SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5395

Kriptiranje komunikacije uz pomoć evolucijskih algoritama

Dorotea Protrka

Zagreb, lipanj 2018.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Zahvaljujem mentoru, Domagoju Jakoboviću, za sve dobre savjete i odlično vodstvo u trenutcima kada sam bila izgubljena među idejama. Zahvaljujem i svojoj obitelji koja me kroz godine podržavala i vjerovala u mene. Hvala i Ivanu na inspirativnim programerskim vještinama.

SADRŽAJ

1. Uvod

Računarska znanost danas je široko područje. Postoji mnogo različitih grana kojima se računarska znanost bavi, a koje se međusobno isprepliću. Cilj je ovog rada pokušati povezati kriptografiju i genetsko programiranje kao dvije takve grane, naizgled nepovezane.

Kriptografija je znanstvena disciplina bazirana na teoriji brojeva koja promatra načine i algoritme pretvorbe teksta u šifriran tekst kojeg samo strana kojoj je tekst upućen može pročitati s razumijevanjem. Njezina je vrijednost ogromna u računarskoj sigurnosti. Zbog korištenja komunikacijskih mreža kao osnovnog načina komuniciranja, velika je potreba i za kvalitetnom zaštitom te komunikacije. O sigurnosti komunikacije ovise, ne samo razne tvrtke koje tu komunikaciju koriste u svom poslovanju, nego i privatnost svake osobe.

S druge strane, genetsko programiranje je algoritam za rješavanje optimizacijskih problema. Problema kojima, u zadanom prostoru pretraživanja, treba pronaći optimalno ili dovoljno dobro rješenje.

U ovom radu pokušala sam povezati to dvoje. Uz pomoć uobičajenih kriptografskih primitiva[1], za inspiraciju kojih su poslužili neki standardni algoritmi simetrične kriptografije, genetskim programiranjem gradila sam stabla koja su predstavljala algoritme kriptiranja. Ciljeva je više u kriptografiji. Zbog toga je funkcija dobrote[2] rješenja linearna kombinacije tih ciljeva (zahtjeva). Postoje neka osnovna pravila koja se trebaju poštovati: treba postojati funkcija bijekcije između skupa kojeg čine otvoreni tekstovi i skupa kojeg čine šifrati. Također, osnovni cilj je da se poruka zaštiti od napada. Zbog toga je provedena evolucija s funkcijom dobrote kojoj je jedna od varijabli neuspjeh napadača, odnosno uspjeh krajnjeg člana komunikacije da dekriptira šifrat. Zbog različitih zahtjeva kriptografije, rad istražuje koliko dobra rješenja pojedine kombinacije primitiva i funkcija dobrote mogu dati.

Sljedeće poglavlje pruža teorijski pristup genetskom programiranju i kriptografiji i

[1] Funkcije koje rade osnovne operacije nad bitovima u sklopu kriptografije.

[2] Više u poglavlju 2.1.2.

objašnjava pojmove potrebne za razumijevanje daljnjeg rada. Zatim slijedi poglavlje opisane pojmove povezuje s ovim radom i daje uvid u njegovu strukturu.
Zatim slijedi ključno poglavlje u kojem su vidljivi rezultati rada. Također su objašnjeni parametri koji su bili mijenjani u svrhu dobivanja različitih rezultata. Cilj tih promjena bila je potraga za što boljom kombinacijom primitiva i zahtjeva koji su dio linearne kombinacije funkcije dobrote, tj. što boljim rješenjem. Također se istražuje ovisi li kvaliteta rješenja koja se kreiraju o duljini teksta ili vjerojatnosti odabira pojedinih primitiva.
Na kraju, u zadnjem poglavlju je zaključak o radu i rezultatima te mogućnostima koje genetsko programiranje ima u sklopu kriptografije.

2. Genetsko programiranje i kriptografija

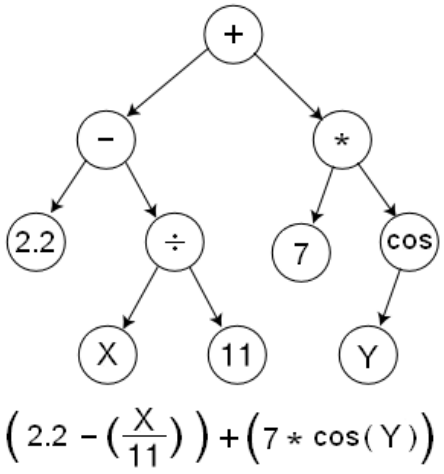
2.1. Genetsko programiranje

Razni se problemi mogu prikazati i riješiti uz pomoć računala. Skup problema koje zovemo optimizacijski problemi za svako rješenje definiraju ocjenu tog rješenja. Optimalna rješenja imaju veću ocjenu (maksimizacija) ili manju (minimizacija) od ostalih. Postoje razni optimizacijski algoritmi koji pretražuju prostor rješenja s ciljem pronalaska rješenja sa što boljom ocjenom rješenja. Podvrsta optimizacijskih algoritama su i evolucijski algoritmi.
Evolucijski algoritmi su algoritmi koji su inspiraciju pronašli u biološkoj evoluciji i darvinističkoj filozofiji. Takav je pristup moguć zbog činjenice da rješenja problema imaju svoju ocjenu kvalitete. Zato možemo birati bolja rješenja (selekcija), kombinirati više dobrih rješenja (križanje) ili jednostavno pokušavati slučajnim promjenama dobiti nove i bolje osobine rješenja (mutacija).
Genetsko programiranje spada u evolucijske algoritme.

2.1.1. Struktura riješenja

Kao i kod svih optimizacijskih problema za početak je važno odabrati strukturu rješenja. Česta struktura kod genetskog programiranja je stablo. Stablo je nelinarna struktura podataka koja se sastoji od čvorova međusobno povezanih u hijerarhiju kao što je prikazano na slici 2.1. Stablo prikazujemo kao aciklički povezani graf (Odjel za matematiku). Početni čvor zvan korijen stabla ima oznaku razine 0 i može se granati u nove čvorove. Svaki čvor ima svoju vrijednost, oznaku razine za jedan veću od roditeljskog čvora i pokazuje na čvorove djecu ukoliko ih ima. Ako se čvor ne grana zove se list (engl. *leaf*), a ako se grana zove se unutarnji čvor (engl. *node*). Razlikujemo uloge unutarnjih čvorova i listova. Unutarnji čvorovi služe kao operatori (engl. *function set*

F) dok su listovi operandi (engl. *terminal set T*).



Slika 2.1: Primjer stabla

Važni pojmovi su još i:

Dubina stabla - maksimalan broj razina stabla.

Stupanj čvora - maksimalan broj djece čvorova među svim roditeljskim čvorovima stabla.

Za svaki problem postoje predefinirani operatori i operandi.
Stablo je dakle definirano hijerarhijom čvorova, ali i načinom obilaska. Tri su načina posjećivanja čvorova:

Preorder - oblilazak u kojem se prvo posjećuje trenutni čvor, a zatim rekurzivno posjećuju podstabla djece čvorova u zadanom redoslijedu.

