SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6389

Kriptiranje komunikacije koristeći suparničko učenje evolucijskih algoritama

Ante Gazibarić

Zagreb, lipanj 2019.

SADRŽAJ

1. Uvod

Tajnost informacija uvijek je imala veliku važnost u društvu. Posjedovanje informacija predstavlja moć, a njihova tajnost može presuditi tko će izaći kao pobjednik u ovom kompetitivnom društvu. Upravo zbog toga ljudi pokušavaju otkriti načine zaštite informacija prilikom njihovog prijenosa preko određenog medija kada bivaju izložene raznim napadima. Kroz povijest ljudi su se koristili raznim kriptografskim postupcima kako bi skrili informacije: od Cezara koji je svako slovo poruke zamijenio s odgovarajućim slovom pomaknutim za tri mjesta, do suvremenih kriptografskih algoritama za koje trenutno nemamo niti dovoljno resursa, a ni vremena kako bi dekriptirali poruku grubom silom.

Međutim, uvijek se traže novi sigurniji načini kriptiranja u slučaju da se pronađu mane trenutno korištenih algoritama. Područje računarske znanosti koje sve više svoju primjenu pronalazi u mnogobrojnim granama industrije i koje nosi nova rješenja u području kriptografije je umjetna inteligencija. Području umjetne inteligencije pripadaju i evolucijski algoritmi koji su svoju inspiraciju pronašli u Darwinovoj evoluciji. Izmjenom generacija opstaju samo bolja rješenja koja prosljeđuju svoj genetski materijal na iduće generacije pokušavajući usmjeriti tok algoritma ka optimalnom rješenju. U ovom radu pokušat će se pronaći kriptografski algoritam uz pomoć evolucijskog algoritma koji kroz iteracijski postupak primjenom raznih operatora selekcije, križanja i mutacije omogućuje pronalazak dovoljno dobrog rješenja u velikom prostranstvu mogućih rješenja.

2. Kriptografija

Iz potrebe da sačuvamo tajnost informacija koje prenosimo do nekog odredišta, razvili su se razni algoritmi kriptiranja. Od jednostavnih metoda supstitucije gdje se svaki znak zamjenjuje svojim zamjenskim znakom, došli smo do naprednijih algoritama poput simetričnih i asimetričnih kriptografskih algoritama koji se danas koriste.

Klasični slučaj u komunikaciji prikazuje pošiljatelja koji želi prenijeti određenu informaciju do odredišta preko nesigurnog komunikacijskog kanala. Radi jednostavnosti pridaju se sljedeća imena sudionicima u komunikaciji: Alice predstavlja pošiljatelja poruke, Bob je primatelj, a Eve predstavlja prisluškivača koji na nekom mjestu komunikacijskog kanala ima pristup porukama koje prolaze kanalom, ali nema mogućnost izmjene poruka. Time se narušava tajnost informacija, ali ne i njihov integritet. Alice prije slanja poruka koje nazivamo otvoreni tekst (eng. plaintext), kriptira poruke pomoću ključa i tako dobiva odgovarajuće šifrate (eng. ciphertext). Šifrati zatim stižu do primatelja Boba čiji je zadatak dekriptirati poruke uz pomoć istog ključa koji je Alice koristila. Budući da se kriptiranje i dekriptiranje izvodi pomoću istog ključa riječ je o simetričnom kriptografskom algoritmu. Postoje i asimetrični kriptografski algoritmi koji za kriptiranje i dekriptiranje koriste različite ključeve, ali u ovom radu se neće obrađivati.

2.1. DES

DES algoritam je razvijen ranih 1970-ih prema tadašnjim sigurnosnim potrebama. DES je postavio standard kriptiranja informacija i imao je veliki utjecaj na algoritme razvijene nakon njega. Algoritam šifrira otvoreni tekst u blokovima po 64 bita uz pomoć tajnog ključa duljine 56 bita, a kao rezultat dobijemo šifrat od 64 bita. Upravo duljina ključa predstavlja i njegovu slabost budući da se šifriran tekst može razotkriti i primjenom čiste sile (eng. brute

force) zbog čega se danas smatra nesigurnim i više nije u upotrebi. Algoritam se sastoji od tri etape:

1. Permutiraju se bitovi otvorenog teksta duljine 64 bita pomoću inicijalne permutacije. Kao rezultat toga dobijemo x_0 koji razdvojimo na 32 lijevih L_0 i 32 desnih R_0 bitova, odnosno $x_0 = L_0R_0$.

2. Pomoću određene funkcije f ponavljamo 16 puta sljedeće:

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

   gdje u svakoj iteraciji lijeva polovica postaje desna iz prethodnog koraka, a desna se određuje „ekskluzivnim ili“ (XOR) lijeve polovice iz prethodnog koraka i funkcije f.

3. Konačno, primijenimo inverznu permutaciju bitova i time dobivamo rezultat algoritma

Zbog ranjivosti algoritma nastale su nove inačice kao nadogradnje na postojeći DES poput algoritma 3-DES. On rješava problem koji je prvotna verzija DES imala s veličinom ključa, međutim vremenski je barem duplo zahtjevniji zbog čega dolazi potreba za razvojem boljih algoritama poput IDEA i AES.

2.2. IDEA

IDEA je simetrični blok algoritam koji je prvi put opisan 1991. Nastao je kako bi zamijenio DES i pokrio njegove slabosti. Algoritam za šifriranje koristi ključ duljine 128 bita i šifrira tekst u blokovima po 64 bita. Temelji se na tri osnovne operacije koje su i sklopovski jednostavno izvedive: XOR, zbrajanje modulo 2^{16}, i multiplikacija modula 2^{16+1}. Upravo preplitanjem ovih operacija algoritam dobiva na sigurnosti što ga danas po mnogim stručnjacima čini jednim od najsigurnijih simetričnih algoritama. U početku svog nastanka algoritam je bio patentiran zbog čega je bila potrebna licenca da bi se koristio, međutim patent je istekao 2012. čime je algoritam postao potpuno besplatan za korištenje.

2.3. AES

AES je simetrični kriptografski algoritam koji je proizašao iz natječaja objavljenog 1997. za kriptosustav koji bi trebao zamijeniti DES i postati opće prihvaćen standard. Algoritam šifrira podatke u 128 bitnim blokovima koristeći ključ duljine 128, 192 ili 256 bitova, što je ujedno bio i uvjet natječaja. Algoritam ulazni blok podataka smješta u matricu koja predstavlja matricu stanja. Matrica se sastoji od četiri retka, a u jednom se nalazi broj bitova koji dobijemo tako da podijelimo ukupnu duljinu bloka s 32. Matrica zatim prolazi kroz određeni broj koraka gdje se kroz svaki od koraka izvršavaju četiri transformacije nad oktetima:

1. Zamjena okteta na temelju supstitucijske tablice
2. Posmak redaka u matrici stanja
3. Miješanje podataka unutar svakog stupca matrice stanja
4. Dodavanje podključa u matricu stanja

Broj koraka se određuje na temelju duljine ključa. Konačna vrijednost matrice predstavlja izlazni šifrirani blok podataka.

3. Kartezijsko genetsko programiranje

Mnogi problemi s kojima se susrećemo se mogu prevesti u problem matematičke optimizacije. U takvim problemima za svako rješenje moguće je odrediti njegovu dobrotu, odnosno pridijeliti mu određenu ocjenu koja nam daje mogućnost rangiranja rješenja od boljeg ka lošijem. Ovisno o problemu želja će nam biti pronaći rješenje s maksimalnom ili minimalnom ocjenom, pa tako govorimo o maksimizaciji, odnosno minimizaciji određene funkcije.

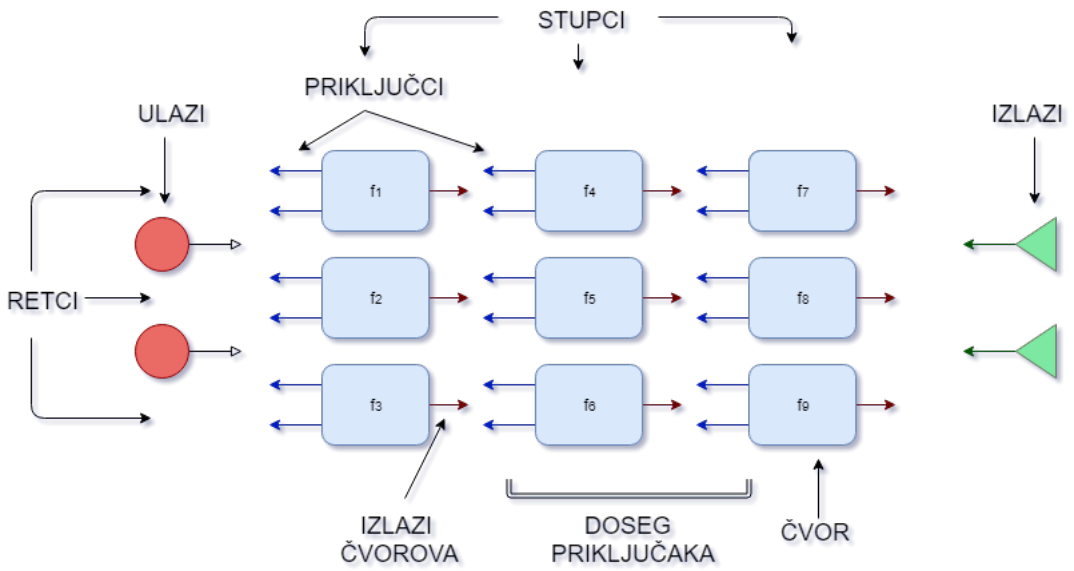
Evolucijski algoritmi se u potrazi za rješenjem upravo koriste optimizacijom gdje kroz iterativni postupak primjenjuju različite operatore kojima uvode male izmjene dosadašnjim rješenjima ne bi li pronašli ono s boljom dobrotom. Iterativni postupak evolucijskih algoritama kao i operatori kojima se koriste, proizašli su iz Darwinove evolucije koja već dugi niz godina optimizira žive organizme kako bi se što bolje prilagodili životu na Zemlji. Jedan od evolucijskih algoritama je i Genetsko programiranje (GP).

