržaj od treće osobe, a tako da dane poruke može pročitati samo onaj kome su namijenjene. Razmatrat ćemo komunikaciju između dvoje sudionika: *pošiljatelja (Alice)* i *primatelja (Bob)*. Alice želi preko nesigurnog komunikacijskog kanala poslati poruku, tako da treća osoba (Oscar ili Eva) ne može saznati sadržaj poruke. Poruku zovemo *otvoreni tekst*. Kako bi sakrila sadržaj poruke, Alice transformira otvoreni tekst uporabom *ključa*. Cijeli postupak transformacije sa ključem nazivamo *šifriranje*, a rezultat šifriranja *šifrat*. Alice šalje Bobu šifrat preko nesigurnog komunikacijskog kanala. Treća osoba pokuašava ukrasti šifrat te saznati njegov sadržaj, no ako je Alice koristila "dobro" šifriranje, treća osoba to neće bit u mogućnosti napraviti. Kada poruka dođe do Boba, on će, u slučaju da ima odgovarajući ključ, šifrat ponovno transformirati u otvoreni tekst, što zovemo *dešifriranjem*. *Kriptoanaliza (dekriptiranje)* je znanstvena disciplina koja proučava metode za čitanje skrivenih poruka bez poznavanja ključa. Kriptologija je znanstvena diciplina koja objedinjuje i kriptografiju i kriptoanalizu. Često za šifriranje odnosno dešifriranje koristimo određene postupke - *kriptografske algoritme (šifre)*. Ti postupci su zapravo određene funkcije - funkcija *enkripcije* općenito otvorenom tekstu (njegovim sastavnim jedinicama) pridružuje osnovne sastavnice šifrata, dok funkcija *dekripcije* opet sastavnim jedinicama šifrata pridružuje sastavne jedinice otvorenog teksta. *Prostor ključeva* je skup čiji su elementi svi mogući ključevi. *Kriptosustav* objedinjuje kriptografski algoritam, šifrate, ključeve te sve moguće otvorene tekstove. Sada navodimo formalnu definiciju kriptosustava:

Definicija 2.3.1. *Kriptosustav je uređena petorka* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ *gdje su redom:*

- $\mathcal{P}$ *konačan skup čije elemente nazivamo simbolima otvorenog teksta*
- $\mathcal{C}$ *konačan skup čije elemente nazivamo simbolima šifrata*

- $\mathcal{K}$ *je konačan skup čije elemente nazivamo ključevi*
- $\mathcal{E} = \{e : e$ *je funkcija definirana sa* $e : \mathcal{P} \to C\}$ *i* $\mathcal{D} = \{d : d$ *je funkcija definirana sa* $d : C \to \mathcal{P}\}$ *su skupovi funkcija za koje vrijedi:*

$$(\forall K \in \mathcal{K})(\exists e_K \in \mathcal{E})(\exists d_K \in \mathcal{D}) : d_K(e_K(x)) = x, \forall x \in \mathcal{P}$$

Napomena 2.3.2. *U primjeni se često radi jednostavnosti kaže da je* $\mathcal{P}$ *konačan skup osnovnih elemenata otvorenog teksta, a* C *skup osnovnih elemenata šifrata. U zadnjoj točki definicije, radi jednostavnosti, navodi da je* x *otvoreni tekst (odnosno* $x \in \mathcal{P}^k$*, za neki* $k \in \mathbb{N}$*). Naravno, to je zbog toga što se prirodno može uvesti "proširenje" funkcije* e_K *i to na sljedeći način:*
za $\vec{x} = (x_1, ..., x_n) \in \mathcal{P}^n$ *definira se da je*

