ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი

ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი

კომპიუტერულ მეცნიერებათა დეპარტამენტი

გიორგი გელაშვილი

გენეტიკური ალგორითმების გამოყენება კრიპტოანალიზში

ინფორმაციული სისტემები

ნაშრომი შესრულებულია ინფორმაციული სისტემების მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ზურაბ ქოჩლაძე

ტექნიკურ მეცნიერებათა კანდიდატი;
გამოყენებითი ინფორმატიკის კათედრის
ასოცირებული პროფესორი;

თბილისი
2017

ანოტაცია

ნაშრომში განხილულია ინფორმაციის დაშიფვრის ერთ-ერთი კლასიკური მეთოდის, კერძოდ კი **„ვიჟინერის შიფრის“** კრიპტოანალიზი, **გენეტიკური ალგორითმების** დახმარებით.

კრიპტოგრაფია, კომპიუტერული ტექნიკის განვითარებასთან ერთად კიდევ უფრო დიდ როლს თამაშობს ჩვენს ცხოვრებაში, რადგან საჭირო ხდება უფრო და უფრო მეტი ინფორმაციის დაშიფრული სახით არსებობია, ამისათვის კი საჭიროა დასაშიფრი ალდორითმების ვარგისიანობის შემოწმება(**კრიპტოანალიზი**).

აქ მოყვანილია ინფორმაციის დაშფვრის ზოგადი და კერძო მეთოდები. ასევე მოყვანილია ვიჟინერის შიფრის აღწერა და მისი გატეხვის, კრიპტოანალიზის ცნობილი საშვალებები.

ჩვენს პროგრეამულ უზრუნველყოფას საშვალება აქვს დაშიფროს ქართული ტექსტი ვიჟინერის ალგორიტმის საშვაებით ასევე ჩაუტაროს კრიპტოანალიზი ამავე მეთოფდით დაშიფრულ ტექსტს, სადაც დეშიფრაციის **სიტყვა გასაღების** სიგრძის დასადგენად გამოიყენება **კასისკის ტესტი**, ხოლო თვითონ გასაღების გამოსათვლელად, შედარებით ახალ მეთოდს გამოთვლით სისტემებში გენეტიკურ ალგორითმებს, ჩვენს მიერ შედგენილი ალგორითმი.

ნაშრომის მთავარი მიზანი არის, კრიპტოანალიზის სფეროში, გენეტიკური ალგორითმების გამოყენების მაღალი წარმატებულობის ჩვენება.

Annotation

In this paper discussed one of the classical methods of encryption, particularly cryptanalysis of **Vigenere cipher** using **Genetic Algorithms**.

After improving of computer technologies Cryptography plays greater role in our life than earlier, because there is a necessity to exist more encrypted information and this definitely needs to check usefulness of encryption algorithms (**Cryptanalysis**).

Here discussed general and separate methods of encryption also described Vegenere cipher method and the famous ways of it's cracking and cryptanalysis.

Our software can encrypt Georgian texts using Vigenere algorithm and also cryptanalyze it if this text is encrypted with the same method. We use **Test of Kasiski** for finding length of keyword and modified Genetic Algorithms for calculating keyword itself, it is comparatively new method in this field.

To show productivity of GA-s in a cryptanalysis is the main goal of this paper.

სარჩევი

შესავალი

ტექნოლოგიების განვითარებასთან ერთად უფრო და უფრო მნიშვნელოვანი ხდება პირადი, თუ ზოგადი ინფორმაციის დაცვა. რადგან ინფორმაცია ზოგისათვის ცნობილი უნდა იყოს და ზოგისათვის დამალული, ამიტომ საჭიროა მისი დაშიფვრა. დაშიფვრისათვის კი გამოიყენება შიფრაციის ალგორითმები. იმისათვის რომ, ამა თუ იმ ალგორითმის ნდობა გაგვიჩნდეს, საჭიროა მისი გამოსადეგობის შემოწმება, უნდა ვიცოდეთ რამდენად იცავს ინფორმაციას, რათა უცხო პირის ხელში ადვილად არ ჩავარდეს. ამისათვის ის უნდა შემოწმდეს ისეთივე მეთოდებით, როგორსაც ბოროტგანმზრახველი გამოიყენებდა, ანუ, უნდა ჩაუტარდეს კრიპტოანალიზი, რომელიც მრავალგვარია.

ამ ნაშრომში წარმოდგენილია კრიპტოანალიზისთვის საკმაოდ ახალი, არაკლასიკური, ბუნებრივ პროტოტიპებზე დაფუძვნებული მეთოდი „გენეტიკური ალგორითმები“.

შესავალი კრიპტოგრაფიაში

გარკვეული ინფორმაციის დაშიფვრა რომელსაც შეისწავლის კრიპტოგრაფია (მეცნიერება ინფორმაციის დაფარვის შესახებ) ან დამალვა რომელსაც შეისწავლის სტენოგრაფია (მეცნიერება ინფორმაციის დამალვის შესახებ) ოდითგანვე ითვლებოდა წარმატების, დაცულობისა თუ უპირატესობის მოპოვების ძირითად საშვალებად. ისტორიულ წყაროებში მრავლად გვხვდება, სხვადასხვა მეთოდები და საშვალებები ამისათვის.

1.1 ტერმინოლოგია

კრიპტოგრაფიული სისტემა ყოველთვის გულისხმობს 2 ან მეტ მონაწილეს, „გამგზავნს“ და „მიმღებს“, რომელთაც სურთ ერთმანეთს გადასცენ რაიმე სახის ინფორმაცია, ისე რომ ამ სისტემის გარეშე პირმა ვერ შეძლოს ამ ინფორმაციის მოპოვება. (კერძო შემთხვევებში მონაწილე შეიძლება იყოს ერთი პირი, რომელსაც სურს მოახდინოს ინფორმაციის საიმედოდ შენახვა, ანუ ინფორმაცია გადასცეს თავის თავს, ოღონდ მომავალში). თვითონ გადასაცემ ინფორმაციას ეწოდება ღია ტექსტი. ინფორმაციის სახეცვლილებას ისე, რომ დაფარულ იქნას მისი აზრი, შიფრაციას უწოდებენ, უკუპროცესს — დეშიფრაციას, მიღებულ შედეგს — შიფროტექსტს,

ხოლო შიფრაცია/დეშიფრაციის ალგორითმს — შიფრს. ეს პროცესი უმეტეს შემთხვევაში დამოკიდებულია გასაღებზე. ეს არის საიდუმლო პარამეტრი, რომელიც ცნობილია მხოლოდ ურთიერთმოკავშირე მხარეებისათვის. შიფრები გასაღების გარეშე უსარგებლოა, რადგან არსებობს მათი გატეხვის ტრივიალური ხერხები. კრიპტოსისტემა წარმოადგენს შიფრის, ყველა შესაძლო ღია ტექსტის, შიფროტექსტის და გასაღების ერთობლიობას. კრიპტოგრაფიული პროტოკოლები წარმოადგენენ წესების ერთობლიობას, რომლებიც სრულდება კრიპტოგრაფიული სისტემის მონაწილეების მიერ, თანამიმდევრობით და უპირობოდ.

ტერმინი „კოდი“ კრიპტოგრაფიაში აღნიშნავს ჩვეულებრივი ტექსტის ნაწილის შეცვლას კოდურის სიტყვით ან ფრაზით (მაგ. „ხმალი“ შესაძლოა აღნიშნავდეს „გაანადგურე მტერი“). კოდები სერიოზულ კრიპტოგრაფიაში აღარ გამოიყენება რადგან, მისი გამოყენება მოითხოვს, მონაცემთა გადაცემამდე მოკავშირეთა შორის ყველა შესაძლო მონაცემი შესაბამისი სიტყვებით კოდირებული და ურთიერთშეთანხმებული იქნას . ამავე დროს ყოველთვის არსებობს მონაცემთა სიმრავლე, რომელთათვისაც შესაბამისი კოდური სიტყვა ჯერ არ შეთანხმებულა.

1.2 კრიპტოგრაფიის ისტორიიდან

თანამედროვე ერამდე, კრიპტოგრაფია გამოიყენებოდა შეტყობინებათა საიდუმლოების დასაცავად, ტექსტის გარდაქმნით სიმბოლოების გაურკვეველ ნაკრებად, ისე რომ უკუგარდაქმნა იოლი ყოფილიყო მხოლოდ შეტყობინების ადრესატისათვის.

ჯერ კიდევ ძველი წელთაღრიცხვის მეხუთე საუკუნიდან ცნობილია გადანაცვლების შიფრი სციტალა, რომელსაც იყენებდა სპარტელი მხედართმთავარი ლისანდრე. სციტალა ნიშნავდა კვერთხს, რომელსაც ატარებდნენ იმდროინდელი მხედართმთავრები. დაშიფვრა ხდებოდა ამ კვერთხის საშუალებით და ალგორითმმაც აქედან მიიღო ეს სახელი. დაშიფვრა ხდებოდა შემდეგნაირად: ვიწრო პაპირუსის ლენტს მჭიდროდ დაახვევდნენ სციტალაზე და ამგვარად მიღებულ ზედაპირზე ჰორიზონტალურად ჩაწერდნენ ტექსტს. შემდეგ ლენტს მოაშორებდნენ სციტალას და მიიღებდნენ პაპირუსის ლენტს მასზე გაბნეული ასოებით. ასეთ ლენტს უგზავნიდნენ ადრესატს. თუ მოწინააღმდეგე ჩაიგდებდა ხელში ასეთ ლენტს, ის ვერ შეძლებდა ინფორმაციის წაკითხვას მაშინაც კი, როდესაც მისთვის ცნობილი იყო დაშიფვრის ალგორითმი, რადგანაც, მისთვის უცნობი იყო კვერთხის დიამეტრი.

როგორც ვხედავთ, ამ შემთხვევაში ღია ტექსტის ასოები არ იცვლება, ისინი იცვლიან ადგილებს, ესე იგი სციტალა წარმოადგენს გადანაცვლების შიფრს. იმისათვის, რომ მოწინააღმდეგემ შეძლოს დაშიფრული ტექსტის გაშიფვრა, მან უნდა იპოვოს გასაღები, ანუ გაიგოს იმ კვერთხის დიამეტრი, რომელზედაც იყო დახვეული პაპირუსის ლენტი. ძალისმიერი შეტევა ამ შემთხვევაში არც თუ ეფექტური იქნება, რადგანაც მოგვიწევს გამოვცადოთ სხვადასხვა დიამეტრის კვერთხები. სციტალა გატეხა ცნობილმა ბერძენმა მეცნიერმა არისტოტელემ. მან შემოგვთავაზა შემდეგი, დავახვიოთ პაპირუსის ლენტი კონუსის ფორმის კვერთხზე. იქ, სადაც კონუსის დიამეტრი დაემთხვევა სციტალას დიამეტრს გამოჩნდება აზრის მქონე ასოთშეთნხმება, რაც იქნება იმის ნიშანი, რომ სციტალას გასაღები ნაპოვნია.

სრულიად განსხვავებულ შიფრს იყენებდა რომის იმპერატორი გაი იულიუს ცეზარი. ეს შიფრი ისტორიაში შევიდა ცეზარის შიფრის სახელით. ცეზარი ღია ტექსტის ასოს ცვლიდა ასოთი, რომელიც ღია ტექსტის ასოდან დაშორებული იყო სამი ადგილით.

A B C D E F G H I K L M N O P Q R S T V X
D E F G H I K L M N O P Q R S T V X A B C

როგორც სურათიდან ჩანს **A** იცვლებოდა **D**-თი, **B** – **E**-თი და ასე შემდეგ. შეიძლება ასე გამოიყურებოდა დაშიფრული სახით მისი ცნობილი ფრაზა:

"BHQM, BMGM, BMFM" („მივედი, ვნახე, გავიმარჯვე“).