3.1. Struktura

GP interno sadrži određenu strukturu koja predstavlja računalni program, odnosno traženo rješenje problema. Upravo odabirom strukture GP-a utječemo na mogućnosti reprezentacije mogućih rješenja. Zbog toga je potrebno odabrati prikladnu strukturu za svaki konkretan problem. Neke od struktura GP-a su: stablo, stog, graf. U ovom radu bit će primijenjena reprezentacija pomoću grafa koja se onda naziva Kartezijsko genetsko programiranje (eng. Cartesian genetic programming, CGP).

CGP računalni program predstavlja usmjerenim grafom koji se zapravo sastoji od dvodimenzijskog polja čvorova. Čvor je osnovna jedinica grafa koji sadrži određenu funkciju, ulazne parametre (priključke) te izlaz koji predstav-

lja rezultat funkcije nad ulazima. Čvor se interno predstavlja kao niz cijelih brojeva gdje se prvi broj odnosi na indeks funkcije u odgovarajućoj tablici koja sadrži sve potrebne definirane funkcije, a ostali brojevi odnose se na priključke koji služe kao ulazi nad kojima se izvršava određena funkcija. Broj priključaka čvora određuje sama funkcija, a oni mogu biti ulazni parametri u naš računalni program ili izlazi iz drugih čvorova. Konačnih izlaza iz programa može biti proizvoljno mnogo i odabire se neki od izlaza iz čvorova ili sam ulazni parametar. Kako bi se kontrolirale mogućnosti za odabir priključaka određenog čvora, uvodi se parametar doseg priključaka (eng. levels-back). Ako postavimo doseg priključaka na vrijednost jedan, tada mogući ulazi u čvor su samo izlazi čvorova iz prethodnog stupca, a u slučaju ako je doseg priključaka jednak broju stupaca tada ne postoji ograničenje na ulaze čvorova. Slika 3.1 prikazuje općenitu strukturu CGP-a.



Slika 3.1: Općenita struktura CGP-a

CGP se interno predstavlja kao lista cijelih brojeva koji čine genotip (eng. genotype). Dekodiranjem genitipa CGP-a dobije se fenotip (eng. fenotype) koji predstavlja jedan konkretan računalni program. Može se primijetiti da određeni čvorovi u genotipu uopće ne moraju biti iskorišteni u fenotipu i nemaju nikakav utjecaj na ponašanje računalnog programa. Upravo to je glavna karakteristika strukture CGP-a gdje više različitih genotipa može proizvesti isti fenotip. Ako čvor ima utjecaja na fenotip takav čvor se naziva aktivan čvor, a čvorovi koji nemaju utjecaj na fenotip nazivaju se neaktivni čvorovi i mogu

doći do izražaja tek nakon primjene nekih od operatora CGP-a o čemu će biti riječ nešto kasnije u radu.

3.2. Tijek algoritma i operatori

U početku algoritma nasumično se stvara početna populacija jedinki, odnosno rješenja. Zatim započinje iterativni postupak u kojem selekcijom želimo izdvojiti bolje jedinke koje potom prolaze kroz operatore križanja i mutacije te dospijevaju u iduću generaciju populacije. Postupak se ponavlja unaprijed određeni broj puta ili se zaustavlja ako smo pronašli traženo rješenje ili dok pretraživanje ne počne konvergirati.

```
Stvori pocetnu populaciju
Izracunaj dobrotu populacije
PONAVLJAJ:
        Selekcija
        Krizanje
        Mutacija
DOK uvjet zaustavljanja nije zadovoljen
```

Slika 3.2: Tijek genetskog algoritma

3.2.1. Funkcija dobrote

Kako bi iz populacije izdvojili naprednije jedinke moramo imati mogućnost ocjene dobrote nekog rješenja. Taj zadatak obavlja funkcija dobrote koju trebamo definirati specifično za svaki konkretan problem. Određivanje funkcije dobrote može biti presudno u pronalasku rješenja zbog toga što nas upravo ta funkcija usmjerava ka traženom rješenju.

3.2.2. Selekcija

Operator selekcije odabire iz populacije jedinke na osnovu njihove dobrote koje će proći kroz operatore križanja i mutacije te proslijediti svoj genetski materijal na iduću generaciju. Upravo mehanizam selekcije omogućuje da bolja rješenja opstaju i time nas usmjerava prema optimalnom rješenju iz generacije

u generaciju. Neki od najčešće korištenih operatora selekcije su proporcionalna selekcija (eng. Roulette wheel selection) i turnirska selekcija (eng. Tournament selection).

Proporcionalna selekcija pridaje svakoj jedinki vjerojatnost odabira proporcionalno njenoj dobroti. Takva selekcija se može slikovitije prikazati pomoću rulteskog kola gdje je svaka jedinka dobila svoj kružni isječak na kolu čija je površina jednaka njenoj dobroti. Kolo zatim zavrtimo te ga zaustavimo pritiskom na jednu točku kruga. Kružni isječak u kojem se našla odabrana točka predstavlja jedinku koju smo izabrali. Kod turnirske selekcije se odabire slučajni uzorak veličine n iz populacije i zatim se odabire jedinka s najvećom dobrotom iz uzorka.

3.2.3. Križanje

Kod operatora križanja selekcijom odabiremo dvije roditeljske jedinke i zatim stvaramo novu jedinku kopiranjem dijela gena iz prve i dijela gena iz druge roditeljske jedinke. Time nastaje nova jedinka s genetskim materijalom oba roditelja koja ulazi u iduću generaciju. Kako imamo više načina za odabir roditeljskih gena, tako razlikujemo više vrsta operatora križanja.

Križanje s jednom točkom prekida (eng. Single-point crossover) određuje nasumično točku u genetskom materijalu roditelja. Nova jedinka dobit će od prvog roditelja dio gena lijevo od točke, a od drugog roditelja desno od točke. Križanje s n-točaka prekida (eng. n-point crossover) radi na istom principu kao i s jednom točkom, samo sada imamo n prekida koje trebamo kopirati novoj jedinci od jednog, odnosno drugog roditelja. Kod jednolikog križanja (eng. Uniform crossover) svaki gen nove jedinke se bira s jednakom vjerojatnošću od nekog od dva roditelja.

3.2.4. Mutacija

Kako ne bi bili ograničeni na samo kombinaciju rješenja dobivenima križanjem jedinki iz inicijalne populacije uvodi se operator mutacije koji je ključan u pretraživanju što većeg prostora mogućih rješenja. Operator mutacije s određenom vjerojatnošću mutira neki gen odabrane jedinke, odnosno dodjeljuje joj novu vrijednost odabranog gena iz njegove domene.

Najjednostavnija mutacija u CGP-u je mutacija jedne točke (eng. Point mu-

tation). Izvodi se na način da se nasumično odabire jedan gen koji može biti gen funkcije čvora, priključka čvora ili neki od izlaza. Na osnovu vrste gena kojeg smo odabrali pridjeljujemo mu novu vrijednost iz pripadajuće domene. Možemo primijetiti da u ovom slučaju možemo odabrati gen koji pripada neaktivnom čvoru pa mutacija neće imati utjecaj na fenotip.

Nadogradnja ove mutacije može se izvesti tako da se izvršava mutacija jedne točke sve dok nismo mutirali aktivni gen (eng. New parameterless mutation procedure).

Iz podjele čvorova na aktivne i neaktivne proizašli su novi operatori mutacije specifični za CGP. Takozvana „tiha“ mutacija (eng. Silent mutation) mutira neki od neaktivnih čvorova. Postavlja se pitanje zašto bi koristili takvu mutaciju ako nema nikakvog utjecaja na fenotip. Mutacija sama za sebe nema koristi u pronalaženju novih rješenja, međutim u kombinaciji s drugim operatorima mutacije njene promjene dolaze do izražaja. Upravo suprotno radi takozvana „netiha“ mutacija (eng. Non-silent mutation). Ona nasumično odabire neki od neaktivnih čvorova i pretvara ga u aktivni. To pretvaranje se izvršava tako da se odabrani neaktivni čvor prespoji na neki od aktivni čvorova koji su bliže izlazima (u grafu desno od njega) ili na sam izlaz. Sve mutacije moraju se pridržavati ograničenja postavljenim u strukturi CGP-a.