Inorder - obilazak u slučaju da je stupanj stabla 2; prvo se rekurzivno obilazi prvi čvor dijete pa trenutni čvor i zatim drugi čvor dijete.

Postorder - obilazak u kojem se prvo posjećuju podstabla djece čvorova, a potom trenutni čvor.

2.1.2. Funkcija dobrote

Spomenuto je već da svako rješenje ima svoju ocjenu u kontekstu zadanog problema. Tu ocjenu definira funkcija dobrote. Funkcija dobrote govori koliko je dano rješenje

blizu optimalnog u postojećem prostoru pretraživanja rješenja. Ocjenu pojedinog rješenja nazivamo dobrota jedinke. Ona je važna komponenta zbog darvinističkog pristupa problemu. Bolje jedinke, tj. rješenja dodaju se u populaciju rješenja s kojima algoritam radi da bi se u konačnici pronašlo dovoljno dobro ili najbolje rješenje.

2.1.3. Populacija

Genetsko programiranje je algoritam koji radi u iteracijama. Svaku iteraciju definira njezina populacija. U početnoj iteraciji generira se početna populacija. Postoje različiti načini da se stvore početne jedinke. Dvije su osnovne metode najčešće korištene: metoda *full* i metoda *grow*.
Početak obje metode je jednak: iz skupa unutarnjih čvorova s uniformnom se vjerojatnošću izabire jedan primitiv koji se postavlja za početni, korijenski čvor. Nakon toga se metode razdvajaju. *Full* metoda nadalje bira za djecu-čvorove nove primitive iz skupa unutarnjih čvorova. Metoda *grow* bira za djecu-čvorove nove primitive iz unije skupova unutarnjih čvorova i listova. Kada je dubina stabla za jedan manja od dopuštene dubine, u obje se metode biraju primitivi iz skupa terminalnih čvorova.
Naravno, postoje još neke metode koje mogu kombinirati ove dvije, a jedna od takvih je i *ramped-half-and-half* sa specificiranom dubinom stabla korištena u ovom radu. U svim ostalim populacijama, populacija iteracije je rezultat modificiranja jedinki populacije prethodne iteracije. Takve modifikacije u slučaju genetskih algoritama zovemo genetski operatori.

2.1.4. Genetski operatori

Genetski operatori definiraju način izmjene postojećih rješenja u svrhu dobivanja još boljih. Razlikujemo osnovne operatore: reprodukcija, mutacija i križanje (Donđivić i Kokan). Reprodukcija je kopiranje neke jedinke i umetanje u novu populaciju. U takvom pristupu nema promjene, ali se mogu sačuvati u potpunosti dobra rješenja. Algoritmi koji određeni broj najboljih rješenja jedne populacije reproduciraju i stavljaju u novu populaciju zovu se elitistički algoritmi.
Mutacija je operator modifikacije jedne odabrane jedinke. U slučaju kada je struktura prikaza rješenja stablo, moguće je mutaciju definirati različitim postupcima. Najčešće se odabere postablo koje se želi mutirati. Zatim se ono može obrisati, tj. zamijeniti jednim terminalnim čvorom, zamijeniti čitavim novogeneriranim podstablom ili zamijeniti postablom neke druge jedinke iz populacije. Također je moguće kombinirati ove postupke. Nakon mutacije nastaje jedna nova jedinka.

Križanje je operator koji rezultira jednom ili više novih jedinki. U križanje ulaze najčešće dva roditeljska stabla selektivno odabrana među jedinkama trenutne populacije. Križanje se odvija odabirom podstabala u jednom i drugom roditelju i njihovom zamjenom.

2.1.5. Selekcija

Selekcija je postupak uz pomoć kojeg odlučujemo koje jedinke će sačinjavati novu populaciju rješenja. Selekcijom se također odabiru roditelji za križanje i stabla koja će se mutirati ili reproducirati. Selekcija najčešće daje vjerojatnost odabira jedinke proporcionalnu dobroti jedinke. Ipak, selekciju najčešće ne radimo nad čitavom populacijom zbog jednostavnog problema: lokalnog optimuma. Kada se sva rješenja koncentriraju u lokalnom optimumu, jako je teško nastaviti evoluciju prema globalnom optimumu problema.
Neke od selekcija su: proporcionalna selekcija, turnirska selekcija, selekcija odsjecanjem, Boltzmannova selekcija (Čupić, 2013). Odabir selekcije ima velik utjecaj na brzinu kojom će algoritam doći do dobrih rješenja, ali i vjerojatnost zaglavljivanja u lokalnom optimumu.

2.1.6. Algoritam genetskog programiranja

Bez obzira na problem koji se rješava genetskim programiranjem, algoritam je, u suštini, uvijek isti. U početku se inicijalizira početna populacija. Svakoj se jedinki dodijeli dobrota. Algoritam je iterativan i kroz svaku iteraciju ponavlja se isti postupak. Cilj je svake iteracije stvoriti novu i bolju populaciju. Dok ima potrebe za stvaranjem novih jedinki, algoritam odabire genetski operator i potrebne jedinke. Zatim kreira novu jedinku i odlučuje hoće li ju staviti u novu populaciju. Ponekad u novu populaciju može staviti jedinke prethodne populacije. Nakon što je kreirana nova populacija, svakoj se jedinki uz pomoć funkcije dobrote dodjeljuje ocjena. Ukoliko nije ispunjen uvjet za kraj, slijedi nova iteracija.
Uvjeti za kraj mogu biti: broj iteracija, pronalazak dovoljno dobrog riješenja, nemogućnost pronalaska boljeg rješenja (ako je algoritam zapeo u lokalnom optimumu) i slično. U nastavku je prikazana iteracija ovog algoritma.

```
novaPopulacija.dodaj(staraPopulacija.najbolji());
while novaPopulacija.veličina < veličinaPopulacije do
    novaAkcija = odaberiSlučajnuAkciju();
    if novaAkcija == REPRODUKCIJA then
        novaJedinka = turnirskaSelekcija(staraPopulacija);
        novaPopulacija.dodaj(novaJedinka);
    else if novaAkcija == MUTACIJA then
        jedinka = turnirskaSelekcija(staraPopulacija);
        mutiranaJedinka = jedinka.mutiraj();
        novaPopulacija.dodaj(mutiranaJedinka);
    else if novaAkcija == KRIŽANJE then
        roditelj1 = turnirskaSelekcija(staraPopulacija);
        roditelj2 = turnirskaSelekcija(staraPopulacija);
        novaJedinka = križaj(roditelj1, roditelj2);
        novaPopulacija.dodaj(novaJedinka);
end
```

Algorithm 1: Iteracija algoritma genetskog programiranja

2.2. Kriptografija

Kriptografija je znanost koja proučava načine slanja poruke u obliku u kojem je poruka čitljiva samo sudioniku kojem je i namijenjena. Tajnost poruka odavno je tražena osobina pa se tako kriptografija razvijala kroz stoljeća. U današnje vrijeme ona je fokusirana na algoritme koji omogućuju prolazak poruka kroz računalne i komunikacijske mreže bez napada trećih osoba.