აშკარა, რომ ამ შემთხვევაში, შიფრის გასაღებს წარმოადგენს მანძილი ღია ასოს მდებარეობიდან იმ ასომდე, რომლითაც ღია ასო იცვლება შიფროტექსტში. როგორც ვნახეთ, ცეზა-

რი იყენებდა გასაღებს K = 3, მაგრამ შესაძლებელია სხვა მნიშვნელობის გამოყენებაც, თუმცა არც თუ ისე ბევრის (შესაძლებელია მხოლოდ ოცი გასაღები), თუ ასოების რაოდენობაა ოც-დაერთი (როგორც იყო ცეზარის დროს), ზოგადად კი K = N -1, სადაც N არის ასოების რაო-დენობა მოცემულ ანბანში). ასევე დასტურდება რომ საიდუმლო დამწერლობას და გასაიდუმლოების სხვადასხვა მეთოდებს იყენებდნენ უძველესი შუმერები, ეგვიპტელები და ჩინელები.

ცნობილია, რომ შუა საუკუნეებისა და რენესანსის ეპოქაში საიდუმლო ალგორითმების მისაღებად იღვწოდნენ იმ დროის გამოჩენილი ადამიანები, მათ შორის, ფილოსოფოსი ფრანსის ბეკონი, მათემატიკოსები ფრანსუა ვიეტი, ჯორდანო კარდანო, ჯონ ვალისი და სხვ. გაჩნდა დასაიდუმლოების შედარებით რთული მეთოდებიც.

საინტერესოა, რომ IX საუკუნის ბოლოს არაბების მიერ გამოქვეყნებულ ენციკლოპედიაში ცალკე თავი დაეთმო კრიპტოგრაფიის საკითხებს. ნიშანდობლივია, რომ სხვა საკითხებთან ერთად პირველად არის განხილული კრიპტოგრაფიული ალგორითმის ანალიზის, ანუ კრიპტოანალიზის, სტატისტიკური მეთოდი. სტატისტიკური მეთოდი გულისხმობს, რომ ცალკეულ სიმბოლოებს, კერძოდ, ბუნებრივი სასაუბრო ენის ცალკეულ ასო-ნიშნებს გააჩნია განსხვავებული ალბათობები (მაგალითად, ქართულ ტექსტში „ბ“ ან „გ“ ასო შედარებით მეტი სიხშირით ჩნდება, ვიდრე, ვთქვათ „ც“ ასო-ნიშანი და სხვ.). ნიშნების სიხშირული მნიშვნელობების გათვალისწინება დღესაც წარმოადგენს კრიპტოანალიზის ერთ-ერთ შესაძლებლობას ალგორითმის გასატეხად. 1467 წელს იტალიელმა ლეონ ბატისტა ალბერტიმ გამოიგონა პოლიალფაბეტური შიფრი, რომელითაც ტექსტის სხვადასხვა ნაწილი სხვადასხვა შიფრით იშიფრებოდა. მანვე გამოიგონა პირველი მექანიკური დამხმარე მოწყობილობა (ბორბალი). ვიგენერის პოლიალფაბეტურ შიფრში გამოიყენებოდა პაროლი, რომელიც განსაზღვრავდა, ტექსტის რომელი სიმბოლო შიფრის რომელი სიმბოლოთი უნდა შეცვლილიყო. 1800 წელს ბებიჯმა დაამტკიცა, რომ ასეთი ტიპის პოლიალფაბეტური შიფრებიც ექვემდებარებოდა გარკვეულ სიხშირულ ანალიზს.

მიუხედავად იმისა, რომ სიხშირული ანალიზი საკმაოდ მძლავრი საშუალებაა კრიპტოანალიზისათვის, ის მაინც შედარებით უცნობ მეთოდად რჩებოდა. ამ მეთოდის გარეშე შიფრის გატეხვა მოითხოვდა შიფრის ალგორითმის ცოდნას, რასი მოპოვებაც შპიონაჟით, მოქრთამვებით და გამოძალვებით ხერხდებოდა. XIX საუკუნეში დადგინდა, რომ შიფრის ალგორითმის საიდუმლოდ შენახვა შიფრის დაცულობის გარანტი არ არის და რომ შიფრი

დაცული უნდა რჩებოდეს მაშინაც კი, როდესაც მოწინააღმდეგე სრულად ფლობს გამოყენებული შიფრის ალგორითმის შესახებ ინფორმაციას. სხვა სიტყვებით, იდეალური შიფრის დაცულობა დამოკიდებული უნდა იყოს შიფრის გასაღების (პაროლის) დაცულობაზე. კრიპტოგრაფიის ეს ფუნდამენტური პრინციპი პირველად ჩამოაყალიბა 1883 წელს ავგუსტ კერჰოფმა („კერჰოფის პრინციპი“), შემდგომში ეს პრინციპი განავრცო კლოდ შენონმა — „მტერი იცნობს სისტემას“.

კომპიუტერების გამოჩენამ შესაძლებელი გახადა შექმნილიყო გაცილებით რთული და დახვეწილი შიფრები. შესაძლებელი გახდა, დაშიფრულიყო ყველანაირი მონაცემი ბინარულ ფორმატში. კომპიუტერული შიფრები უმეტესწილად მანიპულირებენ ბიტებზე ან ბიტების ჯგუფებზე, განსხვავებით კლასიკური პოლიალფაბეტური შიფრებისაგან, რომლებიც ტექსტის ცალკეულ სიმბოლოებს ამუშავებდნენ. ამავე დროს გააადვილდა შიფრების კრიპტოანალიზი. მიუხედავად ამისა, თანამედროვე შიფრები მონაცემების დაცვის მაღალდონეს უზრუნველყოფენ, მათი კრიპტოანალიზის გზით გატეხვა იმდენად კოლოსალურ გამოთვლით და დროით რესურსებს მოითხოვს, რომ პრაქტიკულად შეუძლებლად მიიჩნევა.

ინტენსიური ღია კვლევები კრიპტოგრაფიის დარგში შედარებით ახლო პერიოდში დაიწყო — 70-იან წლებში ეს იყო DES-ის (Data Encryption Standard) გამოქვეყნებული სპეციფიკაციები, დიფი-ჰელმანი და RSA ალგორითმი. მას შემდეგ ინტენსიურად დაიწყო კრიპტოგრაფიის გამოყენება კომუნიკაციებში, კომპიუტერულ ქსელებსა და მთლიანად კომპიუტერულ უსაფრთხოებაში. თანამედროვე კრიპტოგრაფიული სისტემების დაცულობა დამოკიდებულია გარკვეულ მათემატიკურ პრობლემებზე, მაგ. რიცხვის ფაქტორიზაცია, დისკრეტული ლოგარითმები და ელიპსური მრუდეები. დამტკიცებულია, რომ ასეთი კრიპტოგრაფიული სისტემა დაცულია, თუ მათემატიკური პრობლემის ეფექტურად და მოკლე დროში გადაჭრა შეუძლებელია. გამონაკლისია „ერთჯერადი ბლოკნოტი“, არ არსებობს მისი გატეხვის თეორიული გზა, და წარმოადგენს იდეალურ კრიპტოგრაფიულ ალგორითმს.

თუ XX საუკუნემდე კრიპტოგრაფია ძირითადად მანიპულირებდა სიმბოლოებზე და იყენებდა ლინგვისტიკის ელემენტებს, XX საუკუნიდან ხდება მათემატიკის ინტენსიური გამოყენება, ინფორმაციის თეორიის, სტატისტიკის, კომბინატორიკის, აბსტრაქტული ალგებრის და რიცხვთა თეორიის ჩათვლით. ბოლო წლებში მიმდინარეობს კრიპტოგრაფიაში კვანტური ფიზიკის ელემენტების შემოტანაც (იხ. კვანტური კრიპტოგრაფია და კვანტური გამოთვლები)

1.3 მექანიკური შიფრატორები

I მსოფლიო ომსა და კომპიუტერების საყოველთა გავრცელებამდე (1950-1960-იანი წლები) პერიოდში შიფრაციისათვის ფართოდ გამოიყენებოდა ჩანაცვლების პოლიალფაბეტური შიფრები, რომლებიც მექანიკური ან ელექტრომექანიკური აპარატების სახით იყო რეალიზებული. ზოგიერთი რეალიზაცია ერთდროულად გამოჩნდა და დაპატენტდა კიდეც 1919 წელს. მათ შორის გამორჩეული იყო გერმანული საშიფრი მანქანა ენიგმა, რომელსაც გერმანელი სამხედროები 1930 წლიდან იყენებდნენ. ყველა ეს მანქანა მსგავსი სქემით მოქმედებდა - შეტანილი სიმბოლოები ჩანაცვლდებოდა სახვა სიმბოლოებით, რომელთაც თვით მანქანა ირჩევდა განსაზღვრული წესით. ეს ჩანაცვლების წესები წინასწარ დაიტანებოდა როტორულ დისკებზე. დისკები მექანიკურად იყო დაკავშირებული ერთმანეთთან, რაც ჩანაცვლების ანბანთა ასტრონომულ რაოდენობას უზრუნველყოფდა. მიუხედავად ამისა, ზოგიერთი ამ მანქანებით დაშიფრული ინფორმაციის გატეხვა მაინც მოხერხდა - ნაწილი II მსოფლიო ომისდაწყებამდე, ნაწილი კი ომის პერიოდში. ომის მიმდინარეობის დროს, პოლონელმა კრიპტოანალიტიკოსმა მარიან რეჯევსკიმ შეძლო ენიგმას პირველი წარმატებული კრიპტოანალიზი განეხორციელებინა. შედეგები გადაეცა დიდი ბრიტანეთის კრიპტოანალიტიკურ ცენტრს (Bletchley Park), სადაც ომის წლებშივე შექმნეს მანქანა (Bombe) ამ შიფრატორის საწყისი პარამეტრების აღსადგენად.

ამავე დროს, ორი მსგავსი როტორული შიფრატორი, SIGABA და Typex დღემდე გაუტეხელია.

1.4 თანამედროვე კრიპტოგრაფია

ინფორმაციის დაშიფვრის საჭიროება დღესდღეობით უდიდესია, ის ახლა უფრო მნიშვნელოვანია ვიდრე როდისმე ყოფილა, რადგან უფრო მეტია მოთხოვნა, უფროდაფროდაუფორ მეტი რამ ხდება დამოკიდებული კომპიუტერსა და ავტომატურ სისტემებზე, რაც დაცვის საჭიროებას მეტად ზრდის.

1.4.1 სიმეტრიული კრიპტოგრაფია

სიმეტრიული კრიპტოგრაფია იყენებს მეთოდებს, რომლის დროსაც ინფორმაციის გამგზავნი და მიმღები იყენებენ ერთსა და იმავე გასაღებს (იშვიათად სხვადასხვას, მაგრამ ამ შემთხვევაში ერთი გასაღები იოლად გამოითვლება მეორიდან). 1976 წლამდე ეს შიფრაციის ერთადერთი მეთოდი იყო. თანამედროვე სიმეტრიული კრიპტოგრაფია დაკავშირებულია ძირითადად ბლოკურ შიფრებთან, ნაკადურ შიფრებთან და მათ გამოყენებასთან.