4. Kriptiranje komunikacije koristeći CGP

4.1. Uvod u rad

Pitanje koje se postavlja u ovom radu: može li se proizvesti kriptografski algoritam pomoću optimizacije? Konkretno, možemo li iskoristiti strukturu CGP-a i pomoću nje prikazati kriptografski algoritam kao usmjereni graf jednostavnih kriptografskih funkcija? Obilaskom grafa prolazimo kroz čvorove čije funkcije vrše izmjene nad ulaznom porukom koju time pretvaraju u šifrat. Time smo dobili kriptografski algoritam. Budući da je poprilično teško ručno osmisliti dobar niz funkcija tu zadaću prepuštamo CGP-u koji će kroz postupak evolucije proizvesti optimalan kriptografski algoritam. Za to nam je potreban način da razdvojimo bolje algoritme od onih lošijih.

Prikladan način ocjenjivanja za ovaj problem nudi suparničko učenje objašnjeno u poglavlju 4.5. Funkcije koje će vršiti izmjene nad ulaznim tekstom bit će detaljnije obrađene u poglavlju 4.3. Kvaliteta kriptografskog algoritma može se gledati i iz aspekta kriptografije gdje algoritam mora zadovoljiti određene kriptografske kriterije, o čemu će biti riječ u poglavlju 4.6.

4.2. CGP primjenjen u kriptografiji

Kako bi iskoristili ponuđenu strukturu CGP-a, evaluaciju ćemo morati prilagoditi konkretnom problemu koji se u ovom radu obrađuje, a to je kriptiranje, odnosno dekriptiranje. Na početku evaluacije imamo ulaznu poruku proizvoljne veličine. Nju ćemo gledati kao tok ulaznih bitova, a funkcije će vršiti izmjene nad trenutno promatranim bitom poruke. Svaki put kada neki čvor primjeni svoju funkciju nad promatranim bitom, promatrani bit se posmiče za

jedno mjesto. Za razliku od uobičajene evaluacije CGP-a koja prolazi usmjerenim grafom s lijeva na desno, ovdje će se izvršavati obrnutim smjerom. Prednosti obrnutog smjera izvođenja bit će vidljivi kod funkcija koje omogućuju uvjetno grananje. Evaluacija započinje tako da nam izlaz iz CGP-a predstavlja zapravo ulaznu točku programa i određuje put kojim ćemo prolaziti kroz graf. Prvi čvor u kojeg ulazimo bit će čvor na kojeg izlaz pokazuje. Kroz graf dalje se usmjeravamo pomoću priključaka čvorova. Većina funkcija određuje samo jedan priključak tako da je sljedeći čvor kojim ćemo proći upravo taj koji je spojen na priključak trenutnog čvora. Kako bi omogućili uvjetno grananje uvodimo funkciju koja ima dva priključka, a priključak kojim će se nastaviti kretanje kroz graf bira se na osnovu trenutno promatranog bita (primjer takve funkcije je funkcija IF). Kada dođemo do čvora koji nema više priključaka (skroz lijevo u grafu), vraćamo se na početak grafa i sljedeći čvor nam je ponovno čvor na kojeg izlaz pokazuje. Prolazak kroz graf se izvršava sve dok nismo obradili čitavu ulaznu poruku, tj. dok trenutni bit nije veći od veličine poruke. Ovakva evaluacija predstavlja proces i kriptiranja i dekriptiranja.

Može se primijetiti da postoje odstupanja od klasične evaluacije CGP-a i zbog toga su korišteni neprimjereni termini za strukturu CGP-a. Izlaz u klasičnom CGP-u ovdje zapravo predstavlja ulaz u graf, a priključci čvorova koji su inače ulazni parametri funkcije su zapravo ovdje iskorišteni kao način usmjeravanja kroz graf.

4.3. Funkcije čvorova

Funkcije koje vrše izmjene nad ulaznim tokom bitova detaljnije su opisane u ovom poglavlju. Sve funkcije imaju jedan priključak, osim funkcije IF koja ima dva. Funkcije su sljedeće:

- **XOR1** – nad trenutnim bitom vrši se operacija XOR s jedinicom
- **XOR0** – nad trenutnim bitom vrši se operacija XOR s nulom
- **ROTL** – trenutni bajt se rotira za jedno mjesto u lijevo
- **ROTR** – trenutni bajt se rotira za jedno mjesto u desno
- **IF** – ako je trenutni bit 1 put kroz graf se nastavlja prvim priključkom, inače drugim. Trenutni bit ostaje neizmijenjen

- **XOR** – nad trenutnim bitom poruke se vrši operacija XOR s odgovarajućim bitom ključa. Odgovarajući bit ključa je trenutno promatrani bit nad kojim se izvrši operacija modulo s brojem bitova ključa
- **XOR BYTE** – nad trenutnim bajtom poruke se izvrši operacija XOR s odgovarajućim bajtom ključa. Odgovarajući bajt ključa je trenutno promatrani bajt nad kojim se izvrši operacija modulo s veličinom ključa
- **SWAP** – funkcija zamjenjuje dva bita na proizvoljnim mjestima. Mjesta su definirana na način da prvi put kada se naiđe na funkciju SWAP trenutno promatrani bit se pamti. Zamjena slijedi idući put kada naiđemo na istu funkciju kada trenutno promatrani bit zamjenjuje mjesto s ranije zapamćenim bitom

Sljedeća podjela funkcija će biti bitna kod dodjele funkcija određenim ulogama u komunikaciji:

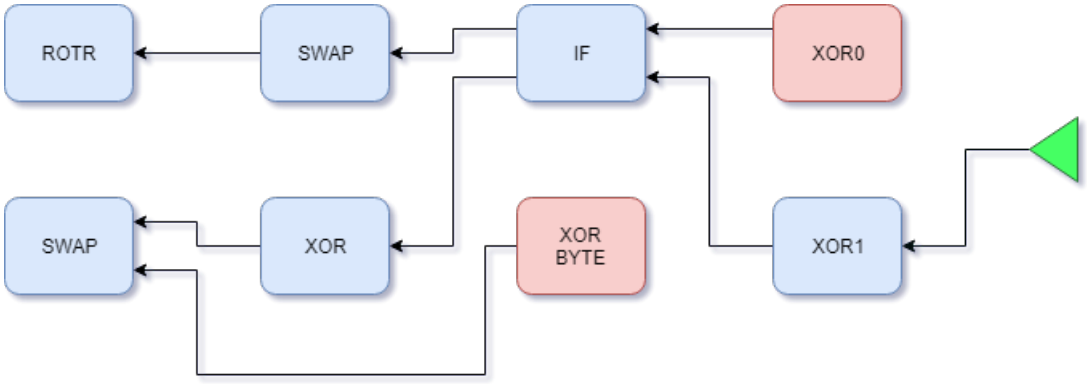
- Funkcije koje koriste ključ: **XOR, XOR BYTE**
- Funkcije koje ne koriste ključ: **XOR1, XOR0, ROTL, ROTR, IF, SWAP**

4.4. Primjer evaluacije

Primjer evaluacije dan je na grafu sa slike 4.1 koji ima sljedeće karakteristike:

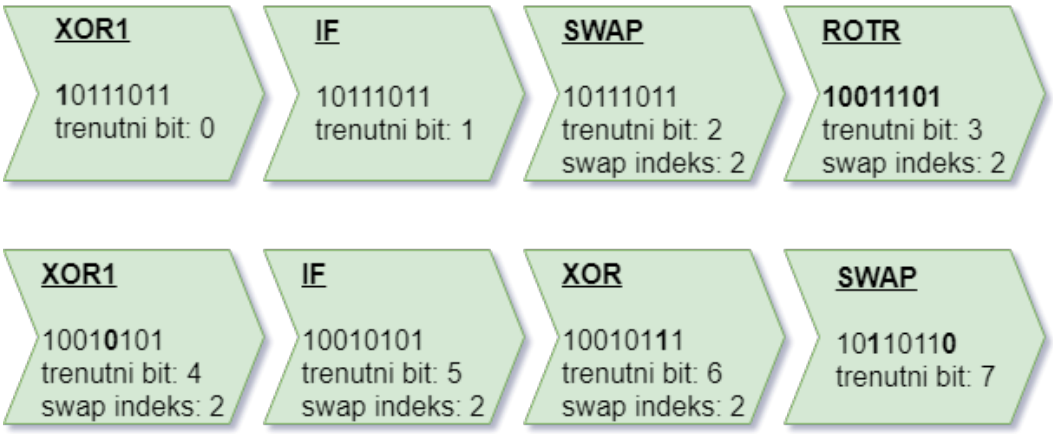
- Broj redaka: 2
- Broj stupaca: 4
- Broj izlaza: 1
- Doseg priključaka: 2

Zeleni trokut predstavlja ulaznu točku u graf. U čvorovima grafa prikazane su funkcije čvora te pripadajući priključci koji povezuju graf. Dodatno, aktivni čvorovi su obojeni plavom, a neaktivni crvenom bojom.



Slika 4.1: Primjer strukture CGP-a

Graf nema ograničenja kod izbora funkcija. Ulazna poruka na kojoj će se provesti evaluacija je 00111011, a korišteni ključ je 10100110. Svaki korak evaluacije prikazan je na slici 4.2. Jedan korak prikazuje stanje nakon primijene funkcije pojedinog čvora gdje su prikazane sljedeće informacije: funkcija čvora, poruka nakon primijenjene funkcije (izmijenjeni bitovi su zadebljani), trenutno promatrani bit (odnosi se na indeks bita u ulaznoj poruci s indeksiranjem od nule), te dodatne informacije kao što je zapamćeni indeks funkcije SWAP.