Pošiljalac poruke u literaturi (a i u ovom radu) naziva se *Alice*. Poruka predstavlja niz bitova i naziva se otvoreni tekst (engl. *plaintext*). Kada se poruka šifrira uz pomoć ključa, dobivamo šifrat (engl. *ciphertext*) ili kriptogram (Dujella). Pošiljalac šalje poruku komunikacijskim kanalom. Napadač koji *prisluškuje* kanal ne smije moći otkriti poruku. Napadača zovemo *Eve* ili *Intruder* (nadalje *Trudy*). Kada poruka dođe do primaoca (u literaturi i nadalje *Bob*), on ju uz pomoć ključa može dešifrirati i tako ponovo dobiti otvoreni tekst.

Kriptografija ima dva osnovna smjera s obzirom na vrstu ključa koji se koristi pri šifriranju i dešifriranju. Simetrična kriptografija je ona u kojoj se koristi isti ključ za šifriranje i dešifriranje.

Asimetrični kriptosustavi imaju različite ključeve za ta dva postupka. Svaki sudionik

komunikacije u tom slučaju posjeduje dva ključa. Jedan je javno poznat i služi za kriptiranje poruka. Drugi je tajan i svrha mu je da jedino sudionik (vlasnik ključeva) kojem je poruka namijenjena može dešifrirati šifrat.
U ovom radu implementirana je simetrična kriptografija.

2.2.1. Simetrična kriptografija

Simetrična kriptografija zove se još i kriptografija tajnog ključa. Njena osnovna značajka je da u komunikaciji postoji samo jedan tajni ključ koji služi i za šifriranje i za dešifriranje. Osim prijenosa šifriranih poruka, simetrična se kriptografija koristi i za autentifikacije poruka (engl. *message authentification codes*).

Simetrične kriptografske algoritme možemo podijeliti na (Blažević):

1. Algoritme koji kriptiraju blokove podataka
2. Algoritme koji kriptiraju tokove podataka
3. Kodovi autentifikacije poruke

Neki od najpoznatijih simetričnih algoritama za šifriranje su DES, IDEA i AES.

Data Encryption Standard (DES)

DES je algoritam koji je imao jako veliku ulogu mnogo godina. Ipak, s razvojem tehnologije postao je preslab i moguće ga je uspješno napasti. Otvoreni tekst se generira u 64-bitnim blokovima. Šifrat je također 64-bitan. Ključ za šifriranje je 56-bitan broj. Prvi korak je inicijalna permutacija bitova. Slijedi 16 koraka u kojima se svaki put niz bitova podijeli na lijevu i desnu polovicu. Desna polovica postaje lijeva polovica sljedećeg koraka dok se nad lijevom polovicom izvršavaju operacije u ovisnosti o 48-bitnim podključevima izvedenim iz originalnog ključa. Posljednji korak predstavlja inverziju inicijalne permutacije. Da bi se omogućila veća sigurnost ovog kriptiranja nastale su i inačice ovog algoritma. Najpoznatije među njima su 3DES, DESX te inačice koje koriste modificiranje S-blokova (CARNet CERT, 2003b).

International Data Encryption Algorithm (IDEA)

IDEA je kriptografski algoritam vrlo otporan na napade. Smatra se jednim od najsigurnijih algoritama simetrične kriptografije. Patentiran je pa je za njegovu upotrebu

potrebna licenca. Neke od osnovnih operacija koje algoritam koristi su XOR, zbrajanje modulo 216 i multipliciranje modulo 217. Sam algoritam se sastoji od osam sličnih koraka. Iterativno se koriste 16-bitni blokovi generirani u prethonoj iteraciji nad kojima se obavljaju operacije. Kao i prethodno, od ključa se dobiju podključevi koji se u svakoj iteraciji mijenjaju uz pomoć operacije rotacije (CARNet CERT, 2003c).

Advanced Encryption Standard (AES)

AES je algoritam koji je nastao kao zamijena za DES, a temelji se na Rijndael algoritmu. Osnovna struktura algoritma je dvodimenzionalan niz okteta, tj. matrica stanja. Nad tom se matricom provode operacije algoritma. Otvoreni tekst se u početku postavlja u matricu stanja i nad njim se izvršavaju određene transformacije. Te se transformacije rade na temelju supstitucijske tablice, ključa i podključeva te same arhitekture prikaza blokova okteta (posmak redaka tablice, miješanje podataka unutar istog stupca), (CARNet CERT, 2003a).

3. Kriptiranje komunikacije uz pomoć evolucijskih algoritama

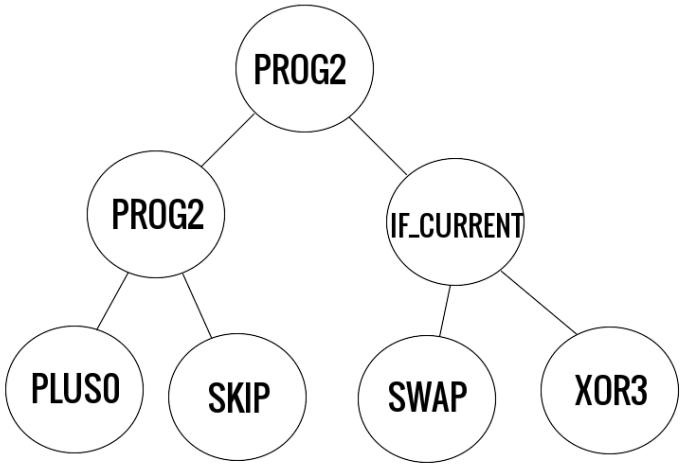
3.1. Uvod u rad

Ovaj rad promatra ima li mjesta za genetsko programiranje u kriptografiji. Već se pokazalo da genetskim programiranjem možemo riješiti različite probleme vrlo efikasno. Pitanje je: koliko dobar algoritam kriptiranja može osmisliti? Prvi je korak rada bilo osmišljavanje primitiva, tj. manjih algoritama koji će kombinirani dati konkretan algoritam. Ti primitivi grade stablo koje predstavlja algoritam za kriptiranje.
Stabla koja predstavljaju riješenja, tj. algoritme, čine populaciju koju evaluiramo, mijenjamo i od koje nastaju nove populacije kroz iteracije algoritma genetskog programiranja. Ona kroz generacije teže biti sve bolja. Definiran je prolazak kroz stablo i način na koji on utječe na tekst koji se kriptira, odnosno dekriptira.
Važan problem predstavlja i određivanje funkcije dobrote svakom algoritmu. Kada su definirane ove komponente, genetsko programiranje započinje pretragu.