1.4.2 ბლოკური შიფრი

ბლოკური შიფრი წარმოადგენს ფაქტობრივად პოლიალფაბეტური შიფრის მოდიფიკაციას: აიღება საწყისი ტექსტის გარკვეული სიგრძის ნაწილი (ბლოკი) და გასაღები, შედეგად მიიღება იგივე (იშვიათად განსხვავებული) სიგრძის შიფროტექსტი. შიფროტექსტის შემადგენელი ბლოკების ერთმანეთთან შერწყმისათვის გამოიყენება სხვადასხვა მეთოდები, რომლებსაც მთლიანობაში ქმედების რეჟიმი ეწოდებათ. მონაცემთა შიფრაციის სტანდარტი (Data Encryption Standard — DES) და გაუმჯობესებული შიფრაციის სტანდარტი (Advanced Encryption Standard — AES) წარმოადგენენ ბლოკურ შიფრებს. DES (და მისი ნაირსახეობა 3DES) ჯერაც რჩება ერთ-ერთ ყველაზე პოპულარულ ალგორითმად და ფართოდ გამოიყენება. თუმცა მისი გასაღების სიგრძის არასაკმარისობის გამო, ხდება მისი ჩანაცვლება სხვა, უფრო თანამედროვე ალგორითმებით. დღემდე გამოგონილია მრავალი ბლოკური შიფრი, მათი უმეტესობა გატეხილია წარმატებული კრიპტოანალიზის შედეგად.

1.4.3 ნაკადური შიფრი

ნაკადური შიფრი ქმნის განუსაზღვრელი სიგრძის გასაღებს, რომელიც შემდგომ უერთდება საწყის ინფრომაციას (ბიტობრივად ან ბაიტობრივად). გამომავალი ინფორმაცია დამოკიდებულია შიფრის შინაგან მდგომარეობაზე, რომელიც მოქმედების

მიმდინარეობისას იცვლება. საწყისი მსგომარეობა დამოკიდებულია შიფრის გასაღებზე (ზოგიერთ ნაკადურ შიფრში ტექსტზეც). ნაკადური შიფრის მაგალითია RC4.

1.4.4 ჰეშ ფუნქციები

კრიპტოგრაფიული ჰეშ-ფუნქციები (ტექსტის ანაბეჭდის ფუნქციები) წარმოადგენენ კრიპტოგრაფიული ალგორითმების მნიშვნელოვან კლასს. ისინი იღებენ საწყის მნიშვნელობად ტექსტს და უკან აბრუნებენ ფიქსირებული სიგრძის ჰეშს, რომელიც დაკავშირება საწყის მნიშვნელობასთან პრობლემას წარმოადგენს. ასეთ ფუნქციებს ცალმხრივ ფუნქციებსაც ეძახიან. საუკეთესო ალგორითმებისათვის კოლიზიები (ორი ტექსტი, რომელთა ჰეში ერთი და იგივეა) რთული მოსაძებნი უნდა იყოს და ამის ალბათობა მინიმუმამდე უნდა იყოს დაყვანილი.

1.4.5 შეტყობინების აუთენტიფიკაცია

შეტყობინების აუთენტიფიკაციის კოდები ჰეშ-ფუნქციების მსგავსია, იმ განსხვავებით, რომ ჰეშ-მნიშვნელობის შესამოწმებლად გამოიყენება საიდუმლო გასაღები.

1.5 კრიპტოანალიზი

იმისათვის რომ დაშიფვრის ესა თუ ის მედოდი გამოვიყენოთ და ვიცოდეთ მისი შესაძლებლობები, საჭიროა ჩაუტარდეს კრიპტოანალიზი, რათა დავრწმინდეთ მის მედეგობაშ. კრიპტოანალიზის უმთავრესი ამოცანა კი არის იპოვოს სუსტი წერტილები კრიპტოგრაფიულ სისტემაში, რათა შეძლოს ამ სისტემით დაცული ინფორმაციის მოპოვება. კრიპტოანალიზი შეიძლება გამოყენებულ იქნას, როგორც ბოროტგანმზრახველის, ასევე კრიპტოსისტემის შემქმნელის მიერ ამ სისტემაში ნაკლის აღმოსაჩენად (გაცილებით იოლია სისტემის ნაკლის პოვნა, ვიდრე მისი სრულყოფილების დამტკიცება).

ფართოდ გავრცელებული (და ამავე დროს მცდარი) აზრია ის, რომ ნებისმიერი შიფრის გატეხვა შესაძლებელია სასრულ დროში. II მსოფლიო ომის პერიოდში კლოდ შენონმა დაამტკიცა, რომ შიფრის გატეხვა თეორიულად შეუძლებელია, თუ მისი გასაღები ნამდვილად

შემთხვევითია, არ მეორდება, ტექსტის სიგრძის ტოლი ან მეტია და დაცულია პირდაპირი წვდომისაგან სხვა პირებისაგან. ამ პირობებს მხოლოდ ე.წ. ერთჯერადი ბლოკნოტი აკმაყოფილებს. სხვა შიფრები, გარდა ამ შიფრისა, შეიძლება გატყდეს ე. წ. უხეში ძალის მეთოდით (მიუხედავად მისი დროისა). ამავე დროს შესაბამისი გამოთვლების ოდენობა ექსპონენციურად დამოკიდებულია გასაღების სიგრძეზე. სისტემა ითვლება დაცულად, თუ ამ გამოთვლების ოდენობა აღემატება ნებისმიერი მოწინააღმდეგის შესაძლებლობებს, ამავე დროს თუ არ არსებობს გატეხვის სხვა მეთოდი, რომელიც უფრო სწრაფი იქნებოდა, ვიდრე უხეში ძალის მეთოდი. დღესდღეობით ერთადერთ თეორიულად გაუტეხელ შიფრად ერთჯერადი ბლოკნოტი რჩება.

არსებობს კრიპტოანალიზის რამდენიმე ტიპი. ძირითადი განსხვავება მათ შორის არის მოწინააღმდეგისათვის ცნობილი ინფორმაციის ოდენობა და სახე, რის საფუძველზეც ხდება შემდეგ ანალიზი. ცნობილი შიფროტექსტის შემთხვევაში კრიპტოანალიტიკოსს ხელთ აქვს მხოლოდ დაშიფრული ინფორმაცია. ცნობილი ღია ტექსტის შემთხვევაში კრიპტოანალიტიკოსს აქვს შიფროტექსტი და მისი შესაბამისი ტექსტი, მაგრამ არ აქვს გასაღები. არჩეული ტექსტის შემთხვევაში ანალიტიკოსს შეუძლია თვითონ შეარჩიოს ტექსტი და მიიღოს მისი შესაბამისი შიფროტექსტი. და ბოლოს არჩეული შიფროტექსტის შემთხვევაში ანალიტიკოსს შეუძლია შეარჩიოს შიფროტექსტები და შეისწავლოს მათი შესაბამისი ტექსტები.

სიმეტრიული კრიპტოსისტემების კრიპტოანალიზი ბლოკურ ან ნაკადურ შიფრებში გატეხვის უფრო ეფექტური მეთოდების პოვნას ემსახურება, ვიდრე გასაღებების სიმრავლის უბრალო გადარჩევა, ანუ უხეში ძალის მეთოდია. მაგ. უხეში ძალის მეთოდი DES ალგორითმის წინააღმდეგ მოითხოვს 1 ცნობილ ტექსტს და 2^{56} დეშიფრაციის ოპერაციას. ამავე დროს წრფივი კრიპტოანალიზი მოითხოვს 2^{43} ცნობილ ტექსტს და 2^{43} შიფრაციას, რაც უხეში ძალის მეთოდთან შედარებით წინ გადადგმული ნაბიჯია.

ასიმეტრიული ალგორითმები რომელიმე მათემატიკური პრობლემის გამოთვლით სირთულეს ემყარება. მათ შორის ყველაზე ცნობილია რიცხვის მამრავლებად დაშლის პრობლემა, ასევე დისკრეტული ლოგარითმების პრობლემა. მათი კრიპტოანალიზის მეთოდები მოიცავს რამდენიმე მეტნაკლებად ეფექტურ რიცხვით მეთოდს ამ პრობლემების გადასაჭრელად. მაგ. ელიპსური მრუდეების პრობლემის ამოხსნას საუკეთესო ალგორითმით გაცილებით მეტი დრო ესაჭიროება, ვიდრე ექვივალენტური სირთულის რიცხვით ფაქტორიზებას შესაბამისი

საუკეთესო ალგორითმით. ამიტომ ბოლო ხანებში ელიპსურ მრუდეებზე დაფუძნებულმა კრიპტოსისტემებმა სულ უფრო მეტი პოპულარობა მოიპოვა.

წმინდა კრიპტოანალიზი იყენებს ალგორითმების სისუსტეებს, მაშინ როდესაც შესაძლებელია კრიპტოსისტემაზე შეტევა განხორციელდეს სხვა კუთხიდანაც, მაგ. კრიპტოანალიტიკოსმა შესაძლოა მოიპოვოს ფიზიკური წვდომა შიფრაციის მოწყობილობაზე და მიიღოს დამატებითი ინფორმაცია ანალიზისათვის, მაგ. შიფრაციის ოპერაციისათვის საჭირო დრო, გატარებული ინფორმაციის სტრუქტურა ან შუალედური მონაცემები. და რა თქმა უნდა რჩება სოციალური ინჟინერია, რაც გულისხმობს ინფორმაციის მოპოვებას თვითონ ადამიანებისგან, ვისაც აქვთ წვდომა კრიპტოსისტემაზე, იქნება ეს შანტაჟით, დაშინებით, მოქრთამვით, შპიონაჟით თუ სხვა. შესაძლოა ეს ყველაზე ეფექტური მეთოდი აღმოჩნდეს ყველა დანარჩენ მეთოდთან შედარებით.

ვიჟინერის შიფრი

2.1 პოლიალფაბეტური შფრები

სრულყოფილი სახე პოლიალფაბეტურ შიფრს მისცა იტალიელმა კრიპტოგრაფმა ჯოვანი ბატისტა ბელაზომ. 1553 წელს მან გამოაქვეყნა ახალი შიფრი, რომელიც იყენებს პრაქტიკულად *tabula resta*-ს (ბელაზო არ იცნობდა ტრითემიუსის შრომებს), მაგრამ ამ შიფრს უკვე გააჩნია გასაღები (*ლოზუნგი,* როგორც უწოდა ბელაზომ). თქვენ უნდა აიღოთ რაიმე სიტყვა, ან მოკლე ფრაზა, და მიუწეროთ ის დასაშიფრ ტექსტს ქვეშ მანამდე სანამ სრულად არ შეავსებთ მას. ამის შემდეგ პირველ სტრიქონში აიღოთ დასაშიფრი ტექსტის პირველი ასო, მოძებნოთ პირველ სვეტში გასაღების პირველი ასო და ღია ტექსტის ასოსა და გასაღების ასოს გადაკვეთაზე მდგომი ასო იქნება შიფროტექსტის პირველი ასო. ესაა პოლიალფაბეტური ჩანაცვლების კლასიკური ალგორითმი, ცნობილი როგორც ვიჟენარის შიფრი, რომელიც თითქმის სამასი წლის განმავლობაში სხვადასხვა მოდიფიკაციების სახით გავრცელებული იყო მთელ მსოფლიოში და არავის გაუტეხავს. ეს არ ნიშნავს, რომ ამ ალგორითმით დაშიფრული ცალკეული ტექსტები არ გატეხილა, მაგრამ ზოგადი მეთოდი, თუ როგორ შეიძლება ასეთი შიფრების გატეხვა შეტევით მხოლოდ შიფროგრამის საფუძველზე, არ არსებობდა.