$$e_K(\vec{x}) = e_K(x_1)e_K(x_2)...e_K(x_n)$$

Analogno bi se "proširila" funkcija d_K*.*

Od načina (tipa operacija) šifriranja razlikujemo *supstitucijske šifre* (svaki element otvorenog teksta se zamijenjuje nekim drugim tekstom, npr. "DNA" → "GRD"), *transpozicijske šifre* u kojoj se permutiraju slova otvorenog teksta (npr. "DNA" → "NDA"). Postoje i kriptosustavi koji koriste i jednu i drugu metodu. Otvoreni tekst možemo pak podijeliti na blokove pa razlikujemo *blokovne šifre* i *protočne šifre* gdje šifriramo element po element otvorenog teksta. Prema javnosti ključeva razlikuejmo *simetrične kriptosustave* i *kriptosustave s javnim ključem*. U simetričnim kriptosustavima se ključ za dešifriranje može saznati iz ključa za šifriranje, pa je taj ključ *tajan*, zbog čega ove kriptosustave nazivamo i *kriptosustavi s tajnim ključem*. U drugom navedenom postoje dva ključa i ključ za dešifriranje se ne može izvesti iz ključa za šifriranje u nekom razumnom vremenu. U

daljnjim razmatranjima pretpostavljat ćemo takozvano Kerckhoffsovo načelo:

Kriptosustav bi trebao biti siguran čak i ako se sve zna o njemu osim ključa.

Pa ćemo pretpostaviti da napadač na kriptosustav zna koji alat za šifriranje koristimo, posjeduje šifrat i njemu odgovarajući otvoreni tekst, pretpostavljamo da ima mogućnost odabira otvorenog teksta i njemu pripadajućeg šifrata pa čak i da ima alat za dešifriranje na temelju kojeg iz danog šifrata može dobiti otvoreni tekst (cilj mu je saznati ključ za dešifriranje). Više o kriptografiji se može naći u [3]

2.4 DES kriptosustav

Kratki prgled DES kriptosustava

"Krajem 60-tih i početkom 70-tih godina 20. stoljeća, razvojem financijskih transakcija, kriptografija postaje zanimljiva sve većem broju potencijalnih korisnika. Dotad je glavna primjena kriptografije bila u vojne i diplomatske svrhe, pa je bilo normalno da svaka država (ili čak svaka zainteresirana državna organizacija) koristi svoju šifru za koju je vjerovala da je najbolja. No, tada se pojavila potreba za šifrom koju će moći koristiti korisnici širom svijeta, i u koju će svi oni moći imati povjerenje - dakle, pojavila se potreba uvođenja standarda u kriptografiji.

Godine 1972. američki National Bureau of Standards (NBS) inicirao je program za zaštitu računalnih i komunikacijskih podataka. Jedan je od ciljeva bio razvijanje jednog standardnog kriptosustava. Godine 1973. NSB je raspisao javni natječaj za takav kriptosustav. Taj kriptosustav je trebao zadovoljiti sljedeće uvjete:

- visoki stupanj sigurnosti
- potpuna specifikacija i lako razumijevanje algoritma

- sigurnost leži u ključu, a ne u tajnosti algoritma
- dostupnost svim korisnicima
- prilagodljivost uporabi u različitim primjenama
- ekonomičnost implementacije u elektoničkim uređajima
- efikasnost
- mogućnost provjere
- mogućnost izvoza (zbog američkih zakona)

Na tom natječaju niti jedan prijedlog nije zadovoljavao sve ove zahtjeve. Međutim, na ponovljeni natječaj iduće godine pristigao je prijedlog algoritma koji je razvio IBM-ov tim kriptografa. Algoritam je zasnovan na tzv. Feistelovoj šifri. Gotovo svi simetrični blokovni algoritmi koji su danas u uporabi koriste ideju koju je uveo Horst Feistel 1973. godine. Jedna od glavnih ideja je alternirana uporaba supstitucija i transpozicija kroz više iteracija (tzv. rundi). Predloženi algoritam je nakon nekih preinaka, u kojima je sudjelovala i National Security Agency (NSA), prihvaćen kao standard 1976. godine i dobio je ime Data Encryption Standard (DES)."[1]

