DNA kriptografija

Kovačić, Antonio

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 nesigurnog 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, osnovnim biološkim konceptima DNA, probijanje DES kriptosustava pomoću DNA računala te algoritmom za enkripciju, odnosno dekripciju podataka pomoću DNA.

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:*

$$d(\varepsilon) := 0$$
$$d(wa) := d(w) + 1, \quad a \in A$$

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) → 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) → uz pretpostavku da $K=\emptyset$:*

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

- *Uoči(K) → 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) → 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* $f : S \subseteq \mathbb{N}^k \rightarrow \mathbb{N}$ *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* $MulS(K) = \{f(\vec{x})\}$.
Kažemo da je funkcija $f : S \subseteq \mathbb{N}^k \rightarrow \mathbb{N}$ ***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.*
Uočimo 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 [12].

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

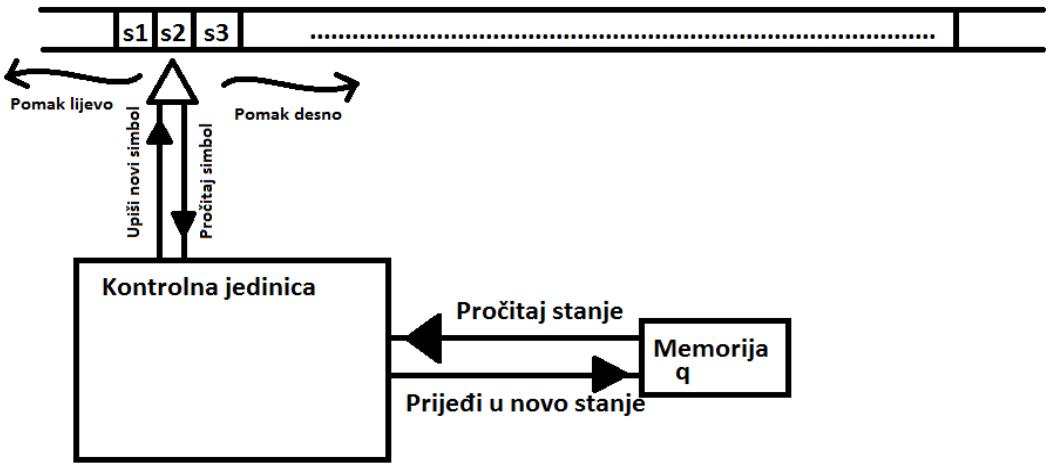
- *Q konačan skup čije elemente nazivamo stanja*
- *Σ je konačan skup, čije elemente nazivamo ulazni simboli, pretpostavljamo da Σ ne sadrži "prazan simbol" kojeg označavamo sa ε*
- *Γ je konačan skup kojeg nazivamo alfabet Turingovog stroja, pretpostavljamo da je $\varepsilon \in \Gamma$, te $\Sigma \subset \Gamma$*
- *$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, D, S\}$ 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 [13, str. 12-25].
Više o Turingovom stroju te nekim pojmovima na koje se pozivamo u idućim rezultatima se mogu naći u:

- Turing prepoznatiljivost, Turing odlučivost [12, str. 141-142]
- Vremenska složenost deterministčkog Turingovog stroja se može naći u [12, str. 248], a nedeterminističkog u [12, str. 255]
- Klase vremenske složenosti:
  - $TIME(f(n))$ u [12, str. 251]
  - P u [12, str. 258]
  - Vezano uz klasu NP u [12, str. 265-267]
- Polinomna reducibilnost u [12, str. 272]
- NP-potpunost u [12, str. 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 [12, str. 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 [10].
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 potpoglavlju 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 [12, str. 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 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, konstruirani na ovaj način, daju mogućnost povezivanja dva vrha koja su povezana nekim birdom (u smislu biokemije, bridovi igraju ulogu komplementarnog lanca koji spajanjem s neka druga dva lanca daje strukturu dvostruke uzvojnice objašnjene u poglavlju 2.1).
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 vrhom v_{in}, odnosno ne završavaju s vrhom 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.

□

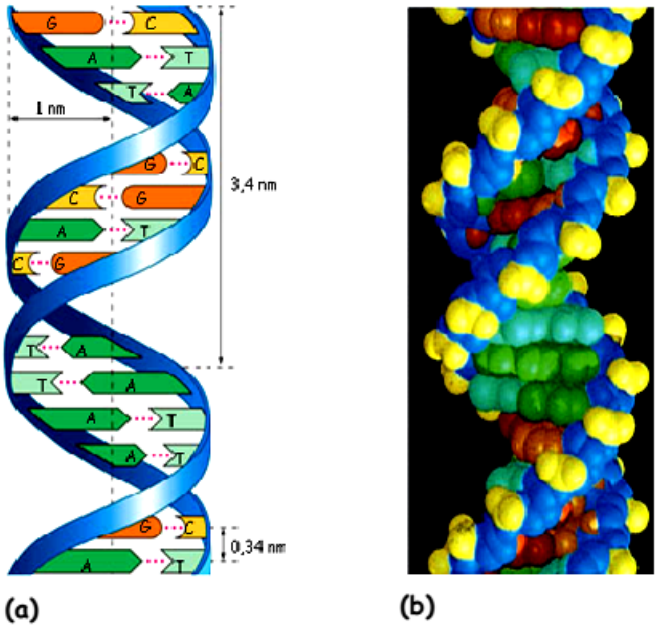
Poglavlje 2

DNA kriptografija

2.1 Osnovni biološki koncepti deoksiribonukleinske kiseline

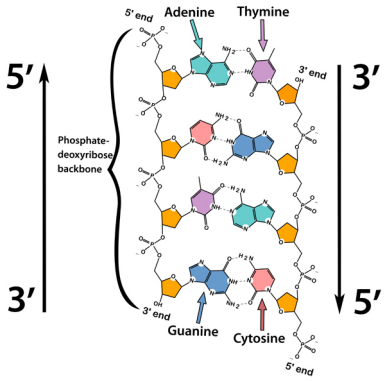
Deoksiribonukleinska kiselina (DNA) sastavljena je od dva duga lanca nukleotida (polinukleotidni lanci) koji su omotani jedan oko drugoga, odnosno ima strukturu *dvostruke uzvojnice* (engleski **double helix**) (slika 2.1). **Nukleotid** je osnovna građevna jedinica DNA, a izgrađena je od jedne od četiri 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 drugi završavati s 3' ugljikovim atomom kao krajem (slika 2.3). Pri povezivanju pripadnih dušićnih baza uvijek se adenin povezuje s timinom, a gvanin s citozinom. Sada kada smo objasnili 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 (single 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*. Ostale kemijske operacije koje ćemo upotrebljavati ćemo opisati u poglavlju 2.4.