3.2. Stabla kao algoritmi kriptiranja

Otvoreni tekst za kriptiranje i kriptogram su nizovi bitova. Nad njima se izvršava kriptiranje, odnosno dekriptiranje. Ta dva procesa su zapravo jednostavni obilasci stabla. U implementaciji se koristi već spomenut *preorder* obilazak stabla.
Niz bitova nad kojim se radi ima svoju duljinu tj. broj bitova. Bit kojeg trenutno promatramo je onaj nad kojim se izvode operacije (ili oktet unutar kojeg je taj bit). U početku obilaska, promatrani bit je prvi bit. Za svaki posjećen terminalni čvor, promatrani bit postaje sljedeći bit. Ukoliko je trenutni bit posljednji u nizu, trenutni bit ponovo postaje prvi bit. Taj proces se događa za vrijeme obilaska cijelog stabla. Kada se stablo obiđe, od otvorenog teksta dobiven je kriptogram, odnosno od kriptograma

otvoreni tekst. Slika 3.1 prikazuje kako izgleda jedno jednostavno stablo. Obilazak se izvodi sljedećim redoslijedom: *PROG2*, *PROG2*, *PLUS0*, *SKIP*, *IF_CURRENT* i zatim ili *SWAP* ili *XOR3*. Konkretnu izvedbu prikazuje slika 3.2. Istaknuti bit predstavlja trenutno promatrani bit.



Slika 3.1: Kriptiranje prevedeno u stablo

3.3. Primitivi

Primitivi su čvorovi uz pomoć kojih se gradi stablo koje predstavlja rješenje problema. U ovom slučaju oni predstavljaju dijelove algoritma za kriptiranje, tj. manje algoritme koje sadrži većina standardnih algoritama i kombinira. Za izgradnju stabla korištene su dvije liste čvorova: unutarnji (funckijski) čvorovi i listovi (terminalni čvorovi).

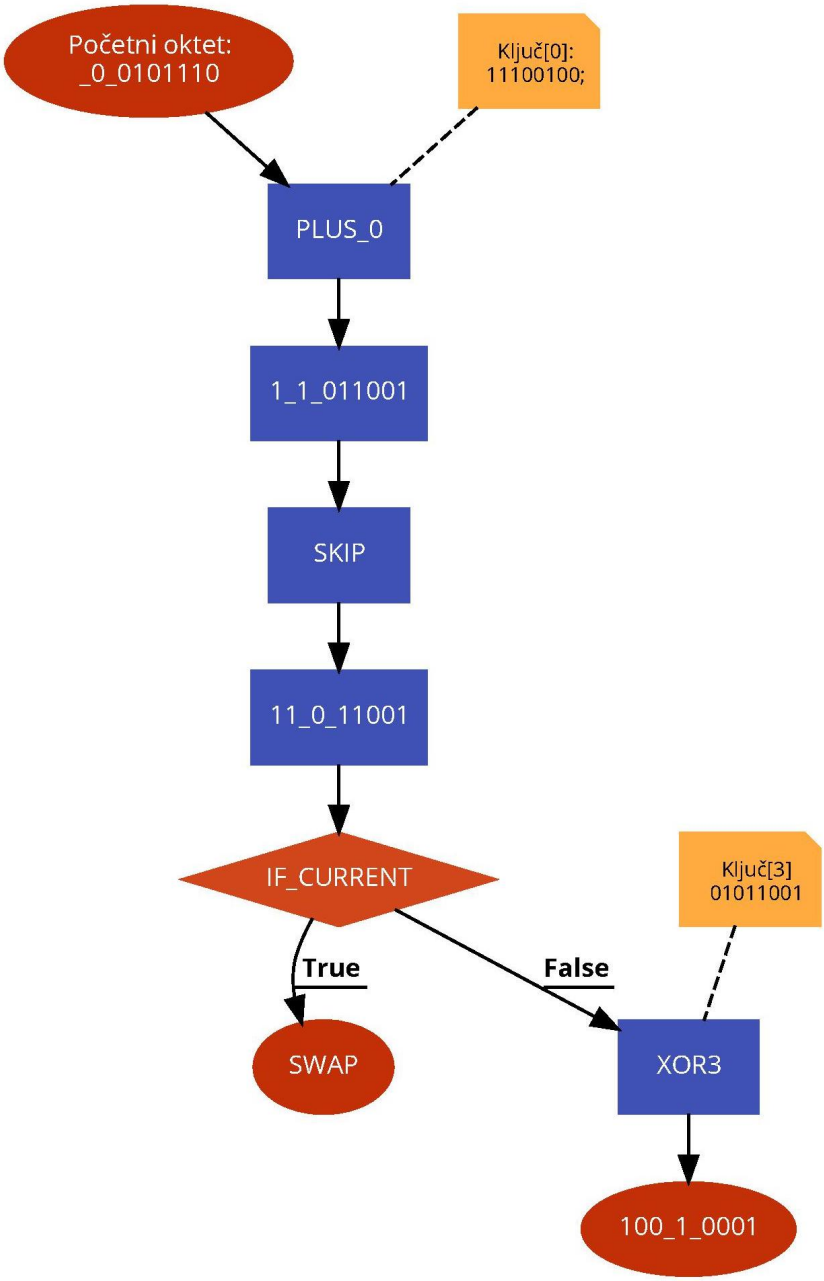
Unutarnji čvorovi su:

PROG2 - obično grananje stupnja 2 u kojem se obilaze svi čvorovi.

IF_CURRENT - grananje stupnja 2 u kojem se, ukoliko je trenutno promatrani bit 1 prolazi lijevo podstablo, a inače desno.

IF_LAST - grananje stupnja 2 u kojem se, ukoliko je prethodno promatrani bit 1 prolazi lijevo podstablo, a inače desno.

IF_NEXT - grananje stupnja 2 u kojem se, ukoliko je slijedeći promatrani bit 1 prolazi lijevo podstablo, a inače desno.



Slika 3.2: Primjer izvedbe algoritma

Terminalni čvorovi su:

XOR_B - označava XOR operaciju okteta ključa sa oktetom teksta kojem pripada promatrani bit. Oktet ključa bira se na osnovu broja B koji predstavlja redni broj okteta u ključu. S obzirom da ključ može biti maksimalno 64 bita dug, broj B poprima vrijednosti od 1 do 8. U slučaju da je ključ manje veličine od traženog okteta, radi se operacija modulo sa veličinom ključa u oktetima.

PLUS_B - primitiv predstavlja zbrajanje zadanog okteta ključa s oktetom teksta u kojem se nalazi promatrani bit. Kao i u prethodnom slučaju, postoji 8 različitih verzija ovog primitiva.

SWAP - primitiv kod prvog pojavljivanja označava trenutni bit kao "spreman za zamijenu", a kada se pojavi drugi put mijenja trenutni bit sa prethodno označenim za zamijenu.

SKIP - preskače se trenutno promatrani bit i trenutno promatrani bit postaje sljedeći.

XOR - nad trenutno promatranim bitom radi se operacija XOR-a sa 1 (negacija).