რაც შეეხება სახელს. ცოტა უფრო მოგვიანებით (1583 წელს), ფრანგმა კრიპტოლოგმა ბლეზ ვიჟენარმა, გამოაქვეყნა პრინციპულად განსხვავებული, მაგრამ გარეგნულად ძალიან მსგავსი *ავტოგასაღებიანი შიფრი*, რომელიც ასევე იყენებდა *tabula resta* -ს. ავტოგასაღებიან შიფრში საიდუმლო გასაღებს წარმოადგენს მხოლოდ ერთი ასო, რომლითაც იშიფრება ღია ტექსტის პირველი ასო, ხოლო გასაღების დანარჩენ ასოებად გამოიყენება ღია ან დაშიფრული ტექსტის ასოები. ერთი შეხედვით ეს შიფრი უფრო უკეთესია იმ თვალსაზრისით, რომ თით-ქოს გვაქვს გასაღები, რომლის სიგრძეც ტოლია დასაშიფრი ტექსტის, მაგრამ ის ფაქტი, რომ გასაღების შექმნაში გამოიყენება ღია ან დაშიფრული ღია ტექსტი (რომელსაც გააჩნია თავისი სტრუქტურა), ქმნის გარკვეულ კორელაციას დაშიფრულ და ღია ტექსტებს შორის.

ცხრილი:

ა	ბ	გ	დ	ე	ვ	ზ	თ	ი	კ	ლ	მ	ნ	ო	პ	ჟ	რ
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ს	ტ	უ	ფ	ქ	ღ	ყ	შ	ჩ	ც	ძ	წ	ჭ	ხ	ჯ	ჰ	
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	

2.2 პოლიალფაბეტური შიფრის მათემატიკური წარმოდგენა

ბელაზოს მიერ შექმნილი (და ნებისმიერი მისი მოდიფიკაციაც) პოლიალფაბეტური შიფრი ძალიან მარტივად შეიძლება წარმოვადგინოთ მათემატიკურად იგივე მოდულის გამოყენებით, როგორც ეს გავაკეთეთ მარტივი ჩანაცვლების შიფრებისთვის. ამისათვის კვლავ უნდა ავიღოთ ცხრილი, რომლის საშუალებითაც ასოებს შევუსაბამებთ რიცხვებს განვიხილოთ მარტივი მაგალითი. გვინდა დავშიფროთ ტექსტი: „დღეს კარგი ამინ-დია“ და გასაღებად ავირჩიეთ სიტყვა „შიფრი“. გადავიყვანოთ როგორც ღია ტექსტი, ასევე გასაღები რიცხვებში და რადგანაც გასაღები ტექსტზე პატარაა, გავიმეოროთ ის, სანამ არ შეივსება მთელი ტექსტი:

დ	ღ	ე	ს	კ	ა	რ	გ	ი	ა	მ	ი	ნ	დ	ი	ა
3	22	4	17	9	0	16	2	8	0	11	8	12	3	8	0
შ	ი	ფ	რ	ი	შ	ი	ფ	რ	ი	შ	ი	ფ	რ	ი	შ
24	8	20	16	8	24	8	20	16	8	24	8	20	16	8	24
27	30	24	0	17	24	24	22	24	8	2	16	32	19	16	24
ძ	ხ	შ	ა	ს	შ	შ	ღ	შ	ი	გ	რ	ჰ	უ	რ	შ

ამ პატარა მაგალითიდანაც აშკარაა, რომ პოლიალფაბეტური შიფრების შემთხვევაში ღია ტექსტის სტრუქტურა ერთი ერთზე აღარ გადადის შიფროტექსტში, ამიტომ სიხშირული ანალიზის გამოყენება ამ შემთხვევაში უკვე აღარ მოგვცემს შედეგს, თუმცა უნდა აღვნიშნოთ, რომ რადგანაც გასაღები მოკლეა ვიდრე დასაშიფრი ტექსტი, ეს ალგორითმიც უშვებს ინფორ-მაციის გაჟონვას.

2.3 პოლიალფაბეტური შიფრის კრიპტოანალიზი

როგორც ვხედავთ, პოლიალფაბეტური დაშიფვრის დროს ღია ტექსტის სტრუქტურა ერთიერთზე აღარ გადადის შიფროტექსტში, ამიტომ სიხშირული ანალიზის პირდაპირი გამოყენება არაფერს მოგვცემს. თითქმისსამასი წლის განმავლობაში არ არსებობდა პოლიაფაბეტური შიფრების გატეხვის მეთოდიმხოლოდ შიფროტექსტის საფუძველზე.

1883 წელს პრუსიის არმიის გადამდგარმა მაიორმა ფრიდრიხ კასისკიმ გამოაქვეყნა ასეთი მეთოდი, რომელიც კრიპტოგრაფიაში ცნობილია კასისკის ტესტის სახელით, თუმცა როგორც შემდეგ გაირკვა, პოლიალფაბეტური შიფრები კასისკიზე ადრე გაუტეხია ინგლისელ მათემატიკოსს, პირველი გამომთვლელი მანქანის ავტორს, ჩარლზ ბებიჯს, მაგრამ რადგანაც ის მუშაობდა ამ საკითხზე ინგლისის ადმირალიტეტის დავალებით, მისი შედეგები არსად გამოქვეყნებულა.

კასისკის ტესტის მთავარი იდეის გასაგებად განვიხილოთ მარტივი მაგალითი.

დავუშვათ, ჩვენ გადავწყვიტეთ დავშიფროთ გალაქტიონის ცნობილი სტროფი

ქარი ქრის, ქარი ქრის, ქარი ქრის,

ფოთლებიც მიქრიან ქარდაქარ...

გასაღები იყოს სიტყვა „მალე“, მაშინ დაშიფვრის შედეგად მივიღებთ შიფროტექსტს

ჰანც ჰრტქ, ჰანც ჰრტქ, ჰანც ჰრტქ ჯოსპჯბტს ღიჯფუაღ ჩმორეჰაც...

ჰ	ჰ	ჰ	ჰ	ჰ	ჰ	ჯ	ქ	ღ	უ	მ	ჰ
ა	ა	ა	ა	ა	ა	ო	ბ	ი	ა	რ	ა
ც	ც	ც	ც	ც	ც	ს	ტ	ჯ	ა	ო	ც
ნ	ქ	ნ	ქ	ნ	ქ	პ	ხ	ფ	ჩ	ე	

ასეთი შიფროტექსტის გატეხვა, მიუხედავად იმისა, რომ გამოყენებულია პოლიალფაბეტური მეთოდი მარტივია. აშკარაა, რომ გასაღების სიგრძე ტოლია ოთხის, რაც იმას ნიშნავს, რომ ღია ტექსტის ყოველი მეხუთე ასო დაშიფრულია ერთი და იგივე გასაღებით. ამიტომ თუ ჩვენ დაშიფრულ ტექსტს დავყოფთ ოთხ-ოთხ სიმბოლოდ და ქვეშ მიუწერთ ერთმანეთს, მიღებული თითოეული სვეტი დაშიფრული იქნება ერთი და იგივე გასაღებით მაშინ ჩვენ შეგვიძლია თითოეული სვეტი განვიხილოთ როგორც მარტივი ჩანაცვლების შიფრით დაშიფრული ტექსტი და გავტეხოთ ცალცალკე. აქედან გამომდინარე, პირველ რიგში გვჭირდება მეთოდი, რომელიც საშუალებას მოგვცემს დავადგინოთ გასაღების სიგრძე, რაც რეალურ შემთხვევებში ასე მარტივად არ ხდება. კასისკის ტესტი იძლევა სწორედ გასაღების დადგენის საშუალებას. განვიხილოთ მეორე მაგალითი. ღია ტექსტი იყოს „კრიპტოგრაფია შედგება ორი სიტყვისგან, „კრიპტოს“ და „გრაფია“. ამ წინადადებაში ორჯერ გვხვდება ასოთშეთანხმება „კრიპტო“, რომელთა შორის მანძილი (ასოების რაოდენობა) ტოლია 33-ის. თუ გასაღების სიგრძე იქნება ოცდაცამეტის გამყოფი, მაგალითად „ჩვენ ეს ვიცით“, მაშინ ეს ერთნაირი ასოთშეთანხმებები ასევე ერთნაირად გადავა დაშიფრულ ტექსტშიც. სწორედ ამ თვისებას იყენებს კასისკის ტესტიც, რომელიც მოითხოვს, რომ დაშიფრულ ტექსტში აღმოვაჩინოთ ერთნაირი ბიგრამები, ტრიგრამები და უფრო დიდი ასოთშეთანხმებები და დავადგინოთ მათ შორის მანძილები. შემდეგ მოვძებნოთ ამ რიცხვების უდიდესი საერთო გამყოფი, რომელიც იქნება გასაღების სიგრძე. დავადგენთ რა გასაღების სიგრძეს, დაშიფრული ტექსტი უნდა დავყოთ ამ სიგრძის ნაწილებად და მივუწეროთ ერთმანეთს ქვეშ. მაშინ მიღებულ მატრიცაში თითოეული სვეტი დაშიფრული იქნება მარტივი ჩანაცვლების მეთოდით და თითოეული სვეტისთვის ცალ-ცალკე უნდა გამოვიყენოთ სიხშირული ანალიზი.

2.4 კასისკიტ ტესტის გამოყენების მაგალითი

განვიხილოთ მაგალითი. ვნახოთ, როგორ უნდა გამოვიყენოთ კასისკის ტესტი. დავუშვათ, მოცემული გვაქვს დაშიფრული ტექსტი.

გნერჯჭნყგშწკწრფჯძმფფვივჰრეცწშჯეჰშწრიჭჩფჯდრეძპწჟზწიქიჩაითმფრლძნქურძიეოოჰჩფა
ითმფგდპნქურკშთშისმშსრშფდრრსრფწჟლაშძლლტმშშბჩაძიცშფძჩშიგშჩშჭფწლშშწთრშშმფრფ
ციდშჰიეჭქრშფწჰქძრეტინბშშრშაშძრიჭჩფჯდრეძპწჟზწიქიჩშხშშწსრშჟწყქდშფრჭჭჩდჭუგნჯრ

შრრჰჭჩთიუძქირფჰქგუშიიგიოქიიშფთნლგხშითქჯძნციჯრგბრშჰქიმფყოჰჟაწშშჩფონბლფწჭნ უზცრშშაძუწლფძქგინწშეჩყრძსვაეოწჯჰშჩგპშყძდისროჰსწრუშნქშჩპოშციმშწაჰწუპცნჩნხბშიშრ დშჩჯოწაოწცდდრგიერრთიყძრპოფჰჩშკფრუჩნყთრრიფტმწჟნფუჰჭჩფჯდრეძპწჟზწიქიჩშბძიჯ რჰჰფჰრვშოძყძჭრეძნძიღრბშმაყუჰჩწბშმქქრმშჰზაიმძპქტეჯადრბიერჯჭნყგშჰმწრვვივჰრეცწქი თოლჭწჰტჩრშძიჩშჭდნქშ

დავიწყოთ ერთნაირი ბიგრამების და უფრო დიდი ნაწილების მოძებნა ამ ტექსტში.