Slika 4.2: Tijek izvođenja evaluacije CGP-a

Evaluacijom CGP-a se iz ulazne poruke 00111011 dobila izlazna poruka 10110110. Budući da se proces evaluacije kriptiranja i dekriptiranja vrši na identičan način, ulazna poruka je mogla biti i otvoreni tekst i šifrat, pa bi odgovarajuća izlazna poruka predstavljala šifrat, odnosno dobiveni otvoreni tekst.

4.5. Suparničko učenje

Klasična situacija u kriptografiji prikazana je sljedećim rasporedom uloga: Alice predstavlja pošiljatelja koji kriptira poruke pri slanju komunikacijskim kanalom. Pri procesu kriptiranja koristi se algoritam simetrične kriptografije gdje Alice koristi ključ prilikom kriptiranja. Ključ je tajan svima osim primatelju kojemu su namijenjene poruke, a to je Bob. Bob koristeći isti ključ dekriptira poruke. Prijenos poruka odvija se nesigurnim komunikacijskim kanalom što želi iskoristiti Eve i otkriti poruke koje se prenose. Međutim, Eve nema pristup ključu kojim su poruke kriptirane što joj otežava posao.

U želji da Alice pokuša kriptirati poruke što je bolje moguće kako ih Eve ne bi mogla raspoznati, a u isto vrijeme Bob da otkrije poslanu poruku, proizvest ćemo simetrični kriptografski algoritam pomoću evolucije CGP-a gdje će Alice predstavljati način kriptiranja, a Bob dekriptiranja poruke. Takav uređaj gdje se više modela međusobno natječu kako bi ostvarili svoje ciljeve koji su međusobno protive jedan drugom, gdje bolja dobrota jednog modela povlači lošiju dobrotu drugog nazivamo suparničko učenje. Za primjer možemo zamisliti slučaj u kojem Alice proizvede jednostavan algoritam kriptiranja gdje će Bob lako otkriti kriptirane poruke što doprinosi dobroti Alice. Međutim, isto tako će i Eve zbog jednostavnog algoritma uspjeti dokučiti o čemu oni govore u njihovoj komunikaciji, što će s druge strane povući lošiju dobrotu Alice. Ako modele pustimo međusobno da se natječu očekujemo da će s vremenom Alice i Bob pronaći način međusobne komunikacije, odnosno razvit će kriptografski algoritam takav da Eve ne može otkriti poruke koje njih dvoje razmjenjuju.

Suparničko učenje primijenjeno na CGP-u objašnjeno je u nastavku. Algoritam započinje nasumičnim generiranjem populacije za Alice koja će pretvarati poruke u šifrate. Broj poruka, veličina poruke kao i veličina ključa mogu se modificirati. Za svaku Alice u populaciji stvara se pripadajuća populacija za Boba te populacija za Eve kojima proslijeđujemo podatke potrebne za ocjenu njihove dobrote. Eve dobiva listu otvorenih poruka i pripadajuće šifrate, a Bob uz to dodatno dobiva ključ korišten pri kriptiranju. Zadatak Boba i Eve je identičan: dekriptirati dobivene šifrate i dobiti odgovarajuće otvorene poruke. Razlika je u tome što Bob smije koristiti sve funkcije navede u poglavlju 4.3., a Eve samo one koje ne koriste ključ.

```
stvoriPocetnuPopulacijuZaAlice()
DOK trenutnaGeneracija <= maxBrojGeneracija
   za svaku Alice iz populacije:
      podaci = Alice.generirajPodatke()
      Bob.pokreniEvoluciju(podaci)
      Eve.pokreniEvoluciju(podaci)
      Alice.ocjeniDobrotu(Bob.najboljiBob(), Eve.najboljaEve())
```

Slika 4.3: Algoritam suparničkog učenja

Za dobrotu Boba i Eve želimo minimizirati broj pogrešnih bitova u poruci, tako da je dobrota određena kao ukupni broj bitova koji se razlikuju u traženoj originalnoj poruci i poruci dobivenoj dekriptiranjem pripadajućeg šifrata, odnosno dobrota predstavlja ukupni broj pogrešnih bitova. Najbolja (minimalna) dobrota za Bob i Eve je nula (svi bitovi su ispravno dekodirani), a najlošija iznosi ukupni broj bitova u svim porukama (broj poruka * veličina poruke * 8).

$$dobrota\_Boba = broj\_pogresno\_dekriptiranih\_bitova\_Boba \tag{4.1}$$

$$dobrota\_Eve = broj\_pogresno\_dekriptiranih\_bitova\_Eve \tag{4.2}$$

Za dobrotu Alice vrijedi sljedeće: što je Bob bolji u dekriptiranju poruka to je bolja i Alice, a što je Eve bolja u dekriptiranju to je Alice lošija. Nakon završenog algoritma evolucije za Boba i Eve, dobrota Alice se određuje pomoću najboljeg Boba i najbolje Eve koju smo uspjeli pronaći, gdje se također koristi minimizacija funkcije. Doprinos dobrote od Boba dobije se tako da se dobroti Alice pridoda dobrota najboljeg Boba, odnosno broj pogrešnih bitova koje je Bob dobio dekriptiranjem.

$$doprinos\_Boba = dobrota\_najboljeg\_Boba \tag{4.3}$$

Kako bi odredili doprinos dobrote od Eve moramo malo bolje analizirati slučajeve koji bi se mogli pojaviti prilikom dekriptiranja. Što se može dogoditi u slučaju da Eve pogriješi sve bitove? Tada jednostavno invertiranjem svakog bita dobiva originalnu poruku. Stoga, gledajući iz perspektive Alice, ne bi se trebala uzimati za cilj maksimizacija pogrešnih bitova Eve. Eve bi se trebala

ponašati kao da nasumično pogađa bitove. U tom slučaju Eve bi trebala u prosjeku pola bitova generirati ispravno, a pola pogrešno. Stoga, udio dobrote od Eve koji pridodajemo dobroti Alice dan je izrazom (4.4), gdje N predstavlja ukupni broj bitova, a $N_{EveWrong}$ broj pogrešno dekriptiranih bitova od najbolje Eve.

$$doprinos\_Eve = apsolutna\_vrijednost(N/2 - N_{EveWrong}) \tag{4.4}$$

Može se primijetiti da se ova komponenta dobrote minimizira u slučaju kada Eve pogriješi točno pola od ukupnog broja bitova, što je bio i cilj gledano iz perspektive od Alice. Konačna dobrota Alice dobiva se kao linearna kombinacija doprinosa od Boba i Eve po izrazu (4.5). Konstante $A1$ i $A2$ dodatno mogu promijeniti udio određenog doprinosa u ukupnoj dobroti Alice. Kako bi se izjednačili udjeli doprinosa od Bob i Eve konstanta $A1$ postavljena je na vrijednost 1, a $A2$ na vrijednost 2. Time se postiglo da je maksimalni doprinos od Boba i Eve jednak upravo ukupnom broju bitova u porukama, a minimalni doprinos iznosi nula.

$$dobrota\_Alice = A1 * doprinos\_Boba + A2 * doprinos\_Eve \tag{4.5}$$

Korisno je analizirati složenost algoritma suparničkog učenja, odnosno broj evaluacija koje je potrebno izvršiti. Ako za Alice, Boba i Eve respektivno označimo veličnu populacije s P_{Alice}, P_{Bob} i P_{Eve}, broj generacija s G_{Alice}, G_{Bob} i G_{Eve} onda je broj potrebnih evaluacija sljedeći:

$$broj\_evaluacija = G_{Alice} * P_{Alice} * (G_{Bob} * P_{Bob} + G_{Eve} * P_{Eve}) \tag{4.6}$$

Zbog stvaranja zasebne populacije Boba i Eve za svaku Alice kroz svaku generaciju može se primijetiti da povećanjem nekog od ovih parametara broj evaluacija se znatno poveća.

4.6. Kriptografski kriteriji

Kriptografski kriteriji predstavljaju svojstva koja bi kriptografski algoritam trebao zadovoljiti kako bi u praksi bio primjenjiv. U ovom poglavlju navedeni su kriteriji bijekcije, iskorištenost ključa i sličnost između otvorenog teksta i šifrata.