3.4. Podjela uloga

Kao i u standardnoj situaciji, u ovom su radu implementirane tri uloge ključne u kriptiranoj komunikaciji. Prva uloga je enkripcija. *Alice* ili pošiljatelj poruke je sudionik komunikacije koji otvoreni tekst šifrira uz pomoć ključa i algoritma simetrične kriptografije. Tako dobiven kriptogram šalje komunikacijskim kanalom.
Poruku koja putuje komunikacijskim kanalom dobit će sljedeća dva sudionika: napadač (*Trudy*) i podrazumijevani primatelj poruke (*Bob*). Oba sudionika žele što točnije dekriptirati kriptogram. Razlika je u tome što napadač nema ključ komunikacije, a *Bob* ima.
Svi sudionici komunikacije teže razviti algoritme enkriptiranja/dekriptiranja sa svrhom što boljeg obavljanja uloge koja ih opisuje. Svatko od njih ima svoju populaciju algoritama (stabala, rješenja) nad kojima izvršavaju evolucijski postupak.

3.5. Evaluacija stabla i suparničko učenje

U procesu evolucije, operator selekcije odabire bolja rješenja i od njih se generiraju stabla nove populacije. Da bi takav odabir bio moguć, rješenja moraju biti evaluirana, tj trebaju na temelju nekog kriterija dobiti ocjenu za ono što im je svrha. S obzirom na ulogu, u ovom radu postoje dvije vrste evaluacija: evaluacija enkriptiranja i evaluacija dekriptiranja.

Da bi objasnili evaluaciju stabala, bitno je razumjeti pojam suparničkog učenja (engl. *adversarial learning*). Suparničko učenje predstavlja istovremeno treniranje više modela čiji pozitivni rast dobrote povlači lošiju dobrotu drugih. U ovakvom treniranju modeli skupa evoluiraju, odnosno ni jedan nije značajno bolji od drugog. Ipak, kroz prilagođavanje jednog drugom postaju globalno bolji. Takva vrsta učenja primjenjena je i u ovom radu.

Evaluacija dekripcije je jednostavna. Za svaki algoritam kriptiranja kojeg *Alice* proizvede, *Bob* i *Trudy* dobiju skup podataka za učenje. Broj podataka za učenje je postavljen na 100. Svaki podatak se sastoji od tri dijela: nasumično odabran otvoreni tekst, šifrat tog teksta i ključ korišten za šifriranje. Naravno, ključ kao podatak koristi samo *Bob* jer je napadaču nedostupan. Na tim podacima *Bob* i *Trudy* uče dešifrirati i njihova funkcija dobrote proporcionalna je uspješno prevedenim bitovima. Zbog jednostavnije interpretacije standardizirana je na intervalu [0, 1].

Evaluacija enkripcije, s druge strane, je malo kompliciranija. Ona je linearna kombinacija različitih parametara.

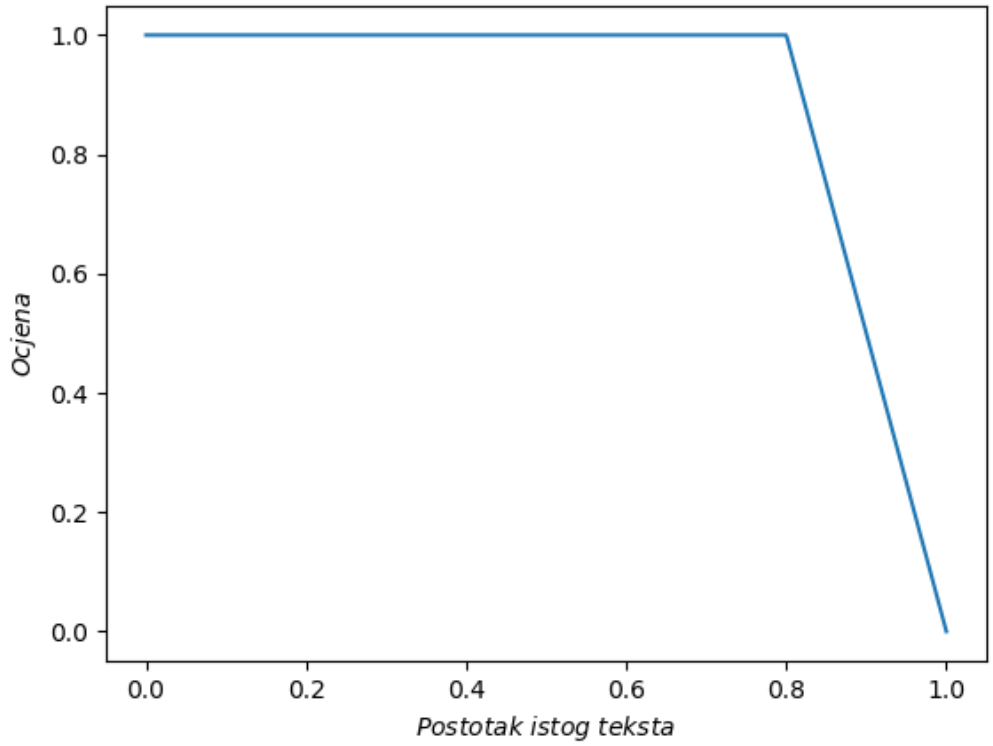
Prvi parametar je u vezi sa samom kriptografijom: ocjenjuje se je li algoritam kojim kriptiramo poruke bijekcija [1]. Bijekcija je važna u kriptografiji jer bi se bez nje kriptirani tekst mogao krivo interpretirati. Ova provjera rezultira ocjenom u obliku postotka. Ocjena je određena izrazom 3.1. Ukupan broj kriptograma ovisi o duljini niza bitova koji se enkriptiraju. On je $2^{\,duljina\ niza}$. Broj generiranih kriptograma je broj različitih kriptograma koje algoritam može generirati.

$$ocjena\ bijekcije = \frac{broj\ generiranih\ kriptograma}{ukupan\ broj\ kriptograma} \tag{3.1}$$

Drugi parametar je iskorištenost ključa. Ukoliko se ne koristi ključ, jednaka je

[1] Funkcija koja iz jednog skupa svaki član skupa preslikava u točno jedan član drugog skupa tako da nikoja dva člana prvog skupa ne preslikava u isti član drugog skupa te su svi članovi drugog skupa slike nekog člana prvog skupa.

šansa za otkrivanje algoritma za *Boba* i za *Trudyja*. Zbog toga se boljima smatraju algoritmi koji koriste što više okteta ključa. Ovaj se parametar računa kao postotak okteta ključa koji su korišteni.
Sljedeći parametar je ocjena provođenja enkripcije uopće. Bitno je da se mijenja dovoljan broj bitova u odnosu na početni, otvoreni tekst. Za svaki bit šifrata provjerava se je li promijenjen. Računa se udio promijenjenih bitova u ukupnom broju bitova. Ocjena je prikazana grafom 3.3 uz mogućnost pomicanja točke u kojoj se događa promjena funkcije. Ukoliko je promijenjeno barem 20% bitova, ocjena je 1.