Pogledajmo sada kako se kriptira DES algoritmom: Pretpostavljamo da je duljina otvorenog teksta x kojeg DES šifrira duljine 64 bita. DES koristi ključ $K = k_1...k_{56}$ duljine 56 bita. Neka je $x = x_1...x_{64}$ otvoreni tekst. Najprije permutiramo x fiksnom *inicijalnom permutacijom* IP te dobivamo $x_0 = IP(x)$ - IP i-ti bit od x zamijenjuje sa $IP(i)$-tim bitom od x. Inicijalna permutacija IP je prikazana na slici 2.5.

[1] [3, s. ?]

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

Slika 2.5: Inicijalna permutacija

Zatim napišemo x_0 u obliku konkatenacije dva *stringa bitova* koji su duljine 32 bita svaki: $x_0 = L_0R_0$ Nakon toga računamo L_i i R_i, gdje je:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

Gdje su $K_1, ..., K_{16}$ *riječi bitova* duljine 48, dobiveni permutiranjem nekih bitova iz K. $\oplus$ je operacija *ekskluzivno ili* opisana slikom 2.6

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

Slika 2.6: Operacija *ekskluzivno ili*

Sada ćemo opisati dijelovanje funkcije $f : \{0,1\}^{32} \times \{0,1\}^{48} \to \{0,1\}^{32}$. Pretpostavimo da su argumenti funkcije riječ A duljine 32 i J duljine 48. Prvo niz A proširimo do riječi bitova duljine 48 primjenjujući funkciju $E : \{0,1\}^{32} \to \{0,1\}^{48}$ danu slikom 2.7 koja i-tom bitu, pridruži $E(i)$-ti bit riječi A koji se nalazi na i-tom mjestu tablice čitajući s lijeva na desno. Očigledno je da se 16 bitova riječi A "ponavlja". Uzmemo da je $B = E(A) \oplus$

E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Slika 2.7: Preslikavanje E

J te prikažemo B kao konkatenaciju osam riječi sastavljenih od bitova koje su duljine šest, odnosno: $B = B_0B_1B_2B_3B_4B_5B_6B_7B_8$. U sljedećem koraku algoritma koristimo $S-$*kutije* (*supstitucijske kutije*) $S_1, ..., S_8$ (prikazane na slici 2.12), od kojih je svaki $S_i \in \{0, ..., 15\}^{4\times16}$ matrica sa 4 retka i 16 stupaca čiji su elementi iz skupa $\{0, ..., 15\}$. Neka je svaki $B_j = b_1^{(j)}...b_6^{(j)}$. Sada računamo $S_j(B_j)$ prema sljedećem opisu:
$r_j = b_1^{(j)}b_6^{(j)}$ binarni zapis r_j-tog retka od S_j, a $c_j = b_2^{(j)}b_3^{(j)}b_4^{(j)}b_5^{(j)}$ binarni zapis c_j-tog stupca matrice S_j. Definiramo: $C_j = S_j(B_j) = S_j(r_j, c_j), j = 1, ..., 8$ zapisano kao riječ (broj) sastavljenu od 4 bita. Sada riječ $C = C_1C_2C_3C_4C_5C_6C_7C_8$ sastavljenu od 32 bita permutiramo pomoću fiksne *završne permutacije* P prikazane na slici 2.8. U konačnici, vrijedi da je $f(A, J) = P(C)$.