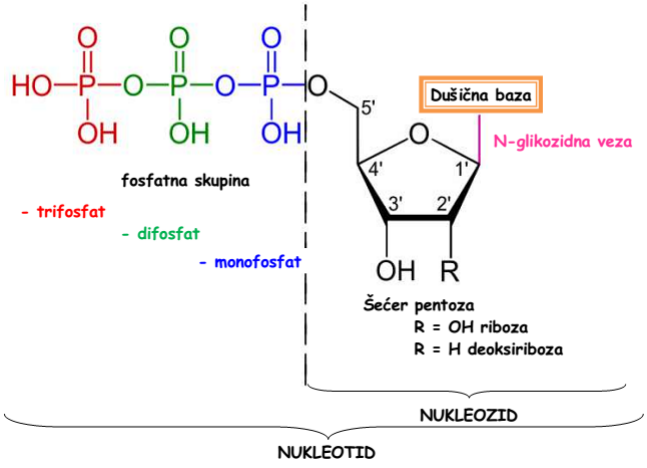


Slika 2.1: Struktura DNA: a) 2D model, b) 3D model

Slika 2.2: Hibridizacija



Slika 2.3: Antiparalelnost DNA



Slika 2.4: Nukleotid

2.2 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 biti 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 2.2.1. *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 2.2.2. *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 se 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 razlikujemo *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). Prirodno pitanje koje se postavlja jest: postoji li savršeno sigurni kriptosustav. O pojmu savšrene sigurnosti govorio je kriptograf Claude Shannon, prema njegovoj definiciji je kriptosustav savršeno siguran ukoliko šifrat ne daje nikakvu informaciju o otvorenom tekstu, preciznije: neka je x fiksirani otvoreni tekst iz prostora otvorenih tekstova koji se pojavljuje s vjerojatnošću $\mathbb{P}(x)$, tada je uvjetna vjerojatnost $\mathbb{P}(x|y)$ da je x otvoreni tekst, ako znamo da je y šifrat, jednaka $\mathbb{P}(x)$, odnosno:

$$\mathbb{P}(x|y) = \mathbb{P}(x), \quad \forall y \in C$$

gdje je C prostor svih šifrata.

Kada se govori o savršeno sigurnom kriptografskom sustavu, često nailazimo na pojam jednostruke bilježnice (*eng.* ***o****ne-****t****ime* ***p****ad*). Kriptosustavi koji koriste koncept jednostruke bilježnice zapravo koriste kriptografske ključeve koji vrlo često budu izabrani na slučajan način te se upotrebljavaju kako bi se šifrirao točno jedan otvoreni tekst, a nakon toga se više ne koriste. Takav koncept se upravo koristi u DNA kriptografiji, čiju ćemo primjenu vidjeti u poglavlju 2.5. Više o kriptografiji se može naći u [7].

2.3 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.

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

[1] [7, str. 56-57]

Zatim napišemo x_0 u obliku konkatenacije 2 *riječi bitova* koje su duljine 32 bita svaka: $x_0 = L_0R_0$. Nakon toga računamo L_i i R_i, gdje je:

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

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

Ovdje su $K_1, ..., K_{16}$ *riječi bitova* duljine 48, dobivene 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 djelovanje 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 $B = E(A) \oplus J$ 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

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

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 $l{\cdot}8, l = 1, ..., 8$ 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 sastavljenoj 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 s inverznom permutacijom djelovali 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. Primjer šifriranja otvorenog teksta pomoću DES kriptosustava može se naći na [8]. O implementaciji DES-a na DNA računalu se može naći u [2], u ovom radu, fokusirat ćemo se više na provaljivanje DES kriptosustava opisano u idućem poglavlju.

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}

i	S_i															
1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Slika 2.12: S-kutije

2.4 Provaljivanje DES-a DNA izračunavanjem

Uvod i oznake

U poglavlju 1.1 smo naveli opisnu definiciju DNA računala koju je predložio Leonard Adleman, a u poglavlju 2.1 su navedene osnovne činjenice i neke od biokemijskih opera-

cija koje se koriste u genetičkom inženjeringu. U ovom poglavlju ćemo detaljno opisati one operacije koje ćemo koristiti za rješavanje napada na DES kriptosustav, ali ih nećemo striktno definirati u definiciji navedenoj u poglavlju 1.1. Naime, ta definicija je služila kao osnovni matematički koncept kojeg je predložio Adleman, a koji se kasnije postupno proširivao. Kako su se proširivala istraživanja o DNA i genetičkom inženjeringu, tako bi se i dana definicija mogla izmjenjivati. Opisat ćemo oznake i biokemijske operacije koje ćemo koristiti. Neka je $\mathtt{x}$ proizvoljna riječ alfabeta $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. Biološki gledano, $\mathtt{x}$ nije još DNA lanac jer nema orijentaciju. Nadalje, uvodimo Watson-Crickov komplement (biološki komplement u poglavlju 1.1). To je preslikavanje opisano sa $(\mathtt{A} \to \mathtt{T}, \mathtt{T} \to \mathtt{A}, \mathtt{G} \to \mathtt{C}, \mathtt{C} \to \mathtt{W})$. Watson-Crickov komplement označavat ćemo sa $\overline{\mathtt{x}}$. Sa $\mathtt{x}^{\mathrm{R}}$ ćemo označavati riječ koja je sastavljena od istih simbola kao i riječ $\mathtt{x}$ samo u obrnutom poretku. Sa $\uparrow \mathtt{x}$ ćemo označavati jednolančani DNA čija je orijentacija $5' \to 3'$, a sastavljena je od niza simbola od kojih se sastoji $\mathtt{x}$. Sa $\downarrow \mathtt{x}$ ćemo označavati lanac orijentacije $3' \to 5'$ sa nizom simbola koji čine Watson-Crickov komplement niza $\mathtt{x}$. U konačnici, sa $\updownarrow \mathtt{x}$ označavamo dvolančanu strukturu nastalu spajanjem $\uparrow \mathtt{x}$ i $\downarrow \mathtt{x}$. Kako bi shvatili sve oznake napravit ćemo jedan primjer:

Primjer 2.4.1. *Neka je* $\mathtt{x} = \mathtt{TGCCGCAG}$ *riječ alfabeta* $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$*, tada je:*

$$\overline{\mathtt{x}} = \mathtt{ACGGCGTC}$$

$$\mathtt{x}^{\mathrm{R}} = \mathtt{GACGCCGT}$$

$$\uparrow \mathtt{x} = 5' - \mathtt{TGCCGCAG} - 3'$$

$$\downarrow \mathtt{x} = 3' - \mathtt{ACGGCGTC} - 5' = \uparrow \overline{\mathtt{x}}^{\mathrm{R}}$$

$$\updownarrow \mathtt{x} = \begin{matrix} 5' - \mathtt{TGCCGCAG} - 3' \\ 3' - \mathtt{ACGGCGTC} - 5' \end{matrix}$$

Reprezentacija riječi sastavljenih od alfabeta $\{0, 1\}$

Neka je $x = x_1x_2...x_n$ proizvoljna riječ alfabeta $\{0, 1\}$ duljine n. Ideja je jednom bitu i pripadnoj poziciji bita i pridružiti jedinstveni 30-mer – oligonukleotida duljine 30 (općenito n-mer bi bila oligonukleotida sastavljena od n nukleotida).

1. $B_i(0)$ označava oligonukleotidu koja reprezentira da je i-ti bit riječi x jednak 0
2. $B_i(1)$ označava oligonukleotidu koja reprezentira da je i-ti bit riječi x jednak 1
3. Za $i \in \{0, 1..., n\}$ neka je S_i 30-mer koji označava separator između susjednih bitova

U konačnici, DNA lanac koji reprezentira binarnu riječ x je:

$$\updownarrow S_0B_1(x_1)S_1B_2(x_2)S_2...S_{n-1}B_n(x_n)S_n$$

Svi B_i-ovi i S_i-ovi su međusobno različiti. Štoviše, kasnije ćemo vidjeti da će biti nužno da svaka dva bita nemaju zajedničke podriječi čija je duljina veća ili jednaka 15. To se može postići tako da riječi $B_i(b)$ i S_i budu izabrane na slučajan način, budući da zajednička podriječ dvije slučajno izabrane riječi ima tendenciju biti što kraća. Upravo smo iz toga razloga uzeli 30-mere kao riječi koje će predstavljati bitove. S druge strane, biokemijskim reakcijama u laboratoriju će trebati duže da se odviju nad 30-merima nego nad nekim kraćim oligonukleotidama. Adleman je tako za svoj eksperiment opisan u [1] koristio 20-mere, no njegov problem je imao manje podataka koje je morao reprezentirati. Uočimo da će svaki bit biti reprezenitran s dvije riječi iz skupa $\mathbb{Y} = \{(y_1, ..., y_{15}) : y_i \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}\}$, pa je vjerojatnost pojavljivanja podriječi duljine 15 unutar dvije različite riječi, odnosno unutar iste riječi na različitim pozicijama jednaka $p = \left(\frac{1}{4^{15}}\right)^2 = \frac{1}{1152921504606846976} \approx 8.6736 \cdot 10^{-19} \approx 0$. Riječi S_0 i S_n će u replikaciji s PCR-om služiti kao početnice (operacija opisana u

idućem potpoglavlju). Uvodimo još oznaku $R_i(x)$ koja će označavati sljedeću riječ:

$$R_i(x) = B_i(x_1)S_iB_{i+1}(x_2)S_{i+1}...S_{i+n-2}B_{i+n-1}(x_n)S_{i+n-1}$$

Opće biokemijske operacije

Hibridizacija (taljenje i hlađenje) Postoje dvije biokemijske reakcije kojima se postiže spajanje dva komplementarna lanca (rezultat im je jednak). Jedna se zove hibridizacija, a druga je metoda taljenja i hlađenja. Osnovni rezultat tih operacija je prikazan donjom jednadžbom:

$$\uparrow \mathtt{x+} \downarrow \mathtt{x} = \updownarrow \mathtt{x}$$

Ona opisuje operaciju spajanja dva jednolančana DNA lanca u jedan dvolančani (ima strukturu dvostruke uzvojnice). Pod hibridizacijom (ili postupkom taljenja i hlađenja) je moguća i sljedeća reakcija:

$$\uparrow \mathtt{xy+} \downarrow \mathtt{yz} = \uparrow \mathtt{x} \updownarrow \mathtt{y} \downarrow \mathtt{z}$$

.

Ekstrakcija ili izdvajanje Služi za izdvajanje onih DNA lanaca koji kao podriječ sadrže riječ `x`. To se biokemijski ostvaruje na sljedeći način:
U epruvetu se dodaje dovoljan broj kopija od $\downarrow \mathtt{x}$ koje se nalaze na određenoj površini premazanoj gelom koji u sebi sadrži biotin-streptavidin (na primjer na staklu ili Petrijevoj zdjelici) te djeluju poput magneta. Zatim se postupkom hibridizacije događa na primjer:

$$\uparrow \mathtt{vxw+} \downarrow \mathtt{x} = \uparrow \mathtt{v} \updownarrow \mathtt{x} \uparrow \mathtt{w} \tag{2.1}$$

Nakon toga se na tom gelu nalaze sve riječi koje poprimaju oblik sličan onome s desne strane jednakosti u jednadžbi 2.1 (no nalaze se i oligonukleotide $\downarrow \mathtt{x}$ jer neće sve biti is-

korištene – u gel se stavlja veći broj kopija od potrebnog). Naime osim spajanja koje je prikazano u 2.1 moguća su i razna spajanja različitoga oblika (na primjer gdje oligonukleotida $\uparrow \mathtt{x}$ može biti vezana na početku nekog DNA lanca koji poprima dvolančanu strukturu). U slučaju kad će to biti potrebno istaknuti, na ovu operaciju ćemo se referencirati kao na ekstrakciju koja koristi niz $\downarrow \mathtt{x}$. Kasnije se $\uparrow \mathtt{v} \updownarrow \mathtt{x} \uparrow \mathtt{w}$ izdvaja pomoću tehnike koja se zove gel elektroforeza, a nakon toga se $\downarrow \mathtt{x}$ odvaja od originalnih lanaca pomoću enzima DNA helikaze. U našoj primjeni, mi ćemo tu operaciju označavati sa:

$$Extract(T, x_i = 1)$$

gdje je T proizvoljna epruveta koja sadrži DNA lance. Ovom operacijom izdvajamo sve one DNA lance koji kao podriječ sadrže $B_i(1)$ (na i-tom mjestu binarne riječi nalazi se 1). Naravno osim jednostrukog bita, možemo izdvojiti cijele podriječi, na primjer:

$$Extract(T, x_i x_{i+1} = 01)$$

će izdvojiti sve one binarne riječi koje na i-tom i $i+1$-om mjestu imaju redom 0 i 1, a za to će se koristiti DNA lanac $R_i(01) = B_i(0)S_iB_{i+1}(1)S_{i+1}$. Uočimo da je to zapravo operacija:

$$Extract(T, x_i = 0 \wedge x_{i+1} = 1)$$

pa smo napravili "i" operaciju – ipak to je moguće napraviti samo kad se radi o susjednim bitovima – inače, moramo napraviti dvije uzastopne operacije izdvajanja. Na gel kojim premazujemo staklenu površniu osim premaza streptavidin-biodina koji sadrži na primjer sekvencu $R_i(101)$, možemo na istu staklenu površinu i koristiti premaz streptavidin-biodina

koji sadrži niz $R_i(001)$. Time će se izvršiti operacija:

$$Extract(T, x_i x_{i+1} x_{i+2} = 101 \vee x_i x_{i+1} x_{i+2} = 001)$$

Iz tehničkih, biokemijskih, razloga moguće je samo izvoditi ovakvu simultanu "ili" operaciju kada su binarne riječi koje su argumenti od R_i jednake duljine.

Replikacija Za razliku od običnog proširivanja opisanog u poglavlju 1.1 ovdje ćemo koristiti biokemijsku operaciju replikacije pomoću polimerazne lančane reakcije (PCR - **p**olymerase **c**hain **r**eaction). Da bi PCR mogao djelovati, moraju se zadovoljiti sljedeći uvjeti:

- svaki lanac mora biti oblika $\updownarrow$ `aBc`, gdje su `a` i `c` fiksne riječi, a `B` proizvoljna
- `a` i `c` se ne smiju pojaviti kao podriječi od `B`

Ako prvi uvjet nije ispunjen, to znači da u epruveti nema takvih riječi. Drugi uvjet će se osigurati duljinom DNA oligonukleotida tako da će biti moguće kreirati točno takve lance – `a` i `b` će u rješavanju našega problema biti takve duljine da će njihova vjerojatnost pojavljivanja u `B` biti jednaka 0, a to se omogućuje i reprezentacijom bitova opisanoj u 2.4. Podriječi `a` i `b` zovemo početnice (*eng. primers*). Ova operacija omogućuje sintetičku proizvodnju kopije proizvoljne riječi B (uočimo da je efekt jako sličan onom efektu operacije proširivanja opisane u poglavlju 1.1).

Označavanje Na određeni DNA lanac, dodajemo novu oligonukleotidu `z`, a postupak je opisan sljedećim koracima:
Uvjet je da svaki lanac ima oblik $\uparrow$ `Xy`, gdje je `X` proizvoljan, a `y` fiksiran:

$$\uparrow \texttt{Xy} + \downarrow \texttt{yz} = \uparrow \texttt{X} \updownarrow \texttt{y} \downarrow \texttt{z} \tag{2.2}$$

Zatim se dodaje enzim DNA polimeraza koji će konvertirati parcijalni dvolančani lanac s desne strane jednakosti jednadžbe 2.2 u potpuni. Najprije se "čitaju" jednolančane porcije s desne strane jednakosti od 2.2 i kreiraju njihovi Watson-Crickovi komplementi, za svaku nukleotidu u jednom trenutku. Odnosno događaju se pojedinačno sljedeće reakcije:

$$\uparrow \texttt{X} \updownarrow \texttt{y} \downarrow \texttt{z+} \downarrow \texttt{X} = \updownarrow \texttt{Xy} \downarrow \texttt{z} \tag{2.3}$$

$$\updownarrow \texttt{Xy} \downarrow \texttt{z+} \uparrow \texttt{z} = \updownarrow \texttt{Xyz} \tag{2.4}$$

Zatim se opet pomoću DNA helikaze dobiva $\uparrow \texttt{Xyz}$.

Plan napada na DES kriptosustav i algoritam

Označimo s $DES(M,k)$ šifrat dobiven enkripcijom otvorenog teksta M koristeći ključ k u DES kriptosustavu. U poglavlju 2.2 smo napomenuli da napadač na dani kriptosustav obično ima odabrani otvoreni tekst i njemu pripadajući šifrat. U tu svrhu pretpostavit ćemo da je parametar M_0 neki fiksan 64-bitni otvoreni tekst, a k ključ kojeg pokušavamo naći. Poznato je da otvorenom tekstu M_0 pripada šifrat E_0. Tada za dani M_0 uvodimo funkciju $f : \{0,1\}^{56} \to \{0,1\}^{64}$ koja za neki ključ k proizvoljan, tekstu M_0 pridružuje šifrat $DES(M_0,k)$, odnosno $f(K) = DES(M_0,k)$. Naša je zadaća naći k_0 takav da je $f(k_0) = E_0$, odnosno naći inverz funkcije f ako su nam poznati otvoreni tekst i njegov pripadni šifrat. Ono što ćemo napraviti jest konsturiranje svih parova $[k, f(k)]$ za svih 2^{56} mogućih ključeva. Pošto su permutacije u DES kriptosustavu fiksirane, a neke od njih samo skraćuju ili nadodaju još bitova na točno određeni način, za naše primjene te permutacijske kutije ne igraju značajnu ulogu i mogu biti ignorirane. Naime, nema potrebe da fizički permutiramo bitove u DNA lancima, već samo možemo pratiti (na primjer na papiru) bitove DNA lanca koji kodiraju bitove u izračunavanju. Za računanje operacije $\oplus$ i S-kutija radimo sljedeće: Pretpostavimo da želimo izračunati ekskluzivno ili (XOR, $\oplus$) i-tog i j-tog bita u epruveti T. To ćemo učitniti

dodajući vrijednost $x_i \oplus x_j$ na DNA lanac koji reprezentira riječ x. Prvi korak u tome je razdvajanje epruvete T na dva rješenja T_0, T_1 gdje vrijedi: $T_0 = \{x \in T | x_i \oplus x_j = 0\}$, a $T_0 = \{x \in T | x_i \oplus x_j = 1\}$. Definiramo međurješenje na idući način:

$$T^{11} = Extract(T, x_i = 1 \vee x_j = 1)$$

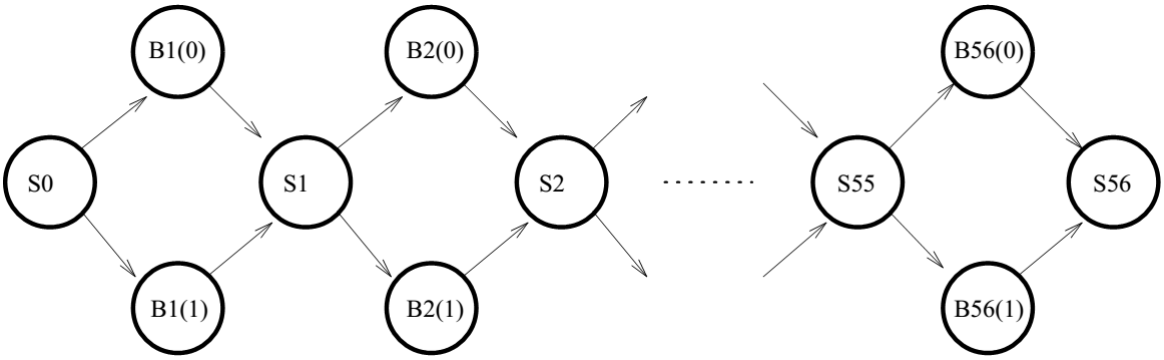
$$T^{1} = Extract(T^{11}, x_i = 0 \vee x_j = 0)$$

$$T^{0} = T \backslash T^{1}$$

Jedna S-kutija zaravo odgovara preslikavanju: $g : \{0,1\}^6 \rightarrow \{0,1\}^4$. Pretpostavimo sada da bitovi koje želimo preslikati (na koje djeluje S-kutija) počinju na poziciji i, a završavaju na poziciji $i + 5$. Svakom DNA lancu x želimo prilijepiti 4-bitnu riječ $g(x)$. To se radi tako da se rješenje T razdvaja na 16 različitih riješenja $T_0, ..., T_{15}$ ovisno prema vrijednosti koja se dodjeljuje. Tada ćemo svakom pojedinačnom rješenju T_j nadodati 4-bitnu riječ j. Na primjer, pretpostavimo da funkcija g preslikava 6-bitne riječi a i b u 0, i ne postoji riječ $c \in \{(x_1, ..., x_6) | x_i \in \{0,1\}\} \backslash \{a, b\}$ takva da je $f(c) = 0$. Tada je:

$$T_0 \leftarrow Extract(T, x_i...x_{i+5} = a \vee x_i...x_{i+5} = b)$$

Preostale operacije ekstrakcije se rade na sličan način. Ovo pokazuje da se pojedinačna S-kutija može izračunati u 16 operacija ekstrahiranja i jednako toliko operacija označavanja. Operacija izdvajanja zahtjeva pripremu od $2^6 = 64$ različitih premaza biotin-streptavidinom. Ti premazi se pak mogu pripremiti na sličan način kojim će biti kreirana *inicijalna otopina* za rješavanje problema, a oni će se kasnije koristiti u svih 16 rundi izračunavanja pa ih je dovoljno pripremiti jednom na početku cijelog procesa. Sada kada smo opisali uvodni dio, krećemo na detaljan opis algoritma za probijanje DES kriptosustava. U tu svrhu pogledajmo donji graf na slici 2.13.



Slika 2.13: Inicijalni graf

Svaki put u tom grafu od čvora S_0 do čvora S_{56} predstavlja jedan mogući ključ u DES kriptosustavu. Ta percepcija nam treba kako bi jasno shvatili korake algoritma. Važno je naglasiti da ako znamo otvoreni tekst M_0 preslikavanje funkcije f koja preslikava ključ u šifrat se zapravo može spremiti u tablicu, a zahvaljujući izuzetnim kapacitetima DNA što se tiče memorije (na primjer 5.5 petabita podataka - što je oko 700 terabajta - je spremljeno u jedan gram DNA)[2], $f(k)$ možemo spremiti bez većih problema na DNA, a njen izračun se računa samo na temelju bitova iz k (jer je prva komponenta u $DES(M_0, k)$ konstanta).

1. Istaknuli smo da za svaki vrh V postoje 15-meri V^l i V^r takvi da je $V = V^l V^r$ i da ne postoji vrh $U \neq V$ takav da je $U = V^l V^r$ ili $U = V^r V^l$.

2. Ulaz: Za svaki vrh v kreiraj epruvetu koja sadrži jednolančane DNA $\uparrow V^l V^r$. Za svaki usmjereni vrh (u, v) kreiraj epruvetu sastavljenu od jednolančanih DNA $U^r V^l$. Kako bi kreirali potpune dvolančane strukture DNA, kreiraj dvije epruvete od kojih će jedna sadržavati $\uparrow S_0^l$, a druga $\downarrow S_{56}^r$. (Uočimo da je taj korak algoritma analogan drugom koraku rješavanja problema Hamiltonovog puta)

3. Pomiješaj sve epruvete u jednu te započni postupak hibridizacije

[2][6]

4. Ekstrahiraj sve DNA lance koji sadrže riječ S_0

5. Ekstrahiraj sve DNA lance koji sadrže riječ S_{56}
   Time smo dobili sljedeće:

   $$\updownarrow S_0 R_1(k)$$

   gdje je $k \in \{0,1\}^{56}$

6. Izračunaj prijelaze DES sklopovlja za sve ključeve, a pojedini ključ označi rezultatom DES prijelaza. Na primjer, ako prvi prijelaz DES sklopovlja, s obzirom na fiksnu poruku M_0 nalaže da se napravi *XOR* na 2 i 5 bitu ključa k, tada će ključ nakon izračuna $k_2 \oplus k_5$ biti označen sa $B_{57}(k_2 \oplus k_5)S_{57}$, odnosno dobit ćemo:

   $$\updownarrow S_0 R_1(k) B_{57}(k_2 \oplus k_5) S_{57}$$

7. Računamo jedan po jedan prijelaz u DES sklopovlju, a za svaki prijelaz, njegov izlaz dodajmo na kraj svakog DNA lanca. Nakon toga svi lanci u rješenju izgledaju kao:

   $$\updownarrow S_0 R_1(k) R_{57}(I) R_r(DES(M_0, k))$$

   gdje je I riječ bitova koja odgovara izlaznim vrijednostima pojedinih unutarnjih prijelaza u DES sklopovlju.