გნერჯჭნყგშწკწრფჯძმფფვივჰრეცწშჯეჰშწრიჭჩფჯდრეძპწჟზწიქიჩაითმფრლძნქურძიეოოჰჩფა ითმფგდპნქურკშთშისმშსრშფდრრსრფწჟლაშძლლტმშშბჩაძიცშფძჩშიგშჩშჭფწლშშწთრშმფრფ ციდშჰიეჭქრშფწჰქძრეტინბშშრშაშძრიჭჩფჯდრეძპწჟზწიქიჩშხშშწსრშჟწყქდშფრჭჭჩდჭუგნჯრ შრრჰჭჩთიუძქირფჰქგუშიიგიოქიიშფთნლგხშითქჯძნციჯრგბრშჰქიმფყოჰჟაწშშჩფონბლფწჭნ უზცრშშაძუწლფძქგინწშეჩყრძსვაეოწჯჰშჩგპშყძდისროჰსწრუშნქშჩპოშციმშწაჰწუპცნჩნხბშიშრ დშჩჯოწაოწცდდრგიერრთიყძრპოფჰჩშკფრუჩნყთრრიფტმწჟნფუჰჭჩფჯდრეძპწჟზწიქიჩშბძიჯ რჰჰფჰრვშოძყძჭრეძნძიღრბშმაყუჰჩწბშმქქრმშჰზაიმძპქტეჯადრბიერჯჭნყგშჰმწრვვივჰრეცწქი თოლჭწჰტჩრშძიჩშჭდნქშ

	1	2	3	4	5		1	2	3	4	5		1	2	3	4	5
ა		2	6	5		მ	3	7	2		1	ღ			1		
ბ	4	1		1	2	ნ	2	10	2		5	ყ			5	5	
გ	7	1		3	1	ო		5	1	1	5	შ	17	8	9	15	8
დ	6		6	3	1	პ	3	1	2		3	ჩ	2	6	8		7
ე	5	1	9	1		ჟ		6			2	ც	3	3		3	
ვ	2	1	2		2	რ	2	10		19	18	ძ	6	2	1	13	3
ზ			5			ს	ს	3	1	1	2	წ	9		12	8	1
თ	6		1	3		ტ		2		2	1	ჭ	4	4	1	3	2
ი	6	20		1	15	უ		3	1	2	7	ხ			2		1
კ	1	2				ფ		3	17	5	4	ჯ	2	2	4		6
ლ	3	2	1		3	ქ	3	2	5	5	6	ჰ	8	2	3	6	3

როგორც ვხედავთ ასეთი ნაწილები ძალიან ბევრია. თუ დავითვლით მათ შორის მანძილებს, ძალიან ადვილად აღმოვაჩენთ, რომ ამ მანძილების უდიდესი საერთო გამყოფია ხუთი. ამგვარად, ჩვენ აღმოვაჩინეთ გასაღების სიგრძე. ახლა დავყოთ დაშიფრული ტექსტი ხუთის ტოლ სტრიქონებად და მივუწეროთ ერთმანეთს ქვეშ. თითოეული სვეტი დაშიფრული იქნება ერთი და იგივე ასოთი და სიხშირული ანალიზის მეთოდის გამოყენებით ჩვენ შევძლებთ

ვიპოვოთ გასაღები. ცხრილში 3.3. მოყვანილია თითოეულ სვეტში ასოების სიხშირეები.
ამ შედეგებიდან გამომდინარე აშკარაა რომ გასაღების პირველი ოთხი ასოა „შიფრ“. რაც შეეხება მეხუთე ასოს, აქ მოხდა მცირე შეცდომა, რაც ალბათ იმის ბრალია, რომ დაშიფრული ტექსტის ზომა არ იყო საკმარისი. თუ გამოვიყენებთ შიფროგრამის გასაშიფრად გასაღებს „შიფრი“, მივიღებთ ღია ტექსტს:

მესაყვედურებიან: „თუ შენ პატრიოტი იყო, შენის ქვეყნის მზეთუნახავს არ დააგდებდი და სხვისას არ დაუწყებდი ტრფიალსო...“ მაგრამ სცდებიან - რა იციან, თუ ჩემ გულში რა არის? აქ ანგარიშია და ანგარიშს ჭკვიანი როდის გაქცევია? ჩემის ქვეყნის მზეთუნახავი ღარიბია! თითონ რა აქვს, რომ მე მარჩინოს? რა ჰმოსია, რომ მე ჩამაცვას? რა შეუძლია რომ მე გამაძლიეროს და თვითონ რა სახელგანთქმულია, რომ მე გამომაჩინოს? და უცხო ქვეყნის მზეთუნახავი კი ამაების ყველაფრის შემძლებელია! აი რისთვის ვეტრფი მას! აი რად მიყვარს!
აბა ამ ბედში ჩააგდეთ ჩემი ქვეყნის მზეთუნახავიც და მაშინ ნახავთ, თუ ვისმე დაგაცადოთ მისი ტრფობა!! მაშ უსაფუძვლო ყოფილა საყვედური! დიახ, პატრიოტი ვარ, ვყოფილვარ და ვიქნები!..
(აკაკი წერეთელი „პატრიოტის აღსარება“)

გენეტიკური ალგორითმები

ძებნის რთული ამოცანების ამოხსნისათვის გენეტიკური ალგორითმი არის საუცხოო საშუალება. ისინი ხშირად გამოიყენებიან ისეთ დარგებში როგორიცაა ტექნიკა, რომ შევქმნათ დაუჯერებლად მაღალი ხარისხის პროდუქტები. მაგალითად, მათ აქვთ შესაძლებლობა მოძებნონ მასალებისა და დიზაინების დიდი რაოდენობა განსხვავებულ კომბინაციებს შორის, რომ იპოვნონ უკეთესი კომბინაცია, რომელიც შედეგად გვაძლევს მაგარ, მსუბუქ და ყოველმხრივ უკეთეს პროდუქტს. ისინი შეიძლება აგრეთვე იყვნენ გამოყენებულნი კომპიუტერული ალგორითმების შესაქმნელად, განრიგის ამოცანებში და სხვა ოპტიმიზაციის საკითხებში.

გენეტიკური ალგორითმები არიან დაფუძნებულნი ბიოლოგიური ევოლუციის პრინციპზე ბუნებრივი შერჩევის გზით, რასაც ვხვდებით ბუნებაზე დაკვირვების დროს. ისინი

არსებითად იმეორებენ ხერხს, რომელსაც ცხოვრება იყენებს რეალური სამყაროს პრობლემების გადაწყვეტის საპოვნელად. მიუხედავად იმისა, რომ გენეტიკური ალგორითმები შეიძლება გამოყენებულნი იყვნენ დაუჯერებლად რთული პროლემების ამოსახსნელად, მოულოდნელად ისინი არიან ძალზე მარტივნი გამოსაყენებლად და გასაგებად.

გენეტიკური ალგორითმების გამოგონების მიზანი იყო, რომ მოეხდინათ გარკვეული ბუნებრივი ევოლუციის პროცესების იმიტირება. ბევრი მეცნიერი, მათ შორის ბიოლოგები, არიან გაოცებულნი, რადგან ცხოვრების დონის სირთულე რომლის წინაშეც ვდგევართ, შეიძლება განვითარდეს შემოთავაზებული ნამარხი ნაშთების ჩანაწერებით. გენეტიკური ალგორითმების იდეაა, რომ გამოიყენოს ევოლუციის ეს ძალა ოპტიმიზაციის ამოცანების ამოსახსნელად.

გენეტიკური ალგორითმების იდეა გამოთქვა ჯ. ჰოლანდმა (John Holland), XX საუკუნის სამოცდაათიან წლებში. ჯ. ჰოლანდი დაინტერესებული იყო ცოცხალი და ხელოვნური სისტემების ბუნებრივ გარემოსთან ადაპტაციის საკითხებით. ევოლუციის პროცესში ქრომოსომები განიცდიან ევოლუციას და არა თვითონ ცოცხალი ორგანიზმები.

3.1 რა არის გენეტიკური ალგორითმები?

გენეტიკური ალგორითმები არის ადაპტირებული ევრისტიკული ძებნის ალგორითმები დაფუძნებული ბუნებრივი სელექციისა და გენეტიკის ევოლუციურ იდეებზე. როგორც ასეთი, ისინი წარმოადგენენ შემთხვევითი ძებნის ინტელექტუალურ ექსპლუატაციას ოპტიმიზაციის ამოცანების ამოსახსნელად. გენეტიკური ალგორითმების საბაზისო ტექნიკა მოწყობილია ისე, რომ ახდენს ევოლუციისთვის საჭირო პროცესების სიმულაციას ბუნებრივ სისტემებში, სპეციალურად იმ პრინციპებისა, რომლებსაც პირველად საფუძველი ჩაუყარა ჩარლს დარვინმა (Charles Darwin) და რომელსაც ეწონდებოდა "ბუნებრივი გადარჩევის" პრინციპები (გადარჩევის გზით უძლიერესის გამრავლების პრინციპებადაც მოიხსენიებენ). რეალურ სამყაროში ეს არის იმის ანალოგი, როცა ძლიერები დომინირებენ სუსტებზე.

3.2 რატომ გენეტიკური ალგორითები?

ეს არის უკეთესი ვიდრე პირობითი ხელოვნური ინტელექტი. ძველი ხელოვნური ინტელექტისგან განსხვავებით, გენეტიკური ალგორითმები არ ვარდება (არ განიცდის კრახს) თუ შესატანი მნიშვნელობები უმნიშვნელოდ შეიცვლება, ან იქნება გონივრული (მცირე) ხმაური

(მათემატიკურად შეშფოთება/ცვლილება). ასევე დიდ სივრცეში ძებნისას, მულტიმოდალურ სივრცეში ძებნისას, ან N განზომილებიანი ზედაპირის შემთხვევაში გენეტიკური ალგორითმები მნიშვნელოვან უპირატესობას ფლობს ტიპური ძებნისა და ოპტიმიზირებულ ტექნიკებთან მიმართებაში (წრფივი პროგრამირება, სიღრმეში ძებნა, სიგანეში ძებნა).

გენეტიკურ ალგორითმებში გამოყენებულია ყველაზე უფრო კარგად შეგუებული ინდივიდების გადარჩენის ევოლუციური პრინციპი. ტრადიციული ოპტიმიზაციის მეთოდებისაგან განსხვავებით GA იყენებს: 1) კოდირებულ პარამეტრებს და 2) ამონახსნის საძიებლად არა ერთ საწყის წერტილს, არამედ საწყის პოპულაციას; 3) მიზნის ფუნქციას მხოლოდ და არა მის წარმოებულს ან რაიმე დამატებით ინფორმაციას; 4) ამორჩევის ალბათურ წესს და არა დეტერმინირებულს. ჩამოთვლილი ოთხი თვისება განაპირობებს GA-ს ეფექტურობას სხვა დანარჩენ ტექნოლოგიებთან.

მიუხედავად უპირატესობებისა აქვე უნდა აღინიშნოს ისიც, რომ გენეტიკური ალგორითმების მიერ მოცემული ამონახსნი არ არის ზუსტი.

3.3 ცნებები და განმარტებები

პოპულაცია – ინდივიდების სასრული სიმრავლე;

ინდივიდები – კოდირებული ქრომოსომები ანუ ამოცანის პარამეტრები – ამონახსნთა სივრცის წერტილები;

ქრომოსომა – კოდური მიმდევრობები ანუ ჯაჭვები, გენების მოწესრიგებული მიმდევრობა;

გენი – ქრომოსომის ატომარული ელემენტი ანუ გენოტიპის ელემენტი;

გენოტიპი – მოცემული ინდივიდის ქრომოსომების ერთობლიობა;

ფენოტიპი – დეკოდირებული სტრუქტურა ანუ ამოცანის პარამეტრების სიმრავლე;

ალელი – კონკრეტული გენის მნიშვნელობა;

ლოკუსი – ქრომოსომაში კონკრეტული გენის პოზიცია;

რეპროდუქცია – აღნიშნავს ახალი ქრომოსომების შექმნას მშობლების გენების რეკომბინაციის შედეგად.

რეკომბინაცია – ამ პროცესის შედეგად მიიღება გენების ახალი კომბინაციები, რისთვისაც გამოიყენება ორი ძირითადი ოპერატორი (ოპერატორი) – შეუღლება და მუტაცია.