Slika 3.3: Graf ocjene provođenja enkripcije

Najvažniji dio ocjene rješenja koje ponudi *Alice* je povratna informacija sudionika koji izvode dekripciju. Napadačeva dobrota uzima se u linearnoj kombinaciji s negativnim preznakom jer mogućnost boljih napadača znači loš algoritam kriptiranja. S druge strane, dobrota sudionika koji treba dekriptirati tekst točno uzima se u linearnoj kombinaciji s pozitivnim predznakom. Njihove dobrote predstavljaju postotak ispravno dekriptiranih podataka dobivenih za učenje. U formuli 3.2 napadačeva dobrota je označena kao *Trudy_faktor*, a dobrota legitimnog dekriptora *Bob_faktor*.
Svim varijablama o kojima ovisi dobrota *Alice* pridružena je konstanta uz pomoć koje

se odlučuje koliku važnost im se dodijeljuje. Konačna formula prikazana je formulom 3.2 gdje su A_i konstante.

$$ocjena = A_1 \times faktor\_bijekcije + A_2 \times faktor\_ključa + A_3 \times faktor\_promjene\_bitova + A_4 \times Bob\_faktor - A_5 \times Trudy\_faktor \quad (3.2)$$

Ostali parametri dobrote određeni su u odnosu na ostale sudionike i prikazani su u donjem pseudokodu. pseudokod prikazuje evoluciju sva tri sudionika komunikacije kroz suparničko učenje. Za svaku iteraciju, prvo se ocjenjuju parametri funkcije dobrote sudionika enkriptora (*Alice*). Zatim se ocjenjuje i ostatak parametara uz pomoć kojih se dobije sveukupna ocjena. To se radi kreiranjem podataka koji služe dekriptorima da nauče algoritam dekripcije koja odgovara zadanom algoritmu enkripcije. Nakon što je algoritam naučen, generira se ocjena za dekriptore s obzirom na točnost dekripcije. Te se ocjene zatim koristi da se do kraja ocjeni enkriptor.

```
alice.inicijalizirajEvoluciju();
while trenutnaIteracija < maxBrojIteracija do
    alice.jednaIteracijaEvolucije();
    alice.evaluiraj();
    while alice.postojiSljedeća do
        novaAlice = alice.dohavtiSljedeću();
        skupPodatakaZaUčenje = novaAlice.generirajPodatke();
        bob.evoluiraj(skupPodatakaZaUčenje);
        trudy.evoluiraj(skupPodatakaZaUčenje);
        novaAlice.dobrota(bob.najbolji() - trudy.najbolji() +
          novaAlice.dobrota);
    end
end
```

Algorithm 2: Suparničko učenje

4. Rezultati

4.1. Optimizacija parametara

Rezultati koji su dobiveni ovim radom uvelike su ovisili o manipulaciji parametrima. U ovom poglavlju objasnit ću o kojim je promjenama parametara riječ, a u sljedećem čime su te promjene rezultirale. Početne vrijednosti parametara optimizacije mijenjala sam postupno. Nakon svakog testa, parametre koji su dali najbolje rezultate zadržala sam kao početne parametre sljedećeg testa.
Na samom početku, tekst je bio duljine 2 okteta. U skupove primitiva uključeni su svi dostupni primitivi. Svim primitivima dana je jednaka vjerojatnost odabira u svakom koraku. Konstante uz parametre funkcije dobrote enkriptora su: 3 za dobrote dekriptora, 1 za *faktor_promjene_bitova*, 1 za *faktor_ključa* te 0 za *faktor_bijekcije*. Duljina korištenog ključa također se može mijenjati, a u napravljenim mjerenjima korištena duljina od 64 bita. Veličina populacije je 15 jedinki. Odabir jedinki koje će se reproducirati, mutirati ili biti roditeljske jedinke u križanju radi se uz pomoć 7-turnirske selekcije. Mutacija se izvršava tako da se slučajno odabere podstablo jedinke i zamijeni sa slučajno generiranim podstablom. Križanje zahtjeva odabir dviju roditeljskih jedinki selekcijom. Zatim se jednoj od njih mijenja slučajno odabrano podstablo podstablom druge. Maksimalna dubina stabala je 7.

Prva skupina parametara koji su mijenjani su dostupni primitivi. Već je rečeno da sudionici komunikacije kojima je dostupan ključ imaju neke primitive različite od sudionika kojima ključ nije dostupan. Uz pomoć njih mogu graditi stabla za kriptiranje. Za neke od tih primitiva pokazalo se da ih je jako teško rekonsturirati tj. dekriptirati na drugoj strani komunikacije. Također, poruke enkriptirane uz pomoć pojedinih primitiva ponekad je mnogo jednostavnije dekriptirati dekriptoru s ključem, nego napadaču zbog same činjenice da ima pristup ključu. Cilj je ove optimizacije pronaći idealan skup primitiva dostupan sudionicima.

U drugom dijelu skupina parametara koji su optimizirani za dobivanje boljih rezultata su konstante koje su dio linearne kombinacije funkcije dobrote enkriptora. Ta skupina parametara davala je veći naglasak nekim svojstvima kriptografije u odnosu na druge. Tu spadaju konstante koje u formuli 3.2 stoje uz sljedeće parametre: *faktor_bijekcije*, *faktor_ključa*, *faktor_promjene_bitova*, *Bob_faktor* i *Trudy_faktor*. Optimalni parametri daju najveću razliku između funkcija dobrote dvaju konkurentnih dekriptora.

Parametar koji je mijenjao izgled i kvalitetu stabala-rješenja je i duljina teksta. Različit broj okteta otvorenog teksta još je jedan parametar čiju sam promjenu proučila kroz rezultate.

Posljednji parametar koji je mjenjan je vjerojatnost s kojom stablo za svoj primitiv bira primitiv *SKIP*. Zanimalo me kakav bi utjecaj na algoritme dekriptiranja imala vjerojatnost da se bitovi manje mijenjaju zasebno, a više u skupinama što veća vjerojatnost ovog primitiva i omogućava.

4.2. Pregled rezultata

4.2.1. Rezultati dobiveni mijenjanjem skupova primitiva

U tablicama 4.1, 4.2 prikazani su skupovi primitiva korišteni u ovom testu. Znak "+" označava da se primitiv koristi u sudionikovom skupu primitiva, a znak "-" da se ne koristi. Napadač i legitimi sudionici razgovora imaju različite primitive zbog dostupnosti ključa. Rezultati su prikazani u tablici 4.3, a dobiveni su uz pomoć svih kombinacija dvaju skupova primitiva. Stupac *Skupovi* tablice 4.3 definira koji se skupovi primitiva koriste. Prvi brojevi označavaju skupove primitiva unutarnjih čvorova, a konkretno predstavljaju *Redni broj skupa* tablice 4.1. Drugi označavaju skupove primitiva terminalnih čvorova, tj. stupac *Redni broj skupa* tablice 4.2.