Ovaj algoritam, kada bi se ukupno raspisale sve operacije, koristi 916 koraka i to da se neki koraci izdvajanja izvedu paralelno, a o detaljnoj analizi algoritma, kao i o daljnjoj optimizaciji u smislu skraćivanja DNA lanaca, se može naći u [3].

2.5 Enkripcija i dekripcija podataka

U uvodu smo napomenuli da je DNA kriptografija kvantno rezistentna. U nastavku ćemo opisati jedan algoritam koji koristi koncept DNA. Na običnom računalu DNA lanac zapravo i nema neku bitnu ulogu (osim što ima Watson-Crickov komplement, ali koji bi se mogao slično definirati za bilo koja druga četiri para slova, DNA je služio samo kao motivacija), ali DNA računalo upravo zbog biokemijskih reakcija olakšava postupak dešifriranja. Pretpostavljamo da Alice želi razmjeniti poruku s Bobom preko nesigurnog komunikacijskog kanala, nakon što su preko sigurnog komunikacijskog kanala razmjenili ključeve. Neka je M otvoreni tekst koji Alice želi kriptirati i poslati preko nesigurnog komunikacijskog kanala. Alice će najprije pretvoriti M u pripadni $ASCII$ kod, a potom će taj $ASCII$ kod prevesti u binarni otvoreni tekst M'. Neka je $m' = d(M')$ duljina binarnog otvorenog teksta M'. Označimo s k_A proizvoljan, slučajno izabrani, jednolančani DNA koji će nam poslužiti kao ključ, a koji je duljine $10m'$. Alice kriptira po sljedećem algoritmu:

1. M pretvorimo u ASCII M_A, a onda M_A u binarni prikaz M' otvorenog teksta M
2. Alice izabire slučajan $d(M') * 10$-mer (ssDNA) i koristi ga kao ključ k_A
3. Čitaj M' s lijeva na desno, a čitaj $\overline{k_A}^R$ s desna na lijevo u blokovima od 10 dušićnih baza (blokove čitaj s lijeva na desno)

   - m=0
   - Dok ne pročitaš sve bitove:
     - Ako je bit jednak 1
       uzmi dani blok i nadodaj na njega m;
       Pošalji blok Bobu
       m:=m+1;

- Inače (ako je bit jednak 0)
  ignoriraj blok i prijeđi na idući

Kada Bob primi sve blokove, on dekriptira šifrat po idućem postupku:

1. Pročitaj broj zalijepljen na kraj dušićnih baza (označimo ga s i) i obriši ga iz bloka
2. kreiraj Watson-Krickov komplement danog bloka i nađi mjesto gdje se nalazi krećući od indeksa $(d(M') - i - 1) \cdot 10$ čitajući blok po blok s desna na lijevo
3. Svaki blok kojeg Bob nađe zamijenjuje s 1, a one blokove iz ključa koje Alice nije poslala označi s 0
4. Dani binarni string Bob preokreće i nalazi mu ASCII vrijednost, a za ASCII vrijednost mu nađe ekvivalentan otvoreni tekst u prirodnom jeziku

Slijedi implementacija u programskom jeziku Pythonu.

```
import string
import math
import binascii
import random

def WatsonCrick(DNArijec):
        DNArijec2=""
        for i in range(len(DNArijec)):
                if (DNArijec[i]=='A'): DNArijec2+='T'
                if (DNArijec[i]=='T'): DNArijec2+='A'
                if (DNArijec[i]=='G'): DNArijec2+='C'
                if (DNArijec[i]=='C'): DNArijec2+='G'
        return DNArijec2
```

```
rijec="DNA"
sifrat=[]
lista=['A','C','G','T']
rijec=list(rijec)
rijecBin=""
m=0
for i in range(len(rijec)):
        rijec[i]=bin(ord(rijec[i]))
        rijec[i]=rijec[i][2:len(rijec[i])]
        if (len(rijec[i])<7) :
                rijec[i]=''.join('0' for i in range(7-len(
                   rijec[i])))+rijec[i]
        m+=len(rijec[i])
        rijecBin+=rijec[i]

kljuc=''.join(random.choice(lista) for i in range(len(
   rijecBin)*10))
print "Otvoreni tekst: " + rijecBin
print "Kljuc: "
for i in range (len(kljuc)/10):
        print kljuc[i*10:(i+1)*10]
k=0
for i in range (len(rijecBin)):
        rijecS=""
        if(rijecBin[i]=="1"):
```

```
                rijecS+=WatsonCrick(kljuc[(m-i-1) * 10 : (m-
                   i-1) * 10 + 10])
                rijecS+=str(k)
                k=k+1
                sifrat.append(rijecS)
print "Blokovi sifrata:"
print sifrat
"""Dekripcija podataka"""
invKljuc=""
myInd=0
bini=""
for i in range(len(kljuc)/10):
        invKljuc+=kljuc[(m-i-1)*10:(m-i-1)*10+10]
for i in range(len(sifrat)):
        p=int(sifrat[i][10:len(sifrat[i])])
        sifrat[i]=sifrat[i][0:10]
        if (i==0):myIndPrev=0;
        else: myIndPrev=myInd;
        myInd=invKljuc.find(WatsonCrick(sifrat[i]), p*10)/10
        bini=bini+''.join('0' for i in range(myInd-myIndPrev
            -1))+'1'
print "Dekriptirani sifrat: " + bini
print "Provjera: otvoreni tekst==sifrat: " + str(bini==
   rijecBin)
```

Primjer 2.5.1. *Alice želi kriptirati tekst M = "DNA". Njegova ASCII vrijednost je* 44 4*E* 41 *Što se nakon pretvaranja u binarnu riječ reprezentira kao:* 1000100 1001110 1000001.