P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Slika 2.8: Završna permutacija

Sada opisujemo računanje $K_1, ..., K_{16}$ iz $K = k_1...k_{64}$. Bitovi, čiji je indeks $l * 8, l = 1, ..., 8$ služe za testiranje pariteta, definirani su tako da svakih osam bitova (jedan bajt), čitajući s lijeva na desno, sadrži neparan broj jedinica. Navedene (paritetne) bitove ignoriramo kod računanja tablice ključeva, a preostale bitove permutiramo pomoću fiksne permutacije $PC1$ dane slikom 2.9. Neka je sada $PC1(K) = C_0D_0$, tako da je $d(C_0) = d(D_0) = 28$. Sada za svaki $i \in \{1, ..., 16\}$ računamo:

$$C_i = \begin{cases} {}^{1}\overleftarrow{C_{i-1}} & \text{ako } i = 1, 2, 9, 16 \\ {}^{2}\overleftarrow{C_{i-1}} & \text{inače} \end{cases}$$

$$D_i = \begin{cases} {}^{1}\overleftarrow{D_{i-1}} & \text{ako } i = 1, 2, 9, 16 \\ {}^{2}\overleftarrow{D_{i-1}} & \text{inače} \end{cases}$$

$$K_i = PC2(C_iD_i)$$

Gdje je $PC2$ fiksna permutacija opisana slikom 2.10, a operator ${}^{j}\overleftarrow{A}$ predstavlja ciklički pomak ulijevo za j mjesta nad riječi A sasatavljenoj od bitova.

PC1						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Slika 2.9: Fiksna permutacija $PC1$

PC2					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Slika 2.10: Fiksna permutacija $PC2$

Nakon tog cijelog postupka primjenimo *inverznu permutaciju* IP^{-1} na $R_{16}L_{16}$, a onda je šifrat y dan sa:

$$y = IP^{-1}(R_{16}L_{16})$$

Primjetimo da smo sa inverznom permutacijom dijelovali na riječ $R_{16}L_{16}$,a ne L_{16} i R_{16} (obrnuti poredak). Permutacija IP^{-1} je prikazana slikom 2.11. Postupak za dešifriranje je

analogan:

Krećemo od šifrata y: na njega najprije primjenimo permutaciju IP sa slike 2.5, $y_0 = IP(y) = IP(IP^{-1}(R_{16}L_{16})) = R_{16}L16$. A sada imamo:

$$R_{15} = L_{16}$$

A iz

$$R_{16} = L_{15} \oplus f(R_{15}, K_{16})$$

Slijedi

$$L_{15} = R_{16} \oplus f(R_{15}, K_{16})$$

Općenito, za sve $i = 1, ..., 16$ dobivamo da vrijedi:

$$R_{16-i} = L_{16-i+1}$$

$$L_{16-i} = R_{16-i+1} \oplus f(R_{16-i}, K_{16-i+1})$$

Iz čega dobivamo niz $R_{14}L_{14}, R_{13}L_{13}, ...R_0L_0$. A zatim zamjenom poretka od R_0L_0 i primjenom na to IP^{-1} dobivamo: $IP^{-1}(L_0R_0) = IP^{-1}(IP(x)) = x$ Pa smo dobili traženi otvoreni tekst.

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

Slika 2.11: Inverzna permutacija IP^{-1}

Provaljivanje DES-a DNA izračunavanjem

2.5 Enkripcija i dekripcija podataka

שורה	מס' עמודה															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	S_1															
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	3	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	13	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
	S_2															
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
	S_3															
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
	S_4															
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
	S_5															
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
	S_6															
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
	S_7															
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
	S_8															
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Slika 2.12: S-kutije

Bibliografija

[1] L. M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266(11):1021–1024, Nov. 1994.

[2] M. Borda. *Fundamentals in Information Theory and Coding*. Springer, 2011.

[3] A. Dujella and M. Maretić. *Kriptografija*. Element, 1st edition, 2007.

[4] J. Hromkovic and W. M. Oliva. *Algorithmics for Hard Problems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2nd edition, 2002.

[5] R. J. Lipton. DNA solution of hard computational problems. *Science*, 268(5210):542–545, Apr. 1995.

[6] M. Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 2nd edition, 2006.

[7] M. Vuković. *Matematička logika*. Element, 1st edition, 2009.

[8] S. Yan. *Computational Number Theory and Modern Cryptography*. Wiley-HEP information security series. Wiley, 2012.