4.6.1. Bijekcija

Svaki kriptografski algoritam bi trebao biti bijektivna funkcija koja preslikava otvoreni tekst u jedinstven šifrat kako bi se omogućilo jedinstveno dekriptiranje. Bijektivnost uključuje injektivnost i surjektivnost. Funkcija je injektivna ako je preslikavanje iz jednog skupa u drugi „jedan na jedan“, odnosno za svaki ulazni otvoreni tekst postoji točno jedan šifrat. Funkcija f je surjektivna ako za svaki y iz kodomene, u ovom sličaju šifrat, postoji odgovarajući x iz domene, tj. otvoreni tekst, tako da vrijedi $f(x) = y$, gdje je f funkcija kriptiranja. Ocjena bijekcije dana je sljedećim izrazom:

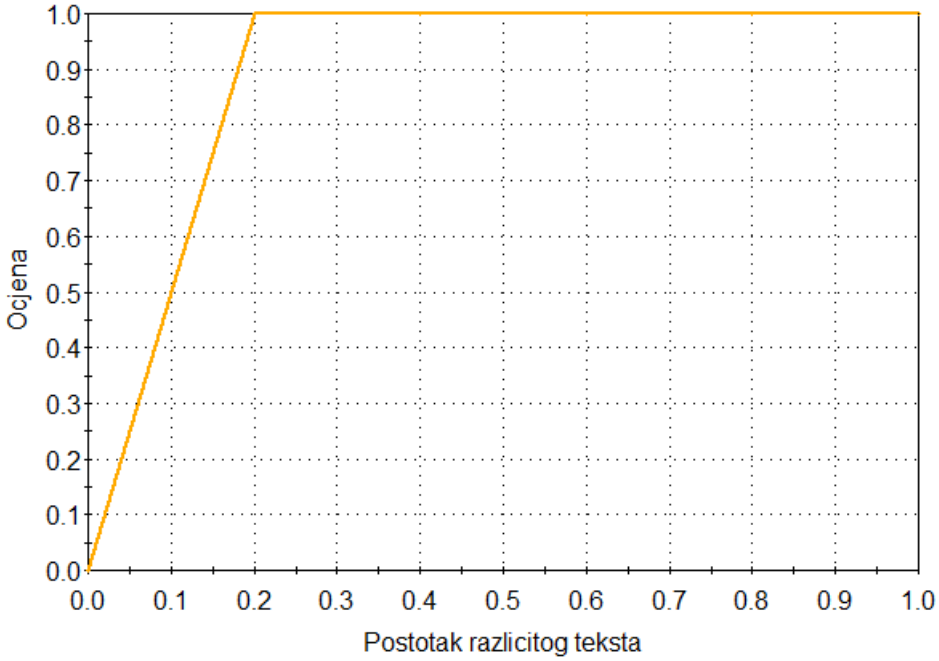
$$ocjena\_bijekcije = \frac{broj\_razlicitih\_sifrata}{broj\_generiranih\_sifrata} \tag{4.7}$$

4.6.2. Iskorištenost ključa

Što je više bitova ključa iskorišteno prilikom kriptiranja to su šifrati sigurniji od potencijalnih napada, budući da informacija o ključu napadaču nije poznata. Ocjena iskorištenosti ključa dobiva se kao udio iskorištenih bitova ključa u ukpnom broju bitova ključa.

4.6.3. Sličnost otvorenog teksta i šifrata

Ovaj kriptografski kriterij zahtjeva dovoljnu različitost između otvorenog teksta i pripadajućeg šifrata. Računa se kao udio različitih bitova između otvorenog teksta i šifrata u odnosu na ukupni broj bitova. Ocjena kriterija prikazana je na slici 4.4.



Slika 4.4: Graf ocjene sličnosti otvorenog teksta i šifrata

5. Eksperimenti i rezultati

Gledajući uloge koje imamo u komunikaciji zanimljiva su tri uređaja prisutnih uloga:

1. U komunikaciji sudjeluju samo Alice i Bob gdje se za nasumično generiranu Alice pokušava pronaći Bob koji će najbolje dekriptirati šifrate dobivene od Alice

2. U komunikaciji sudjeluju Alice i Eve. Ovaj eksperiment se može provesti na dva načina:

   (a) za određenu Alice pokušava se pronaći Eve koja će bez informacije o ključu dekriptirati poruke, gdje se izvodi samo evolucija za Eve

   (b) istodobno evoluiraju i Alice i Eve gdje se pokušava pronaći što bolja Alice

3. U komunikaciji su prisutne sve tri uloge gdje pomoću suparničkog učenja tražimo najbolju Alice, tj. najbolji algoritam kriptiranja, gdje pripadajući najbolji Bob predstavlja način dekriptiranja

Za navedene slučaje u komunikaciji provedeni su eksperimenti i dobiveni rezultati su prikazani u nastavku. Parametri korišteni za CGP prilagođeni su svakom od tri slučaja te su prikazana mjerenja koja pokazuju kako se mijenjaju performanse s promjenom nekih od odabranih parametara. U svim eksperimentima korištena je proporcionalna selekcija koja se pokazala kao prikladna za ovaj problem. Vjerojatnost križanja i mutacije iznosi 0.5.

5.1. Alice i Bob

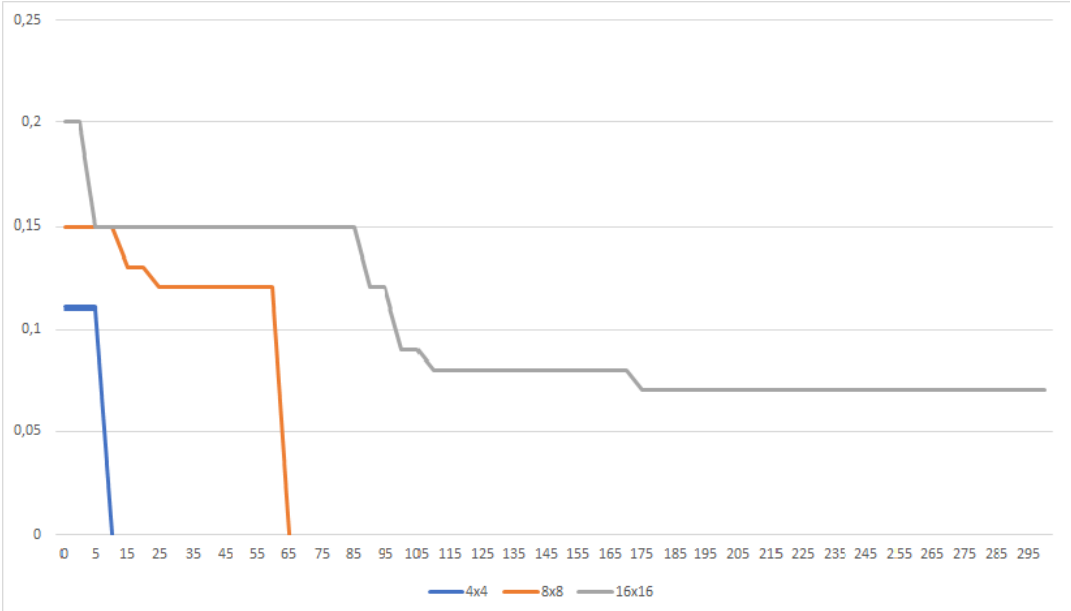
U prvom eksperimentu cilj će biti pronaći Boba koji će uspješno dekriptirati poruke za nasumično generiranu Alice. Sve funkcije koje posjeduje Alice po-

sjeduje i Bob, oboje su istih dimenzija grafa, a ostali parametri koji su korišteni su sljedeći:

Veličina populacije	100	Veličina ključa	1 byte
Broj generacija	300	Broj poruka	10
Doseg priključaka	1	Veličina poruke	5 byte

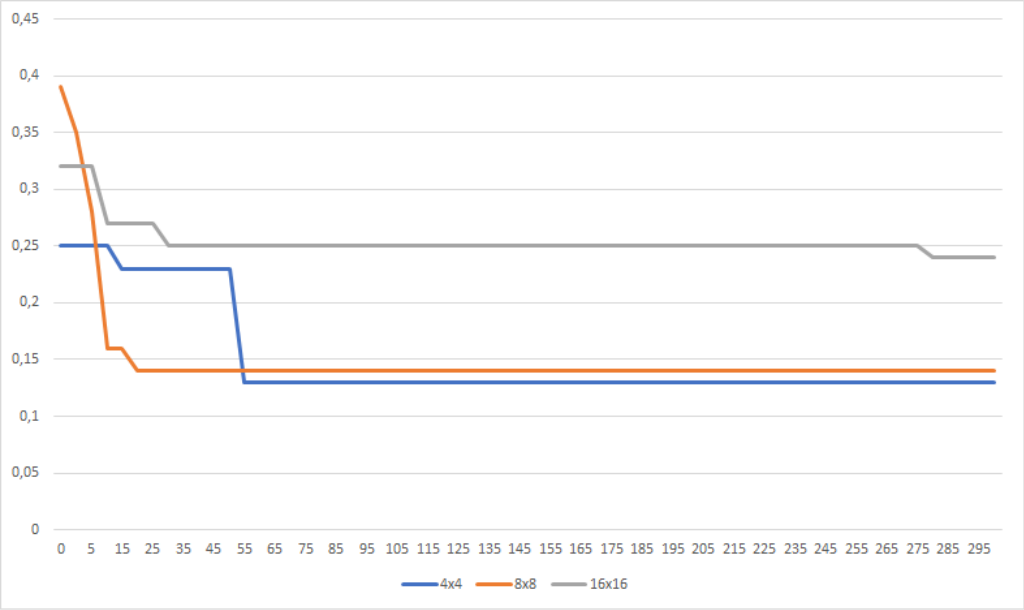
Tablica 5.1: Korišteni parametri u eksperimentu s Alice i Bob

Dobrota Boba određena je izrazom (4.1) i skalirana je na interval $[0, 1]$. Slika 5.1 prikazuje napredak Boba kroz generacije prikazane na x-osi za tri različite strukture grafa (broj redaka x broj stupaca), a y-os prikazuje dobrotu Boba. Alice i Bob pri ovom eksperimentu nisu koristili funkcije rotacije (ROTL i ROTR). Pri jednostavnijim grafovima Bob vrlo brzo uspije dekodirati kriptirane poruke od Alice, što se može vidjeti za grafove 4x4 i 8x8. Međutim, složenijii grafovi kao što je 16x16 predstavljaju problem pri dekodiranju poruka gdje je Bob uspio doći na otprilike 30 pogrešnih bitova.