Rezultati su dobiveni evolucijom *Alice* u 200 iteracija. Za svaku od tih iteracija izvodi se evolucija dekriptora u još 1000 iteracija. Svaka iteracija pamti najbolje jedinke. Stupac *Max Alice* označava dobrotu najbolje *Alice* ikad pronađene tokom izvođenja testa (pokretanja programa). Cilj je ovog testa bio odrediti optimalne skupove primitiva uz pomoć kojih će algoritam kriptiranja biti pretežak napadaču za dekriptiranje. Pokazalo se da se taj skup sastoji od operacije *XOR* i *SKIP* za terminalne čvorove

Tablica 4.1: Skupovi korištenih primitiva, unutarnji čvorovi

Redni broj skupa	Sudionici	PROG2	IF_CURRENT	IF_LAST	IF_NEXT
1	*Trudy*	+	+	+	+
	Bob Alice	+	+	+	+
2	*Trudy*	+	+	-	-
	Bob Alice	+	+	-	-
3	*Trudy*	+	-	-	-
	Bob Alice	+	-	-	-

Tablica 4.2: Skupovi korištenih primitiva, terminalni čvorovi

Redni broj skupa	Sudionici	SKIP	SWAP	XOR_B	XOR	PLUS_B
1	*Trudy*	+	+	-	+	-
	Bob Alice	+	+	+	-	+
2	*Trudy*	+	+	-	+	-
	Bob Alice	+	+	+	-	-
3	*Trudy*	+	+	-	+	-
	Bob Alice	+	-	+	-	-

i jednostavnog grananja (*PROG2*) za unutarnje. Smatram da ovo nije neočekivano. Zbog korištenja ključa na većem broju bitova odjednom, napadaču nije jednostavno rekreirati *XOR* funkciju. Za daljnje testove izabrat ćemo skupinu primitiva unutranjih čvorova rednog broja 3 te terminalnih čvorova rednog broja 3.

4.2.2. Rezultati dobiveni mijenjanjem funkcije dobrote enkriptora

Parametri koji se mijenjaju u ovom dijelu su faktori koji množe varijable linearne kombinacije funkcije dobrote sudionika enkriptora (*Alice*). U prvom dijelu tablice 4.4 dani su iznosi tih konstanti, a u drugom dijelu dobrote najboljih rješenja. Boljim rješenjima smatramo ona za koje je razlika između dobrote dvaju dekriptora što veća. Iznimno, u ovom je testu duljina teksta postavljena na 1 oktet zbog eksponencijalne ovisnosti funkcije provjere bijekcije o njoj.

Prvi test je napravljen tako da svakoj varijabli da jednaku značajnost u funkciji dobrote, a preostali testovi daju posebnu značajnost pojedinoj komponenti (50%). Takvim pristupom se, između ostalog, promatra utjecaj tih komponenti na rezultat. S obzirom na dobivene rezultate možemo zaključiti da sve definirane varijable doprinose boljim re-

Tablica 4.3: Rezultati različitih skupova primitiva

Skupovi	Max *Alice*	*Bob*	*Trudy*
1-1	0.392	1.000	0.861
1-2	0.596	0.937	0.840
1-3	0.622	1.000	0.885
2-1	0.627	0.669	0.604
2-2	0.631	0.661	0.592
2-3	0.648	1.000	0.793
3-1	0.726	0.792	0.611
3-2	0.837	0.913	0.562
3-3	0.840	0.903	0.548

zultatima pa se u dovoljno iteracija ni ne primjeti razlika za različite funkcije dobrote enkriptora.

Tablica 4.4: Rezultati različitih konstanti

Bijekcija	*Ključ*	*Promjena_bitova*	*Bob*	*Trudy*	Max *Alice*	*Bob - Trudy*
1	1	1	1	1	0.986	0.450
4	1	1	1	1	0.991	0.440
1	4	1	1	1	0.992	0.445
1	1	4	1	0	0.993	0.455
1	1	1	4	4	0.956	0.445

4.2.3. Rezultati dobiveni mijenjanjem duljine teksta

Tekst nad kojim radimo enkripciju i dekripciju može biti različitih duljina. Te duljine mjerimo u oktetima. Jedan oktet označava duljinu od osam bitova. Ovaj test provjerava koliko ta duljina utječe na kriptiranje. Zbog već spomenutog problema kod duljine izvođenja programa, funkcija provjere bijekcije ovdje je zanemarena.

Pokazalo se da se bolje rezultate dobiva s manjim brojem bitova teksta (tablica 4.5). Kod teksta duljine 1, 2 ili 3 okteta, rezultati su slični. Ipak, za dulje tekstove primjetan je pad u kvaliteti algoritama kriptiranja.

Tablica 4.5: Rezultati različitih veličina otvorenog teksta

Broj okteta	Max *Alice*	*Bob*	*Trudy*
1	0.948	1.000	0.560
2	0.855	1.000	0.544
3	0.916	1.000	0.557
4	0.813	0.860	0.535
5	0.797	0.840	0.577
6	0.744	0.881	0.680
7	0.632	0.694	0.623
8	0.638	0.703	0.626

4.2.4. Rezultati dobiveni mijenjanjem vjerojatnosti primitiva SKIP

SKIP je primitiv koji koristimo za preskakanje promatranog bita, tj. nad njim (bitom) ne radimo promjene. S obzirom da primitiv *XOR*, koji se pokazao kao važan i jako koristan primitiv u prvom testu, radi nad cijelim oktetom, ovim testom provjeravam je li vjerojatnost odabira primitiva *SKIP* (koji zbog preskakanja bitova daje veći naglasak na pristupu oktetu kao cjelini) u korelaciji s rezultatima. Da bi imalo smisla provoditi ovaj test, duljina teksta postavljena je na dva okteta.
Rezultati su pokazali minimalnu razliku za testove s različitim vjerojatnostima odabira primitiva *SKIP*.

Tablica 4.6: Rezultati različitih vjerojatnosti primitiva SKIP

SKIP vjerojatnost	Max *Alice*	*Bob*	*Trudy*
50%	0.856	1.000	0.539
66%	0.848	1.000	0.552
75%	0.853	1.000	0.546
83%	0.818	1.000	0.546
90%	0.782	1.000	0.546

5. Zaključak

Kriptiranje uz pomoć stabala generiranih genetskim programiranjem dalo zanimljive rezultate.

Za početak, pokazalo se da je većina korištenih primitiva nedovoljno jednostavna za dekriptiranje. Primitivi koji u svom dosegu mijenjaju bit po bit previše su usporavali evoluciju. Također, operacija *PLUS* pokazala se ireverzibilna za dekriptiranje uz pomoć jednostavnih primitiva. S druge strane, *XOR* operacija je imala najbolje rezultate. U kombinaciji s primitivom *SKIP* pokazalo se da je moguće ostvariti kvalitetnu komunikaciju sudionika, bez mogućnosti napada.