GA-ს ერთ-ერთი ძირითადი ცნებაა შეგუებადობის ფუნქციის ცნება (Fitness Function), ანუ შეფასების ფუნქცია, რომელიც გვიჩვენებს პოპულაციაში მოცემული ინდივიდის შეგუებადობის ზომას. ამ ფუნქციის საშუალებით ამოირჩევენ პოპულაციიდან უფრო მეტად შეგუებულ ინდივიდებს, რაც ევოლუციის პრინციპიდან გამომდინარეობს. გადარჩება ის ინდივიდი, რომელსაც მაღალი შეგუებადობის ფუნქციის მაჩვენებელი გააჩნია, ანუ "ძლიერის" გადარჩენის პრინციპი. ეს ფუნქცია უმნიშვნელოვანესია GA ფუნქციონირების პროცესში. ოპტიმიზაციის ამოცანებში ამ ფუნქციას მიზნის ფუნქციას უწოდებენ და ხდება მისი ოპტიმიზაცია (უფრო ზუსტად, მაქსიმიზაცია), მართვის თეორიაში შეიძლება იყოს ცდომილების ფუნქცია, ხოლო თამაშთა თეორიაში – ფასის ფუნქცია. GA-ს ფუნქციონირებისას ყოველ იტერაციაზე თითოეული ინდივიდი ფასდება შეგუებადობის ფუნქციის საშუალებით და მის მიხედვით მიიღება ახალი პოპულაცია შემდგომი თაობისათვის.

3.4 ძირითადი პროცესები

1. **ინიციალიზაცია** - საწყისი პოპულაციის(დასახლების) შექმნა. ეს პოპულაცია ჩვეულებრივ გენერირებულია შემთხვევითად და შეიძლება იყოს ნებისმიერი ზომის, ცოტაოდენი რაოდენობა აღებული ათასებიდან.
2. **შეფასება** - შემდეგ პოპულაციის ყოველი წევრისათვის ვაფასებთ და ვითვლით მის ვარგისიანობას. ვარგისიანობის მნიშვნელობა გამოითვლება, თუ რამდენად კარგია ეს წევრი, ჩვენთვის სასურველი მოთხოვნებისათვის. ეს მოთხოვნები უნდა იყვნენ მარტივნი, უფრო სწრაფი ალგორითმები არიან უკეთესნი, ან უფრო კომპლექსურად, მაგარი მასალები არიან უკეთესნი, მაგრამ ისინი არ უნდა იყვნენ ძალზე მძიმე.
3. **შერჩევა** - ჩვენ გვჭირდება მუდმივად გავაუმჯობესოთ ჩვენი პოპულაციების მთლიანად ვარგისიანობა. შერჩევა გვეხმარება ჩვენ ცუდი დიზაინების მოცილებაში და პოპულაციის მხოლოდ უკეთესი ელემენტების დატოვებაში. არსებობენ მხოლოდ ცოტაოდენი მეთოდებისა, მაგრამ ძირითადი იდეაა, ვარგისი ელემენტები იყვნენ შერჩეულნი ჩვენი შემდეგი გენერაციისათვის.

4. **შეჯვარება** - შეჯვარების დროს ჩვენ ვქმნით ახალ ელემენტებს შერჩეული ელემენტების სახეების შეერთებით. ჩვენ შეგვიძლია მივბაძოთ იმაში, თუ როგორ მოშაობს სექსი ბუნებაში. ჩვენ ვიმედოვნებთ, რომ ერთი ან მეტი ელემენტების ზოგიერთი ნიშნების კომბინირებით, მივიღებთ მემკვიდრეობით მათი მშობლების უკეთეს ნიშნებს.
5. **მუტაცია** - ჩვენ გვჭირდება დავამატოთ ცოტაოდენი შემთხვევითობა ჩვენს პოპულაციურ გენეტიკაში. სხვა სიტყვებით, რომ ვთქვათ, ამოხსნების ყოველი კომბინაცია, რომელიც ჩვენ შეგვიძლია შევქმნათ, სასურველია იყოს ჩვენ საწყის პოპულაციაში. ტიპიურად მუტაცია მუშაობს ძალზე მცირე ცვლილებების გაკეთებით შემთხვევითად ელემენტების გენებში.
6. **და გავიმეოროთ** - ახლა ჩვენ გვაქვს შემდეგი გენერაცია, რომლისგანაც შეგვიძლია დავიწყოთ ახალი ნაბიჯი, სანამ არ მივაღწევთ დამთავრების პირობას.

დასრულება - არსებობს რამოდენიმე მიზეზი, რომ დავამთავროთ ჩვენი გენეტიკური ალგორითმი. ყველაზე სასურველი მიზეზია ის, რომ ჩვენმა ალგორითმმა იპოვა ამონახსნი, რომელიც საკმარისად კარგია და აკმაყოფილებს წინასწარ განსაზღვრულ მინიმალურ კრიტერიუმს. სხვა მიზეზები შეიძლება იყოს შეზღუდვები, როგორიცაა დრო და ფული.

3.5 გენეტიკური ოპერატორები

სელექციის შედეგად ამორჩეულ ქრომოსომების პოპულაციაში გამოიყენება GA-ს ორი ძირითადი გენეტიკური ოპერატორი: შეუღლების - Crossover ოპერატორი და მუტაციის - Mutation ოპერატორი. მუტაციის ოპერატორი შეუღლების ოპერატორთან შედარებით მეორეხარისხოვან როლს ასრულებს. კლასიკურ GA-ში შეუღლება ყოველთვის სრულდება $0,5 \leq Pc \leq 1$, ხოლო მუტაცია კი ძალიან იშვიათად $0 \leq Pm \leq 0,1$. მუტაცია შეიძლება მოხდეს ან შეუღლებამდე ან შეუღლების შემდეგ.

პირველ ეტაპზე მშობლების პოპულაციიდან ამოირჩევა ორი ქრომოსომა. ამ ეტაპზე ხდება ამ ქრომოსომების შეწყვილება შემთხვევით Pc ალბათობით. შემდეგ ისევ შემთხვევით აიღებენ შეუღლების წერტილს, თუ ქრომოსომა შედგება N-გენისაგან (ე.ი. ქრომოსომის სიგრძეა n), მაშინ ლკ-შეუღლების წერტილი არის ნატურალური რიცხვი ნაკლები N-ზე, ანუ n_k წერტილი შემთხვევით აირჩევა[1, N-1] ინტერვალიდან. შეუღლების შედეგად მიიღება შთამომავლების წყვილი.

1) შთამომავალის ქრომოსომები 1-დან n-მდე შედგება პირველი მშობლის გენებისაგან,
ხოლო n_k+1 პოზიციიდან L-მდე – მეორე მშობლის გენებისაგან.
2) შთამომავალის ქრომოსომები 1-დან n_k-მდე შედგება მეორე მშობლის გენებისაგან,
ხოლო n_k+1 პოზიციიდან L-მდე _ პირველი მშობლის გენებისაგან.
მუტაციის ოპერატორი Pm ალბათობით იცვლება ქრომოსომაში გენის
მნიშვნელობა მისი საპირისპიროთი. ანუ 0 იცვლება 1-ით და პირიქით.

Selection:

- მთავარი იდეა: მიეცი უპირატესობა უკეთეს ინდივიდს, რომლის გადაცემაც შეუძლია მომავალი თაობისთვის.
- ინდივიდის სიკარგე დამოკიდებულია შეგუებადობის კოეფიციენტზე.
- შეგუებადობის კოეფიციენტი შეიძლება გამოითვალოს რაიმე ობიექტური ფუნქციით ან სუბიექტური განსჯით.

Crossover:

- ძირითადი გამორჩეული ფაქტორი გენეტიკური ალგორითმებისა სხვა სხვა ოპტიმიზაციის ტექნიკებისგან.
- ორი ინდივიდი აირჩევა პოპულაციიდან selection-ის მეთოდით.
- ე.წ. გაწყვეტის წერტილი crossover-თვის უნდა აირჩეს შემთხვევითად.
- ორი ინდივიდის მნიშვნელობები იცვლება გაწყვეტის წერტილის მიხედვით.
- თუ S_1 ინდივიდი არის 000000 და S_2 ინდივიდი არის 111111 და გაწყვეტის წერტილი არის 3 მაშინ შთამომავლები S_1'=111000 და S2'=000111
- ორი ახალი ინდივიდი, რომელიც მივიღეთ წინა თაობის შეჯვარებით ჯდება უკვე ახალი თაობის პოპულაციაში
- „კარგი“ ინდივიდების ნაწილების რეკომბინირებით, ეს პროცესი მიზნად ისახავს უკეთესი ინდივიდების შექმნას.

Mutation:

- მცირე ალბათობით, ახალი ინდივიდების ნაწილი ბიტებისა უბრალოდ გადანაცვლდება.

- მისი მიზანია შეინარჩუნოს მრავალფეროვნება პოპულაციის ფარგლებში და შეაჩეროს ნაადრევი დაახლოება.
- მუტაცია მარტო იწვევს შემთხვევით გავლას ძებნის სივრცეში.
- მუტაცია და სელექცია (crossover-ის გარეშე) ქმნის პარალელურ, ხმაურით-ტოლერანტულ (noise-tolerant), მთაზე ასვლის (hill-climbing) ალგორითმს.

3.6 ახალი პოპულაციის ფორმირება

გენეტიკური ოპერატორების მოქმედების შედეგად მიღებული ქრომოსომები გაერთიანდებიან ახალ პოპულაციაში, მას უკვე ვუწოდებთ მიმდინარე პოპულაციას მოცემული იტერაციისათვის. ყოველ შემდგომ იტერაციაზე გამოითვლება ქრომოსომების შეგუებადობის ფუნქცია და მოწმდება ალგორითმის გაჩერების პირობა ან გადავდივართ ალგორითმის შესრულების შემდეგ ეტაპზე – სელექციის ეტაპზე. კლასიკურ GA-ში ყოველთვის ამორჩეული ქრომოსომების პოპულაცია მეორე ეტაპზე იცვლება ახალი პოპულაციით, ხოლო ქრომოსომების რიცხვი ყველა პოპულაციაში თანაბარია.

ფუნქციონალი

4.1 პროგრამული უზრუნველყოფის შესახებ

პროგრამული უზრუნველყოფა შექმნილია ობიექტზე ორიენტირებული მიდგომით, C# პროგრამირების ენაზე, Microsoft .NET პლათფორმისთვის. ამ მიდგომას ძალიან ბევრი დადებითი მხარე აქვს. ერთ-ერთი დიდი პლიუსად შეგვიძლია მივიჩნიოთ პროგრამული კოდის სიმცირე, რაგან ობიექტზე ორიენტირებული მიდგომა გვაძლევს იმის საშუალებას, რომ თავიდან ავიცილოთ მოქმედებების დუბლირებები. თუმცა კრიტიკული სისტემებისთვის, ან სისტემების კრიტიკული კომპონენტებისთვის ხანდახან საჭიროა დუბლირებების სპეციალურად შემოტანაც კი. ამ შემთხვევაში საქმე გვაქვს პროგრამული უზრუნველყოფის მიმართ წაყენებულ ისეთ

მოთხოვნასთან, როგორიცაა სანდოობა. რეკომენდირებულია კრიტიკული კომპონენტების დუბლირება - ერთის ავარიის შემთხვევაში, მეორე ამუშავდება. ასევე შესაძლებელია მრავალფეროვნების შემოტანაც, რაც გულისხმობს იმას, რომ მოხდეს ერთი და იგივე ფუნქციონალის სხვადასხვა გზით უზრუნველყოფა. მრავალფეროვნების ერთ-ერთი მაგალითია სხვადასხვა ოპერაციული სისტემის მქონე სერვერების გამოყენება. რაც შეეხება სიჭარბეს კომპანიებმა შეიძლება გამოიყენონ სარეზერვო რესურსები, რომლებზეც გადართვა ავტომატურად მოხდება პროგრამული ან ტექნიკური ავარიის შემთხვევაში.