Slika 5.1: Dobrota Boba kroz generacije za eksperiment bez funkcija rotacije

Drugi eksperiment za Alice i Boba dodatno je uključivao funkcije rotacije, što se pokazalo kao otežavajući faktor pri dekodiranju poruka. Sa slike 5.2 može se vidjeti da Alice i Bob nisu uspjeli uspostaviti komunikaciju niti za jednu strukturu grafa.



Slika 5.2: Dobrota Boba kroz generacije za eksperiment sa svim funkcijama

5.2. Alice i Eve

5.2.1. Evolucija Eve

U drugom eksperimentu testirat će se vrlo značajan slučaj u kojem Eve pokušava rekonstruirati poruke bez ikakvih informacija o ključu. Budući da Eve nema pristup ključu njen zadatak će biti kombiniranjem više funkcija koje ne koriste ključ pronaći način dekriptiranja tako da iz šifrata dobije izvorni otvoreni tekst.

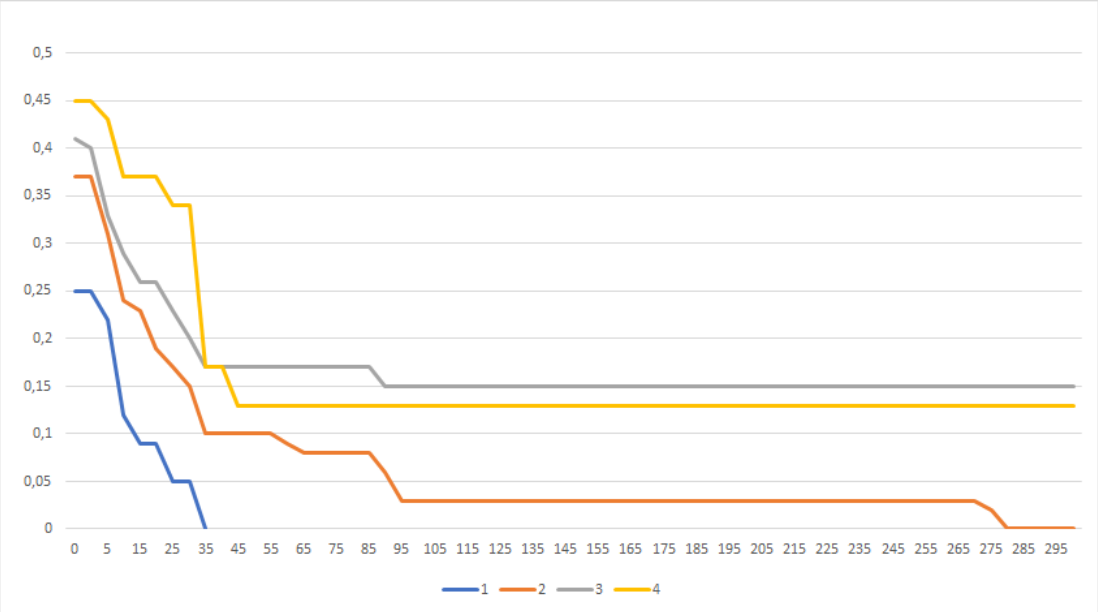
U slučaju da nasumično generirana Alice bude izgrađena samo od funkcija koje ne koriste ključ, tada se vraćamo na problem koje je obrađen u prvom eksperimentu zbog toga što Eve posjeduje sve te funkcije. Zbog toga u ovom eksperimentu za Alice su izbačene sve funkcije koje ne koriste ključ. Također, mora se uzeti u obzir slučaj kada su šifrati koje Alice proizvede jednaki otvorenom tekstu, a to se može dogoditi ako Alice koristi funkciju XOR BYTE nad svakim bajtom otvorenog teksta paran broj puta. Budući da je taj slučaj vrlo jednostavan za Eve izbacit će se mogućnost da Alice koristi funkciju XOR BYTE. Za Alice ostaje samo funkcija XOR koja će nad cijelim ulaznim otvorenim tekstom primijeniti funkciju XOR s odgovarajućim bitom ključa. Za Alice je dovoljan jedan čvor koji će imati funkciju XOR, a budući da Eve mora nekako

nadoknaditi nedostatak informacije o ključu, za nju je korištena kompleksnija struktura s grafom 16x16. Parametri korišteni u ovom eksperimentu prikazani su u tablici:

Veličina populacije	100	Graf	16x16
Broj generacija	300	Broj poruka	10
Doseg priključaka	1	Veličina poruke	5 byte

Tablica 5.2: Korišteni parametri u eksperimentu s Alice i Eve

Provedena su četiri mjerenja za različite veličine ključa, gdje se veličina kretala od jednog do četiri bajta. Dobrota Eve određena je izrazom (4.2) i skalirana je na interval $[0, 1]$. Slika 5.3 prikazuje kretanje minimalne dobrote Eve u populaciji kroz generacije u jednom pokretanju eksperimenta. Brojevi prikazani pri dnu slike koji označuju pojedine linije na grafu odgovaraju upravo veličini ključa. Sa slike 5.3 se vidi da Eve za veličinu ključa od jednog bajta (plava linija) bez problema uspije dekriptirati poruke od Alice za već 35 generacija. S povećanjem veličine ključa dekriptiranje se znatno otežava i graf 16x16 postaje nedovoljan za ovaj problem.



Slika 5.3: Dobrota Eve kroz generacije s različitim veličinama ključa

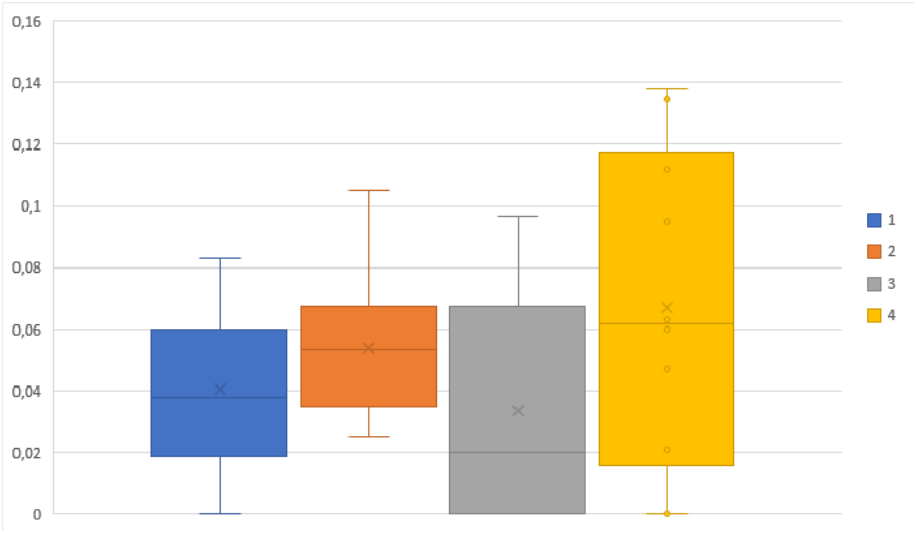
Nad ovim uređajem uloga također su ispitani utjecaji različitih operatora mutacije. Provedena su četiri eksperimenta na deset pokretanja. Operatori

koji su korišteni navedeni su u tablici 5.3 kao i oznaka jesu li se koristili u navedenom eksperimentu. Korišteni parametri ostali su isti kao u prethodnom eksperimentu (vidljivo u tablici 5.2) s veličinom ključa od jednog bajta. Dobiveni rezultati prikazuju raspršenost mjerene vrijednosti, a to su minimalna, maksimalna i prosječna dobrota populacije zabilježene po završetku evolucije.

Eksperiment	Point Mutation	Non-silent mutation	Silent mutation	New parameterless mutation procedure
1	+	-	-	-
2	+	+	-	-
3	+	+	+	-
4	+	+	+	+

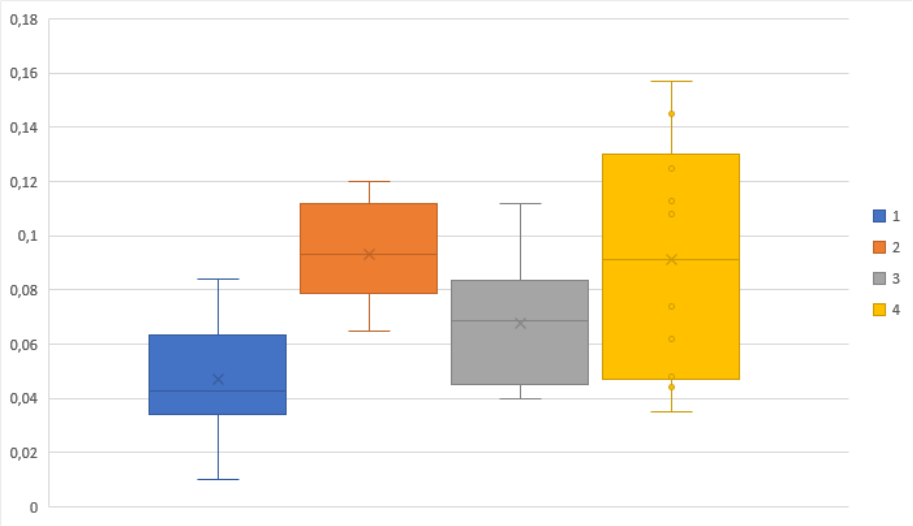
Tablica 5.3: Korišteni operatori mutacije u pojedinim eksperimentima s Alice i Eve

Slike 5.4, 5.5 i 5.6 na y osi prikazuju dobrotu Eve koja se kreće u intervalu $[0, 1]$. Svaki od eksperimenata označen je određenim rednim brojem koji odgovaraju brojevima iz tablice 5.3, a na slikama su prikazani uz desni rub. Iz dobivenih rezultata može se primijetiti trend povećanja raspršenosti podataka, odnosno njihove standardne devijacije s uključivanjem većeg broja korištenih operatora mutacije. Povećanjem broja korištenih operatora mutacije omogućuje se opširnije pretraživanje prostora mogućih rješenja što rezultira većom raznovrsnošću rješenja u populaciji.