Drugi parametar koji je imao veliku ulogu je duljina teksta. Duljina teksta do 3 okteta daje vrlo dobre rezultate. Za veće duljine pada kvaliteta kriptiranja. Pod tim smatram činjenicu da *Bob* više ne može dekriptirati sa stopostotnim učinkom, a dobrota mu se približava napadačevoj.

Istraživanje ima li vjerojatnost jednog primitiva (*SKIP*) utjecaj na rješenje pokazalo je da nema. Razlika između dekriptiranja napadača i *Bob*-a ostala je slična za različite vjerojatnosti.

Sve varijable funkcije dobrote enkriptora doprinose boljim rezultatima. To vidimo iz rezultata drugog testa gdje su mijenjane konstante koje množe varijable te funkcije. Svakoj konstanti je dan iznos od 50% udjela u funkciji i svaki put kriptiranje je dovedeno do odličnih rezultata.

Smatram da bi se algoritam kriptografije dobiven genetskim programiranjem mogao koristiti. Zbog mogućnosti ostvarivanja bijekcije te stopostotnog dekriptiranja, ispunjava osnovne zahtjeve kriptografije. Također, napadač nema uspjeha u napadima na komunikacijski kanal. Ipak, potrebno je voditi računa o činjenici da bi tekst koji se kriptira ovim algoritmom trebalo podijeliti na manje dijelove od 1, 2 ili 3 okteta i takve kriptirati.

LITERATURA

Blažević. Simetrična kriptografija. URL http://web.zpr.fer.hr/ergonomija/2005/blazevic/simkripto.htm.

CARNet CERT. *AES algoritam*. CARNet CERT and Laboratorij za signale i sustave, FER, 2003a. URL https://www.cis.hr/www.edicija/LinkedDocuments/CCERT-PUBDOC-2003-06-24.pdf.

CARNet CERT. *DES algoritam*. CARNet CERT and Laboratorij za signale i sustave, FER, 2003b. URL https://www.cis.hr/www.edicija/LinkedDocuments/CCERT-PUBDOC-2003-06-24.pdf.

CARNet CERT. *IDEA algoritam*. CARNet CERT and Laboratorij za signale i sustave, FER, 2003c. URL https://www.cis.hr/www.edicija/LinkedDocuments/CCERT-PUBDOC-2003-06-24.pdf.

Luka Donđivić i Ivan Kokan. Kriptografija. URL https://web.math.pmf.unizg.hr/~duje/kript/osnovni.html.

Andrej Dujella. Kriptografija. URL https://web.math.pmf.unizg.hr/~duje/kript/osnovni.html.

Sveučilište Josipa Jurja Storssmayera Odjel za matematiku. Stabla. URL http://www.mathos.unios.hr/spa/Files/materijali/SPA_skripta_ch09.pdf.

Marko Čupić. *Prirodom inspirirani optimizacijski algoritmi.Metaheuristike.* 2013.

Kriptiranje komunikacije uz pomoć evolucijskih algoritama

Sažetak

Kriptografija je grana koja u računarstvu danas ima veliku ulogu: očuvanje sigurnosti. Cilj je kriptografije onemogućiti napadačima komunikacije da pročitaju poruke koje idu komunikacijskim kanalom. To se radi uz pomoć posebnih algoritama kriptiranja koji obične podatke tj. otvoreni tekst pretvaraju u kriptirane, nečitljive podatke.
Kroz ovaj rad pokušala sam kriptografiju povezati sa evolucijskim algoritmom: genetskim programiranjem. Ono je jedan od algoritama koje po uzoru na odnose iz prirode traži rješenje zadanog problema. U ovom kontekstu problem je bio izraditi algoritam kriptiranja uz pomoć strukture stabla. Primitivi koji izgrađuju stablo inspirirani su nekim već postojećim algoritmima kriptiranja. Oni određuju korake kriptiranja. Uz pomoć evolucije i evolucijskih operatora, genetskim programiranjem iz iteracije u iteraciju tražimo bolja i bolja rješenja.
Traženje algoritma kriptiranja ostvareno je na temelju suparničkog učenja sudionika komunikacije. Sudionicima koji međusobno komuniciraju komunikacijskim kanalom cilj je održati svoju komunikaciju tajnom, tj. nerazumljivom i nemogućom za dekripciju sudioniku koja nema ključ. S druge strane, napadač je sudionik koji sluša tu komunikaciju i ima za cilj dešifrirati ju. Razlika u odnosu na legitimne sudionike je ta što napadač nema ključ komunikacije. Zbog toga su definirane funkcije dobrote za rješenja sudionika tako da u slučaju da napadač jednostavno dekriptira poruku, sudionik koji ju je kriptirao ima umanjenu dobrotu i obrnuto. Sudioniku koji enkriptira također je bitno da onaj sudionik kojem šalje poruku može dekriptirati tu poruku. Zbog toga su njihove dobrote u pozitivnoj korelaciji. Na kraju, rezultati su pokazali da je neke parametre komunikacije moguće optimirati da bi se dobila bolja komunikacija. To su prije svega skupovi primitiva koji se koriste pri izgradnji algoritma te duljina teksta.

Ključne riječi: genetsko programiranje, kriptografija, kodiranje, dekodiranje, simetrični ključ, napadač, suparničko učenje

Encryption of Communication Using Evolutionary Algorithms

Abstract

Cryptography is a branch that plays a major role in modern computing - preserving security. The goal of cryptography is to prevent attackers from reading messages traversing a communication channel. This is achieved using special cryptographic algorithms that convert exposed data into encrypted, unreadable data.
In this paper I will attempt to make a link between cryptography and evolutionary algorithms using genetic programming. Evolutionary algorithms are a subset of algorithms that use relationships found in nature in order to generate a solution to a given problem. In this context, the task was creating an encryption algorithm using a tree structure. The primitives that make up the tree are inspired by hitherto known encryption algorithms as they determine the encryption steps. With the help of evolution and evolutionary operators, genetic programming with each iteration searches for more superior solutions.
The search for an encryption algorithm was based on adversarial machine learning of the participants in the communication process. For those participants the goal is to keep the communication secret, ie. incomprehensible and impossible to decrypt for a participant without a key. On the other hand, an attacker is a participant who listens to that communication and aims to decrypt it. The thing that differs them from legitimate participants is that that the attacker doesn't have a key. Therefore, we define functions of goodness for a participant's solution so that in case an attacker simply decrypts the message, the participant who encrypted it has a diminished goodness and vice versa. It is also important to the sender who encrypts the data that the participant to whom the message is sent can decrypt it. Consequently, their goodnesses are in positive correlation. In the end, the results showed that there are some communication parameters that can be optimized to better the communication. Those are, primarily, the primitive sets used in constructing the algorithm and the text length.

Keywords: genetic programming, cryptography, encryption, decryption, symmetric key, attacker, adversarial learning