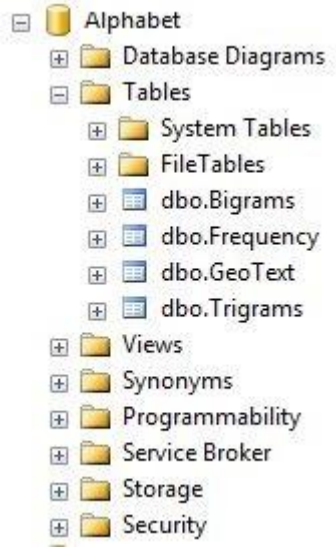
4.2 რესურსები

პროგრამა იყენებს მონაცემთა ბაზას MS SQL Server.

ბაზას გააჩნია შემდეგი სტრუქტურა:

ბაზის სახელია Alphabet, გვაქვს 4 ცხრილი

Frequrncy, Bugrams, Trigrams და GeoText



1. ქართული ასო-ნიშნების სიხშირეების ცხრილი არის Frequrncy-ცხრილი, აქ გვაქვს ქართული სიმბოლოები და მათი სიხშირეების მაჩვენებლები, თუ რომელი ასო-ნიშანი საშუალოდ რა სიხშირით გვხვდება ქართულ ტექსტებში. ამის საშვალებით შეიძლება დაშიფრული სიმბოლოს სავარაუდო მნიშვნელობის გაგება. მაგალითად „ი“-ს სიხშირული მაჩვენებელია 9.776332, თუ დაშიფრულ ტექსტში ამ სიხშირით გვაქვს სხვა სიმბოლო, მაგალითას „ლ“ მივხვდებით რომ ნამდვილ ტექსტში „ლ“-ს მაგივრად იქნებოდა „ი“;

ცხრილის არასრული ნაწილი:

ID	Symbol	SymbolFrequency
0	ა	13.374286
1	ბ	3.119185
2	გ	1.943555
3	დ	3.937019
4	ე	7.611710
5	ვ	3.193002
6	ზ	0.640396
7	თ	2.539385
8	ი	9.776332
9	კ	1.122305
10	ლ	3.381678

…

…

2. ქართულ ენაში არსებული ბიგრამების და ტრიგრამების ბაზა, მათი სიხშირული მაჩვენებლებით. ეს სიხშირული მნიშვნელობები გაოიყენება იმავე სახით როგორც ზემოთ თითოეული ასო-ნიშნების სიხშირეებია, მაგრამ შედეგი უფრო ზუსტია. ეს ორი ცხრილი ჩვენს პროგრამაში ამჟამად არ გამოიყენება მაგრამ განვითარების პერსპექტივაში გააუმჯობესებს ძებნის შედეგს;

ცხრილების არასრული ნაწილი:

ID	BigramCombination	Frequency
1	აა	0.246222
2	აბ	0.200975
3	აგ	0.335230
4	ად	1.037713
5	აე	0.089008
6	ავ	0.893948
7	აზ	0.270220
8	ათ	0.370522
9	აი	0.306403
10	აკ	0.279359

ID	TrigramCombination	Frequency
11915	ები	0.756361
14765	ლებ	0.452311
11907	ება	0.444668
25217	ბის	0.418411
3438	რომ	0.342151
32508	ული	0.304750
3131	რებ	0.303000
14913	ლის	0.294131
12212	ელი	0.283861
25831	ვის	0.255971

3. ყველაზე ხშირად გამოყენებული, სიტყვების ბაზა, ის წარმოდგენილია GeoText ცხრილში

ამ ცხრილში არის დაახლოებით 300000(სამასი ათასი) სიტყვა ქართული ტექსტებიდან, ეს მონაცემები გამოიყენება გაშიფრულ ტექსტში ქართული სიტყვების ძებნისთვის და იმის შესაფასებლად თუ რამდენად ჰგავს ეს ტექსტი ქართულ ლიტერატურულს (ეს ერთ-ერთი მთავარი ფუნქციაა ამ პროგრამისთვის).

	ID	Text	LenText
1	123420	მოუმატეს	8
2	113292	მომგვრიდა	9
3	14121	ნაბერწკალი	10
4	185406	დაუსახავადაც	12
5	73883	უღვთონი	7
6	93921	მოგითხარ	8
7	39150	დავიჩენ	7
8	96733	კონისთვის	9
9	229909	შირვანზე	8
10	75667	ეტანებოდნენ	11

ამ ცხრილში წინებისგან განსხვავებით გვაქვს სვეტი-LenText, რომლის მნიშვნელობაც Text-სვეტში მყოფი სიტყვის სიგრძეა, LenText-სვეტის ერთადერთი დანიშნულება ძებნისა და დალაგების ფუნქციების ასწრაფებაა;

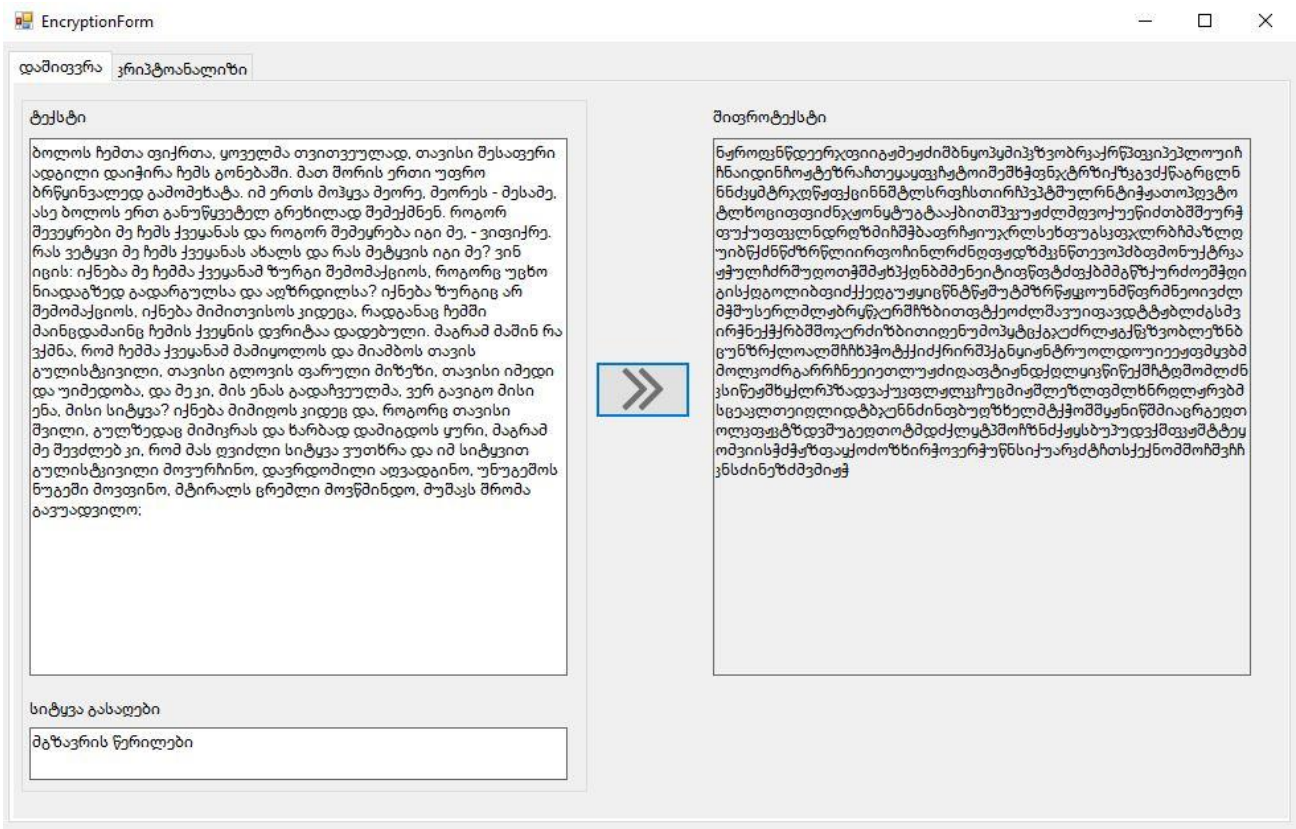
4.3 პროგრამული უზრუნველყოფის მოვალეობები

პროგრამა ახდენს ქართული ტექსტის დაშიფვრას ვიჟინერის ალგორითმით, გასაღების სიგრძის პოვნას კასისკის ტესტით და დასაშიფრი გასაღების პოვნას გენეტიკური ალგორითმებით, ასევე გვაქვს სტატისტიკა, სადაც მოცემულია განვითარების თითოეული ეტაპი

4.3.1 ქართული ტექსტის დაშიფვრა(ვიჟინერის ალგორითმი).

დაშიფვრია სიტყვა-გასაღების დახმარებით. ამ მეთოდის მოქმედების პრინციპი ზემოთაა აღწერილი (მე-2 თავში).

ვიზუალურად კი ასე გამოიყურება:



4.3.2 დაშიფრულ ტექსტში სიტყვა-გასაღების სავარაუდო ზომის დადგენა კასისკის ტესტის საშვალებით;

ეს მეთოდის თეორიული აღწერა ზემოთაა მოცემული (2.3 და 2.4 თავებში), აქ კი ვაჩვენებ პრაქტიკულად C# ენაზე დაწერილი მეთოდების სახით.

ძირითადი იდეა მდგომარეობს იმაშ რომ, დაშიფრულ ტექსტში ვეძებთ ტრიგრამებს(სამ სიმბოლოიან მიმდევრობებს) რომლებიც ტექსტში რამდენჯერმე გვხვდება.

```
string text = CriptoAnalise_textBox.Text.ToString();

List<int> repeatCount = new List<int>(); //მასივი, რომელიც შეიცავს ყველა სიგრძეს მსგავს სიმბოლოებს შორის
//ამ მასივის შევსება, მასში ტრიგრამების ძებნა (რადგან DigramLength=3)
for (int i = 0; i < text.Length - DigramLength + 1; i++)
{
    string temp1 = text.Substring(i, DigramLength);
    for (int j = i + 1; j < text.Length - DigramLength + 1; j++)
    {
        string temp2 = text.Substring(j, DigramLength);
        if (temp1.Equals(temp2))
        {
            repeatCount.Add(j - i);
        }
    }
}
```

`string text` იღებს დაშიფრული ტექსტის მნიშვნელობებს, `List<int> repeatCount` ამ ლისტში იწერება ყველა იმ მანძილის მნიშვნელობა, რომელიც კი გვხვდება ტექსტში, ტრიგრამებს შორის.

```
int[] usgs = new int[maxLength + 1]; // მასივი უსგ-ს ძებნისთვის
// თუ ორ მანძილის მახასიათებელ რიცხვს შორი უსგ k-ს ტოლია, მაშინ usgs[k] გავზარდოთ 1-ით
for (int i = 0; i < repeatCount.Count; ++i)
    for (int j = i + 1; j < repeatCount.Count; ++j)
        usgs[CryptoMethods.USG(repeatCount[i], repeatCount[j])]++;
usgs[0] = 0;
```

კოდის ეს ნაწილი `int[] usgs` მასივში იმ ინდექსის მნიშვნელობას, რომელიც ტოლია რომელიმე ორ ტრიგრამს შრის მანძილის უდუდესი საერთო გამყოფის, ზრდის ერთით.