Duljina otvorenog teksta: $d(M') = 21$. *Alice na slučajan način izabire ključ duljine* 210 *(za svaki bit po jedan 10-mer) sastavljen od simbola iz skupa* $\{A, C, G, T\}$.

k_A = TGTCAACTTCCATCTGATGA CCTAGTCAGTTCGCGTGTGA CAGCCGGTGAAAAGATCAGA TCTGACTAATGCCGAACCCG CTTTTGTTGGTTAGTCGCGA GCGCTAAGGAGGGACATACC GGGCTCAACACCAGTTCGAG TTGACTGAAGCTAGGGCGCT ACTATCTATCCAAGTGACTA CATGTCTAGGCCATTCCGAA GTCGTTTAAC

Nakon toga Alice šifrira: prvi bit koji pročita je 1 pa uzima blok GTCGTTTAAC*, napravi Watson-Crickov komplement tog bloka, nadoda* 0 *i dobije:* CAGCAAATTG0*, iduća tri bloka ignorira (jer su idući bitovi koje je pročitala bili* 0*), a zatim iz bloka* ACTATCTATC *dobije šifrat:* TGATAGATAG1*. Potpuni šifrat dan je sa idućom listom blokova:*

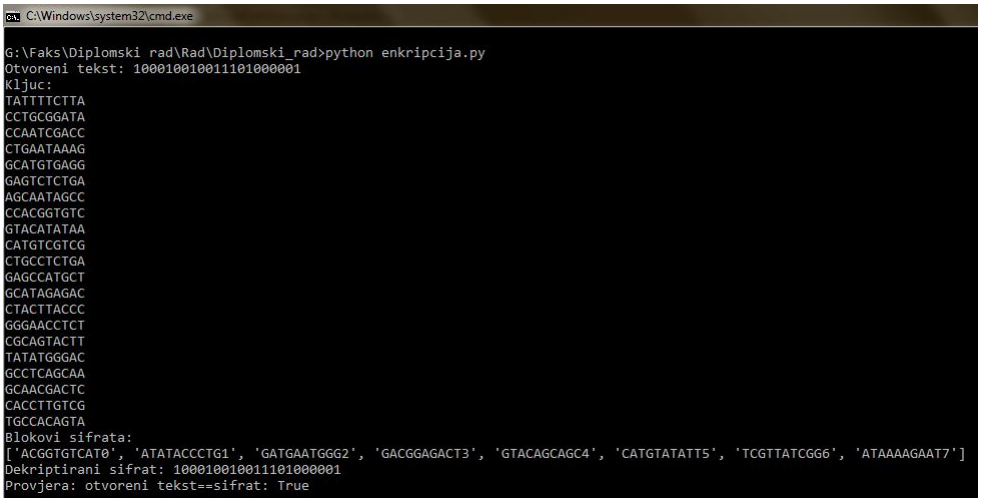
['CAGCAAATTG0','TGATAGATAG1','GGTCAAGCTC2','CGCGATTCCT3',

'AATCAGCGCT4','GAAAACAACC5','AGACTGATTA6','ACAGTTGAAG7']

Bob je primio blok TGATAGATAG1 *i dešifrira ga tako da prvi blok ključa ignorira (1 je zaljepljen na kraj šifrata), uzima Watson-Crickov komplement bloka* ACTATCTATC*, te traži idući blok (čitajući s desna na lijevo) koji je njemu jednak. Pretpostavimo da je Bob već dekriptirao prije* 0*-ti blok. Sada slaže bitovnu riječ tako da svaki blok koji je ignorirao zamjeni s jednom* 0*, a koji nije ignorirao zamjeni s* 1*: pa dobiva* 10001*, gdje je s crvenom bojom označen bit dobiven nakon prve runde dekriptiranja, a plavom bojom je označena riječ dobivena drugom rundom dekriptiranja. Na taj način Bob dobiva binarni prikaz otvorenog teksta, kojeg lagano pretvara u riječ prirodnog jezika.*

Na slici 2.14 se može vidjeti primjena enkripcije i dekripcije pomoću navedene implementacije u Pythonu.

Implementacija pomoću DNA računala je zapravo jako jednostavna. Za dani otvoreni binarni tekst, Alice samo treba kreirati određene Watson-Crickove komplemente slučajno



Slika 2.14: Primjer enkripcije i dekripcije riječi "DNA"

izabranog ključa, odabrati pripadajuće blokove, te ih poslati Bobu preko nesigurnog komunikacijskog kanala. Bob zatim u epruvetu koja sadrži samo jedan primjerak $\uparrow k_A$, ulije dobivene blokove te pokrene proces hibridizacije. Time dobije određeni DNA lanac koji je na određenim mjestima dvolančani, a na određenim jednolančani. Tamo gdje je dvolančani (vidi preko mikroskopa), stavlja bit 1, a jednolančani bit 0 te čita dani binarni string s desna na lijevo kreirajući njemu ekvivalentan otvoren tekst u prirodnom jeziku. O različitim načinima kriptiranja pomoću DNA računala može se naći u [5]. Sigurnost ovog kriptosustava može se povećati uzimanjem ključa koji će biti veće duljine (na primjer da se umjesto za svaki bit uzme 10-mer, možemo uzeti 40-mer), i ona ovisi samo o danom ključu (jednokratnoj bilježnici). Nadalje, što je duži tekst koji se šifrira, time će se povećavati i kompleksnost kriptosustava. Tako, na primjer, za kriptiranje riječi *DNA* čija binarna reprezentacija ima 21 bit, prostor ključeva sadrži $4^{21 \cdot 10} = 4^{210}$ različitih ključeva koji mogu šifrirati danu riječ, a za kriptiranje riječi *Umjetnost DNA kriptografije* (uzimajući u obzir da se jedan simbol zamijenjuje jednom 7-bitnom riječi), prostor ključeva sadrži $4^{189 \cdot 10} = 4^{1890}$ ključeva. Naravno nisu svi ključevi jednako jaki - na primjer ključ koji u sebi sadrži 2

bloka koja su jednaka može zapravo interpretirati dva, pozicijski gledano, različita bita, a time, ovisno kakvi su ti bitovi (na primjer jedan 0, a drugi 1), dva različita otvorena teksta, a drugi primjer je postojanje bloka koji ne počinje na poziciji djeljivoj s 10, a jednak je nekom drugom bloku koji počinje sa spomenutom pozicijom, tada se može dobiti da se dani šifrat ne može dekriptirati na nikakav način. Spomenuti problemi bi se riješili na način da se zahtjeva jedinstvenost blokova čija najdulja zajednička riječ nije dulja od $\frac{n}{2} - 1$, gdje n označava duljinu polimera koji kriptira jedan bit, a problem jedinstvenosti polubloka bi se riješio kriptiranjem bita dužim polimerom ($n \geq 10$ u našem primjeru).