Slika 5.4: Minimalna dobrota populacije za Eve

Slika 5.5: Maksimalna dobrota populacije za Eve



Slika 5.6: Prosječna dobrota populacije za Eve

5.2.2. Evolucija Alice i Eve

U ovom eksperimentu Alice i Eve međusobno evoluiraju. Algoritam izvođenja sličan je kao u sljedećem poglavlju s prisutnim svim trima ulogama u komunikaciji, ali uz isključenje Boba. Ovim eksperimentom se želi utvrditi može li Alice pronaći dovoljno kompleksan algoritam kriptiranja takav da Eve ne uspije dekriptirati šifrate, s tim da Alice ne mora brinuti o Bobovoj mogućnosti dekriptiranja, jer on ne sudjeluje u komunikaciji. Dobrota Alice određena je izrazom (4.5), gdje je iznos konstante $A1$ postavljen na nulu, tako da je dobrota Alice određena samo dobrotom najbolje Eve. Dobrota Eve određena je

izrazom (4.2). Alice smije koristiti sve dostupne funkcije, dok Eve samo one koje ne koriste ključ. Korišteni parametri su navedeni u tablici 5.4, a dobiveni rezultati uzeti po završetku jednog pokretanja eksperimenta prikazani su u tablici 5.5 gdje je dobrota prikazana u intervalu $[0, 1]$, a najbolja dobrota iznosi nula. Budući da Alice želi zadržati Eve na otprilike 50% točno pogođenih bitova, iz rezultata se može vidjeti da je Alice poprilično uspješno obavila svoj zadatak. Pripadajuća najbolja Eve pogrešno je generirala 45.5% bitova (182 od ukupno 400 bitova).

	Alice	Eve
Veličina populacije	20	25
Broj generacija	20	50
Graf	4x4	8x8
Doseg priključaka	1	1
Veličina ključa	1 byte	
Broj poruka	10	
Veličina poruke	5 byte	

Tablica 5.4: Korišteni parametri u eksperimentu gdje Alice i Eve međusobno evoluiraju

Alice	Minimalna dobrota	Maksimalna dobrota	Prosječna dobrota	Dobrota pripadajuće najbolje Eve
	0.09	0.37	0.21	0.455

Tablica 5.5: Rezultati eksperimenta gdje Alice i Eve međusobno evoluiraju

5.3. Alice, Bob i Eve

Kada se utvrdilo da su i Bob i Eve u mogućnosti obaviti svoj zadatak dekriptiranja, treći eksperiment će uključivati sve tri uloge u komunikaciji nad kojima će biti primijenjen algoritam suparničkog učenja iz poglavlja 4.5. Dobrota Alice određena je izrazom (4.5), dobrota Boba izrazom (4.1), a dobrota Eve izrazom (4.2). Budući da je složenost ovog problema poprilično velika broj generacija kao i veličina populacije ograničeni su na nešto manje vrijednosti kako bi se algoritam izveo u razumnom vremenu. Korišteni paramteri

su navedeni u sljedećoj tablici:

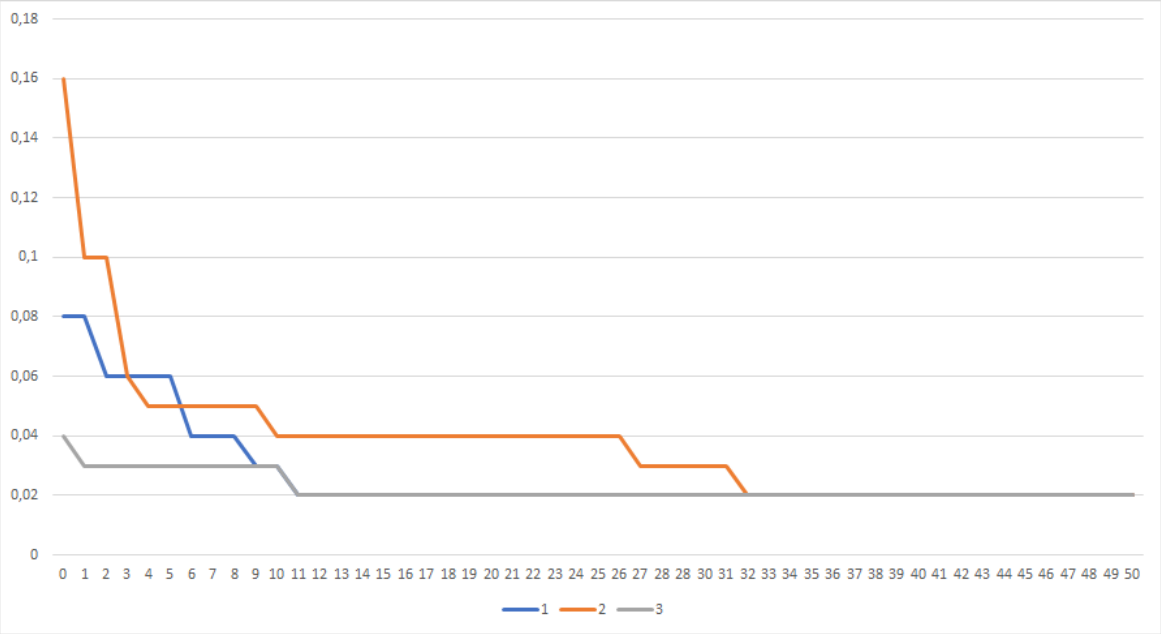
	Alice	Bob	Eve
Veličina populacije	25	10	10
Broj generacija	50	10	10
Graf	4x4	4x4	8x8
Doseg priključaka	2	2	1
Veličina ključa	1 byte		
Broj poruka	10		
Veličina poruke	5 byte		

Tablica 5.6: Korišteni parametri u eksperimentu s Alice, Bob i Eve

Provedena su tri eksperimenta gdje su za prvi eksperiment Alice i Bob koristili sve funkcije osim funkcija rotacije, a Eve je koristila sve funkcije koje ne koriste ključ. Drugi eksperiment je uključivao i funkcije rotacije, dok je u trećem eksperimentu dodatno povečana veličina ključa na 4 bajta. Tablica 5.7 prikazuje rezultate za svaki od eksperimenata gdje je za sve tri uloge prikazana najbolja dobrota. Treba napomenuti da se najbolja dobrota Alice, koja iznosi 0.0, postiže u slučaju da Bob ima nula pogrešnih bitova (dobrota mu iznosi 0.0), dok Eve mora imati točno pola pogrešnih bitova (dobrota joj iznosi 0.5). Rezultati pokazuju da je Bob u svakom od eksperimenata uspješno dekriptirao sve poruke, dok je Eve zadržana na približno 50% pogođenih bitova poruka što je i bio cilj. Graf 5.7 prikazuje napredak evolucije kroz generacije gdje je prikazna dobrota najbolje Alice iz svake generacije za sva tri eksperimenta (brojevi u dnu slike odgovaraju rednim brojevima eksperimenata).

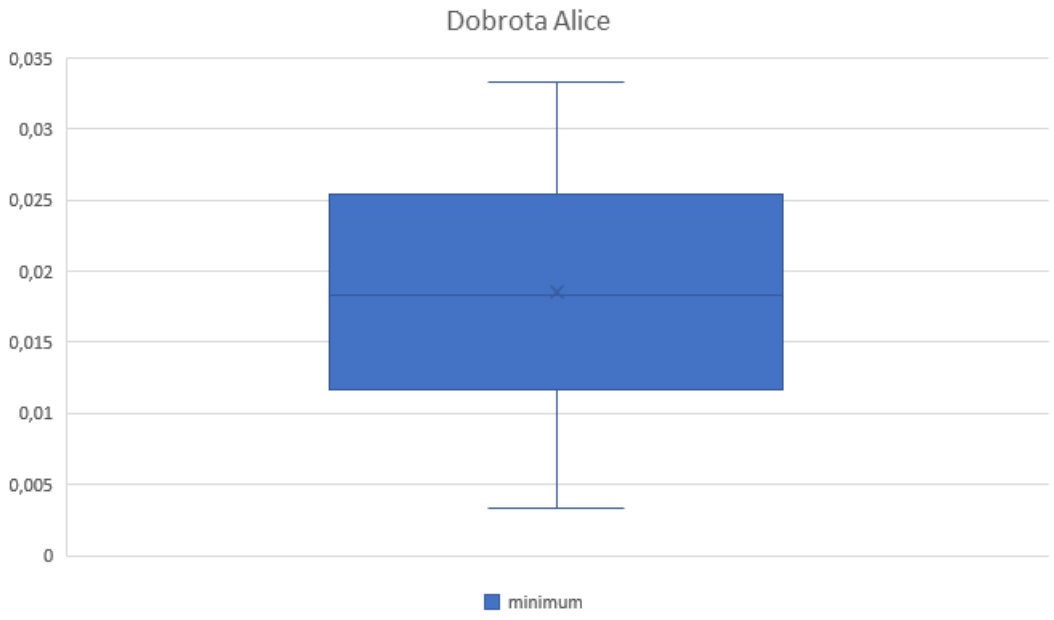
Eksperimenti	Alice	Bob	Eve
1.	0.017	0.0	0.475
2.	0.024	0.0	0.465
3.	0.017	0.0	0.475

Tablica 5.7: Rezultati eksperimenta s Alice, Bob i Eve



Slika 5.7: Dobrota Alice kroz generacije

Dodatno, prvi eksperiment je izveden deset puta i rezultati dobrote najbolje Alice zabilježeni po završetku evolucije prikazani su na slici 5.8. Treba još spomenuti da je u svih deset izvođenja Bob koji pripada najboljoj Alice uspješno dekriptirao sve poruke (njegov doprinos dobroti Alice iznosi nula).