2.6 Prednosti i mane DNA računala i DNA kriptografije

Glavna prednost DNA računala je izrazito veliki kapacitet i gustoća memorije zbog koje se stvari mogu lakše izračunavati i gdje se napadi na kriptosustvae s tehnikom grube sile vrlo lako probijaju. Opet rješavanje nekih NP teških problema se također riješavaju istom tehnikom. Druga prednost je masivna paralelnost DNA računala zbog toga što se jedna operacija obavlja nad skupom riječi, a ne na samo jednoj riječi (broj procesora varira - to je kao da broj procesora konstantno varira ovisno o broju podataka u programu). S druge strane DNA računalo može raditi grešku u izračunavanju, odnosno u izvođenju biokemijskih reakcija, o tome kako utjecati na te greške se može naći u [4]. Daljnja mana je ta što je DNA računalo za sada samo teoretski model izračunavanja jer da bi se moglo konstrukirati i fizički ostvariti trebaju se preći još mnoge tehnološke prepreke kao i fizička ograničenja. DNA računalo dobro računa sa, kako se čini, relativno kompliciranim operacijama (npr. $Replikacija$, $Ekstrakcija$), no pri izračunavanju jednostavnih operacija kao što je "ekskluzivno ili", potrebno je izvesti 4 koraka u postupku. Daljnja mana je ta što su operacije na DNA računalu spore u odnosnu na operacije koje se obavljaju na elektromehaničkim računalima. One postaju sporije što je više parova G i C u nekoj dvolančanoj

DNA zbog trostrukih kovalentnih veza koje se teže razbijaju i teže uspostavljaju između tih dviju dušićnih baza te što je veća duljina samih DNA lanaca u epruveti. Ipak, kako područja genetike i medicine jako brzo napreduju, postoji mogućnost otkrića kako da se te operacije ubrzaju primjenom nekih katalizatora ili električne energije.

Bibliografija

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

[2] L. M. Adleman, P. W. K. Rothemund, and S. Roweis. On Applying Molecular Computation To The Data Encryption Standard. *Journal of Computational Biology*, 6(1):53–63, 1999.

[3] D. Boneh, C. Dunworth, and R. J. Lipton. Breaking DES using a molecular computer. In E. B. B. R. J. Lipton, editor, *DNA based computers*, volume 27 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, pages 37–65. American Mathematical Society, Providence, RI, 1996.

[4] D. Boneh, C. Dunworth, R. J. Lipton, and J. Sgall. Making DNA computers error resistant. In L. Landweber and E. Baum, editors, *DNA Based Computers II*, volume 44 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, pages 163–170. American Mathematical Society, Providence, RI, 1999.

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

[6] G. M. Church, Y. Gao, and S. Kosuri. Next-Generation Digital Information Storage in DNA. *Science*, 337(6102):1628, Sept. 2012.

[7] A. Dujella and M. Maretić. *Kriptografija*. Element, 2007.

[8] J. O. Grabbe. The DES algorithm illustrated. http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm. [Online; pristup 11. lipnja 2014.].

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

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

[11] S. Pramanik and S. Setua. DNA cryptography. In *2012 7th International Conference on Electrical Computer Engineering (ICECE)*, pages 551–554, Dec 2012.

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

[13] M. Vuković. *Matematička logika*. Element, 2009.

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

Sažetak

DNA kriptografija predstavlja noviji model kriptiranja u teoriji računarstva, a koji upotrebljava koncept jednokratne bilježnice. Zbog iznimno guste raspodjele u memoriji DNA lanaca te složenim biokemijskim operacijama koje se mogu nad DNA lancima izvršiti paralelno (koncept masivnog paralelizma), nudi noviji pristup u teorijskom izračunavanju. Za razliku od RSA kriptosustava, DNA kriptografija je kvantno rezistentna te nudi sigurnost koja se oslanja na duljinu i jednokratnu upotrebljivost kriptografskog ključa. U ovom radu je predstavljen jedan način kriptiranja pomoću DNA lanaca. Osim toga, prikazan je matematički koncept DNA računala te kratka analiza njegove složenosti te primjena istoga na dva problema: problem nalaženja Hamiltonova puta, te korištenje DNA računala u svrhu napada na DES kriptosustav.

Summary

DNA cryptography, which uses the concept of a one-time pad, is a newer model of encryption in the computer science. Due to the extremely dense distribution in memory of DNA chains, and complex biochemical operations that can be performed over the DNA strands in parallel (the concept of massive parallelism), it provides greater approach to the theoretical calculation. Unlike RSA cryptosystem, DNA cryptography is quantum resistant and offers security that relies on length and one-usability of cryptographic keys. This thesis discusses a method of encryption using DNA chains. In addition, it presents a mathematical concept of DNA computers and a brief analysis of its complexity and the use of the same to solve two problems: the problem of finding Hamiltonian path, and the use of DNA computers in order to attack the DES cryptosystem.

Životopis

Antonio Kovačić rođen je 23. siječnja 1991. godine u Banja Luci, Bosna i Hercegovina. Od 1992. godine, nakon što je počeo Domovinski rat, odrasta u Zagrebu uz oca, majku, sestru i tri brata. Svoje obrazovanje započinje u Osnovnoj školi Tituša Brezovačkog, a nastavlja u Nadbiskupskoj klasičnoj gimnaziji u Zagrebu koju završava 2009. godine. Nakon završetka srednješkolskog obrazovanja upisuje se na preddiplomski studij matematike na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta Sveučilišta u Zagrebu. Godine 2012. upisuje diplomski studij računarstva i matematike na istom fakultetu. Tijekom visokoškolskog obrazovanja učlanjuje se u mješoviti zbor svete Barbare "Panis angelicus" u sklopu kojeg osvaja brojne nagrade na natjecanjima, te sudjeluje na raznim kulturno-umjetničkim događanjima, kako unutar Republike Hrvatske, tako i van nje. Potaknut članovima zbora, te na osobnu želju, na zadnjoj godini studija upisuje se u Glazbenu školu Vatroslava Lisinskog u Zagrebu na Odjel solo pjevanja u klasi Mr. art. Bojana Pogrmilovića. U teoriji računarstva interes pokazuje na području izračunljivosti, složenosti algoritama i kriptografije, a posebnu pažnju usmjerava na saznanja o DNA računalu te njegovoj primjeni.