Slika 5.8: Minimalna dobrota Alice

5.3.1. Eksperiment s provjerom kriptografskih kriterija

Sljedeći eksperiment je uključivao i ocjenu kriptografskih kriterija iz poglavlja 4.6. za svaku Alice. Dobrota Alice određena je izrazom (5.1). Konstante A_i odabrane su tako da svaki od doprinosa ima jednak udio u ukupnoj dobroti Alice. Vrijednosti navedenih doprinosa skalirane su na interval $[0, 1]$, kao i konačna dobrota Alice, gdje najbolja vrijednost iznosi nula. Korišteni parametri prikazani su u tablici 5.8, a dobiveni rezultati zabilježeni po završetku jednog pokretanja eksperimenta u tablici 5.9. Iz rezultata je vidljivo da je algoritam kriptiranja uspio zadovoljiti sve kriptografske kriterije. Uz to Bob je potpuno dekriptirao sve poruke, dok je Eve pogodila približno 50% bitova poruka.

$$dobrota\_Alice = A1 * doprinos\_Boba + A2 * doprinos\_Eve + A3 * doprinos\_bijekcije + A4 * doprinos\_kljuca + A5 * doprinos\_slicnosti \tag{5.1}$$

	Alice	Bob	Eve
Veličina populacije	30	15	15
Broj generacija	60	10	10
Graf	4x4	4x4	8x8
Doseg priključaka	1	1	1
Veličina ključa	2 byte		
Broj poruka	30		
Veličina poruke	3 byte		

Tablica 5.8: Korišteni parametri u eksperimentu s provjerom kriptografskih kriterija

Alice	Minimalna dobrota	Maksimalna dobrota	Prosjek dobrote	Dobrota pripadajućeg najboljeg Boba	Dobrota pripadajuće najbolje Eve
	0.167	0.472	0.127	0.000	0.458
Iznosi doprinosa dobrote	Doprinos Boba	Doprinos Eve	Doprinos bijekcije	Doprinos ključa	Doprinos sličnosti
	0.000	0.083	0.00	0.000	0.000

Tablica 5.9: Rezultati eksperimenta s provjerom kriptografskih kriterija

6. Zaključak

Prevodeći klasičnu situaciju u komunikaciji s tri glavne uloge Alice, Bob i Eve u problem optimizacije dobila se mogućnost ocjene kriptografskog algoritma što je otvorilo put ka rješavanju ovog problema pomoću optimizacijskih algoritama kao što je CGP. CGP se za ovaj problem pokazao vrlo prikladnim zbog toga što računalni program dobiven pomoću CGP može biti primjenjen na ulazni tok bitova, gdje nema nikakvih ograničenja po pitanju veličine ulaznih poruka.

Kroz glavna tri eksperimenta pokazalo se da su sve tri uloge uspjele ostvariti svoje ciljeve gdje je bio zadatak uspostava komunikacije između Alice i Boba, dekriptiranje šifrata bez posjeda ključa što je bio zadatak od Eve, te u eksperimentu koji je uključivao sve tri uloge, uspostava komunikacije između Alice i Boba, a da pri tome Eve nije u mogućnosti otkriti poruke koje razmjenjuju. Uz to provedena je i provjera kriptografskih svojstava algoritma što se također pokazalo uspješno, pa tako se potencijalno otvara mogućnost primjene u praksi.

LITERATURA

CARNet CERT. Aes algoritam, 2003a. URL https://www.cis.hr/www.edicija/LinkedDocuments/CCERT-PUBDOC-2003-08-37.pdf.

CARNet CERT. Des algoritam, 2003b. URL https://www.cis.hr/www.edicija/LinkedDocuments/CCERT-PUBDOC-2003-06-24.pdf.

CARNet CERT. Idea algoritam, 2003c. URL https://www.cis.hr/www.edicija/LinkedDocuments/CCERT-PUBDOC-2003-06-25.pdf.

Andrej Dujella i Marcel Maretić. *Kriptografija*. Element, 2007. URL https://web.math.pmf.unizg.hr/~duje/kript.html.

Julian F. Miller. *Cartesian Genetic Programming*. University of York, UK, 2014. URL http://ppsn2014.ijs.si/files/slides/ppsn2014-tutorial3-miller.pdf.

Marko Čupić. *Prirodom inspirirani optimizacijski algoritmi*. 2013. URL http://java.zemris.fer.hr/nastava/pioa/knjiga-0.1.2013-12-30.pdf.

Kriptiranje komunikacije koristeći suparničko učenje evolucijskih algoritama

Sažetak

Kriptografija se bavi problemima kriptiranja i dekriptiranja informacija prilikom njihovog prijenosa nesigurnim komunikacijskim kanalom od pošiljatelja ka primatelju, kada bivaju izložene potencijalnim napadima. U ovom radu pokušalo se pronaći kriptografski algoritam pomoću optimizacije, konkretno pomoću Kartezijskog genetskog programiranja (eng. Cartesian genetic programming, CGP) koji pripada evolucijskim algoritmima. Klasična situacija u komunikaciji koja sadrži tri glavne uloge, a to su Alice ili pošiljatelj, Bob ili primatelj i Eve ili prisluškivač, prevedena je u problem optimizacije, gdje za sva rješenja postoji način određivanja njihove ocjene, tj. dobrote. Time je ostvarena mogućnost pronalaska rješenja pomoću CGP-a. CGP interno sadrži graf čvorova koji u sebi sadržavaju određene funkcije pomoću kojih vrše izmjene nad ulaznim tokom bitova. Time je ostvaren način kriptiranja, odnosno dekriptiranja. Ocjena rješenja izvedena je pomoću suparničkog učenja gdje je Alice imala zadatak kriptiranja, a Bob i Eve dekriptiranja poruka, s tim da u tom procesu Alice i Bob koriste isti ključ kojem Eve nema pristom, što ovaj algoritam čini simetričnim kriptografskim algoritmom. U takvom uređaju bolja dobrota Boba povlači bolju dobrotu Alice, a nasuprot toga, bolja dobrota Eve povlači lošiju dobrotu Alice, što upravo predstavlja glavnu karakteristiku suparničkog učenja. Kroz evoluciju CGP-a pomoću takvog način ocjenjivanja, Alice i Bob su uspjeli pronaći način komunikacije, a da pri tome Eve nije saznala koje informacije razmjenjuju, što je pokazano kroz eksperimente provedene u ovom radu.

Ključne riječi: kriptografija, optimizacija, kartezijsko genetsko programiranje, simetrični kriptografski algoritam, kriptiranje, dekriptiranje, suparničko učenje

Encryption of Communication Using Evolutionary Algorithms with Adversarial Learning

Abstract

Cryptography is concerned with encryption and decryption of information when transmitting them over an insecure communication channel from sender to recipient, when they are being exposed to potential attacks. This paper attempted to find a cryptographic algorithm using optimization, specifically by using Cartesian Genetic Programming (CGP) that belongs to evolutionary algorithms. A classic communication situation that has three main roles - Alice or the sender, Bob or the recipient and Eve or the eavesdropper, has been translated into optimization problem, where for all solutions there is a way of determining their ratings, ie fitness. This enabled the possibility of finding a solution by using CGP. The CGP internally contains the graph of nodes that contain certain functions which can change the input stream of bits. This is how encryption and decryption is done. The evaluation of the solution was done by using the adversarial learning where Alice had the task of encryption, and Bob and Eve had to decrypt the message, where Alice and Bob used the same key that Eve does not have access to, which makes this algorithm a symmetric cryptographic algorithm. In such a situation, Bob's better fitness entails Alice's better fitness, and, conversely, Eve's better fitness entails Alice's better fitness, which is precisely the main characteristic of adversarial learning. Through the evolution of CGP and by using such a method of evaluation, Alice and Bob has been able to find a way of communication such that Eve did not know what information was exchanged, as demonstrated through the experiments conducted in this paper.

Keywords: cryptography, optimization, cartesian genetic programming, symmetric cryptographic algorithm, encryption, decryption, adversarial learning