ეს საშვალებას იძლევა, გავიგოთ, ყველაზე ხშირი უსგ-ს მნიშვნელობა, დიდი ალბათობით სწორედ ეს მნიშვნელობაა გასაღების ზომა.

```
// გადავიტანთ ყველაფერს ახალ მასივში, რათა მარტივად დავახარისხოთ
List<Pair> ans = new List<Pair>();
for (int i = 2; i < 500; ++i)
{
    ans.Add(new Pair()
    {
        Index = i,
        Value = usgs[i]
    });
}
IEnumerable<Pair> topAns = ans.OrderByDescending(p => p.Value).Take(10);//ვალაგებთ
```

შემდეგ გადავწერთ ახალ მასივში, დავალაგებთ კლების მიხედვით და ვიღებთ პირველ 10 ჩანაწერს.

ორ რიცხვს შორის უდიდესი საერთო გამყოფი კი ასე გამოითვლება:

```
// *ორ რიცხვს შორის, უდიდესი საერთო აგმყოფის პოვნა*
2 references
public static int USG(int a, int b)
{
    if (b == 0)
    {
        return a;
    }
    else
    {
        return USG(b, a % b);
    }
}
```

4.3.3 საუკეთესო, შესაძლო, სიტყვა-გასაღების პოვნა გენეტიკური ალგორითმების დახმარებით;

```
public class Chromosome
{
    18 references
    public string ChromosomeValue { get; set; }
    8 references
    public int FitnessValue { get; set; }
```

გვაქვს კლასი `class Chromosome` რომელიც შეიცავს ორ ველს `string ChromosomeValue` და `int FitnessValue`, პირველში გვაქვს სიტყვა-გასაღების სიგრძის ზომის შემთხვევითი, ქართული სიმპოლოებისგან შედგენილი სიტყვა, რომელსაც თავიდან არანაირი გრამატიკული მნიშვნელობა არ გააჩნია

```
public static class Symbols
{
    public static Symbol[] GeoSymbols = new Symbol[]
    {
        new Symbol(0, "ა"),
        new Symbol(1, "ბ"),
        new Symbol(2, "გ"),
        new Symbol(3, "დ"),
        new Symbol(4, "ე"),
        new Symbol(5, "ვ"),
        new Symbol(6, "ზ"),
        new Symbol(7, "თ"),
        new Symbol(8, "ი"),
        new Symbol(9, "კ"),
        new Symbol(10, "ლ"),
        new Symbol(11, "მ")
```

სიმბოლოებს ვიღებთ ამ კლასიდან, სადაც გვაქვს მოცემული 33 ქართული ასო და მათ შეესაბამება 0 დან 32-მდე რიცხვები, სწორედ ეს მნიშვნელობები გამოიყენება შიფრაცია-დეშიფრაციისას, როცა ერთ სიმბოლოს ვუმატებთ მეორეს, მაგალითად ბ+გ ⇔ 1+2 = 3 = დ.

შემთხვევითად აღებული ქრომოსომებით პირველ ჯერზე იქმნება ინდივიდების საწყისი პოპულაცია, რომელშიც, თითოეული ელემენტის ზომა სიტყვა-გასაღების ზომისა ტოლია.

ქვემოთ მოცემული კოდი კი ითვლის ამ ქტომოსომის(კლასი ქრომოსომა) ფიტნეს მნიშვნელობას:

```
// ითვლის ამ ობიექტის ფიტნეს მნიშვნელობას C#-დახმარებით
2 references
public void CalculateFitnessValue(List<GeoText> listGeoText, string CipherText)
{
    string DecryptionText = CryptoMethods.Decryption(CipherText, ChromosomeValue);

    foreach (GeoText txt in listGeoText)
    {
        if (DecryptionText.Contains(txt.Text))
        {
            FoundWords.Add(txt.Text);
            DecryptionText = DecryptionText.Replace(txt.Text, "");
        }
    }

    FitnessValue = DecryptionText.Length;
}
```

`List<GeoText> listGeoText` ლისტში წერია SQL-ის მონაცემთა ბაზიდან ამოღებული 300.000-მდე ქართული სიტყვა.

მოცემული ქრომოსომით ჯერ მოახდენს დაშიფრული ტექსტის დეშიფრაციას, ხოლო შემდეგ მასში ეძებს და შლის(ამოშლის) იმ სიტყვებს რომელიც ამ, ქართული სიტყვებისგან შემდგარ, ლისტში გვხვდება. ლისტში სიტყვები დალაგებულია სიგრძის კლების მიხედვით, ეს გაკეთებულია იმისთვის, რათა თავიდან იქნას აცილებული ისეთი შემთხვევები, როგორიცაა შემდეგი, თუ ტექსთში გვაქვს სიტყვა „ხელი“, სიტყვების ლისტში კი „ხე“ და „ხელი“, ჯერ უნდა შემოწმდეს, რომელიც უფრო გრძელის-„ხელი“, რადგან თუ ჯერ ამოიღებს სიტყვა „ხე“, მაშინ ტექსტში დარჩება „ლი“, სიტყვა რომელსაც აზრი არ გააჩნია, ამის გამო უშინაარსო სიტყვები დაგვიგროვდება ტექსტში. საბოლოოდ დარჩენილი სიმბოლოების(რომლებიც ვერ იპოვა ქსრთული სიტყვების ბაზაში) რაოდენობას გადათვლის და მიანიჭებს ამ ქრომოსომის ფიტნეს-მნიშვნელობას. ესეიგი რაც უფრო დაბალია ფიტნეს-ფუნქციის მნიშვნელობა, მით უკეთესად ახდენს დეშიფრაციას ეს, გასაღები და ამ ქრომოსომას გადარჩენის მეტი შანსიც გააჩნია სხვებთან შედარებით. თუკი დეშიფრაციის შემდეგ ტექსტში ყველა სიტყვა ნაცნობი აღმოჩნდება, ფიტნეს მნიშვნელობა მიიღებს მნიშვნელობას 0-ს, რაც საუკეთესო შედეგია(გესაღების გლობალური მაქსიმუმის მნიშვნელობა ნაპოვნია) და ეს გენეტიკური ალგორითმების მოქმედების დასრულების პირობაცაა.

P.S. ეს ფუნქცია საერთო ჯამში მოითხოვს ძალიან დიდ დროს. რაც გამოწვეულია სიტყვების დიდი რაოდენობითა და მისი ძებნით ტექსტში, რომელიც სათითაოდ ყველა ახალ ქრომოსომას უნდა ჩაუტარდეს.

მიმდინარე მომენტშ არსებული პოპულაციიდან ვიღებთ საუკეთესო ფიტნეს-მნიშვნელობის მქონე(რომლის მნიშვნელობაც ყველაზე პატარაა) ქრომოსომას და მისი საშვალებით განვსაზღვრავთ შეუღლების წერტილებს.

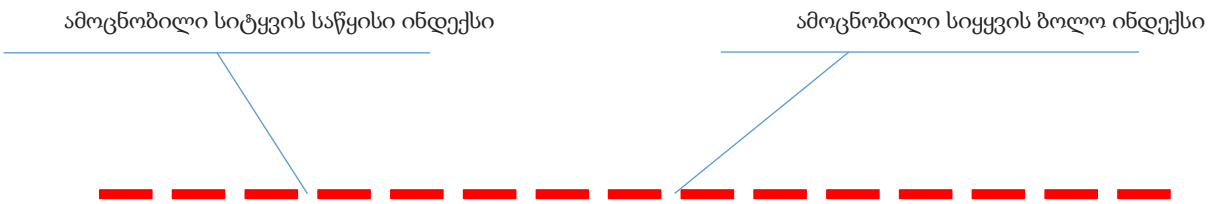
```
// შეუღლების წერტილების განსაზღვრა, საუკეთესო ქრომოსომიდან გამომდინარე (შეიძლება იყოს 1 ან 2 წერტილიანი)
1 reference
public static int[] DefineBreakingPoints(Chromosome BestChromosome, string ChiperText, List<GeoText> FullListGeoText)
{
    int[] BreakingPoints = new int[2];
    string DecryptionTextUsingBestChromosome = CryptoMethods.Decryption(ChiperText, BestChromosome.ChromosomeValue);
    int firstIndex = 0, lastIndex = 0;

    foreach (GeoText gt in FullListGeoText)
    {
        if (DecryptionTextUsingBestChromosome.Contains(gt.Text))
        {
            firstIndex = DecryptionTextUsingBestChromosome.IndexOf(gt.Text);
            lastIndex = firstIndex + gt.LenText;
            break;
        }
    }

    firstIndex = firstIndex % (BestChromosome.ChromosomeValue.Length);
    lastIndex = lastIndex % (BestChromosome.ChromosomeValue.Length);

    if (firstIndex < lastIndex)
    {
        BreakingPoints[0] = firstIndex;
        BreakingPoints[1] = lastIndex;
    }
    else
    {
        BreakingPoints[0] = lastIndex;
        BreakingPoints[1] = firstIndex;
    }
    return BreakingPoints;
}
```

მოცემული ქრომოსომით მოვახდენთ დეშიფრაციას და დავიწყებთ მასში(დეშიფრირებულ ტექსტში) ქართული სიტყვის ძებნას ყველაზე დიდიდან. როდესაც ვიპოვით, სიტყვის დასაწისსა და დასასრულს მოვნიშნავთ, შემდეგ ამოვიღებთ სიტყვა-გასაღების ზომის ნაშთს, მიღებული ორი რიცხვი ჩვენს ქრომოსომას გაყობს ორ ან სამ ნაწილად, საერთო ჯამში კი ორ სიმრავლედ, რომელთაგან ერთერთი ისაა, რომელმაც სწორად მოახდინა, ტექსტის იმ ნაწილის დეშიფრაცია სადაც ქართული სიტყვა აღმოვაჩინეთ. ეს მეთოდი საშვალებას იძლევა გადავარჩინოთ ქრომოსომის ის ნაწილი უცვლელად, რომელიც სწორად ახდენს დეშიფრაციას.



შემდეგ კი მოვახდენთ ამ ინდექსში ქრომოსომების წყვეტას და შეუღლებას:

ეს ხდება იმდენჯერ რამდენ სიტყვასაც სწორად ამოიცნობს, საბოლოოდ კი ვიღებთ წინასწარ განუსაზღვრელი რაოდენობის შეუღლების წერტილებს. მაგალითად აქ წყვეტის წერტილებია: 3, 8, 10, 12, 13 და 14 ლოკუსებში

```
// 1 ან 2 წერტილიანი შეუღლება, ქრომოსომების წყვილზე
1 reference
public static List<Chromosome> Conjugate(List<Chromosome> ChromosomePair, int[] BreakingPoints)
{
    List<Chromosome> NewChromosomePair = new List<Chromosome>();
    string FirstChromosome = ChromosomePair[0].ChromosomeValue;
    string SecondChromosome = ChromosomePair[1].ChromosomeValue;

    // თუ წყვეტის წერტილები ტოლია, გვექნება 1, თუ არა და 2 წერტილიანი შეუღლება
    if (BreakingPoints[0] == BreakingPoints[1])
    {
        int point = BreakingPoints[0];

        string FirstChild = FirstChromosome.Substring(0, point) + SecondChromosome.Substring(point);
        string SecondChild = SecondChromosome.Substring(0, point) + FirstChromosome.Substring(point);

        NewChromosomePair.Add(new Chromosome() { ChromosomeValue = FirstChild });
        NewChromosomePair.Add(new Chromosome() { ChromosomeValue = SecondChild });
    }
    else
    {
        int firstPoint = BreakingPoints[0];
        int secondPoint = BreakingPoints[1];

        string FirstChild = FirstChromosome.Substring(0, firstPoint) + SecondChromosome.Substring(firstPoint, secondPoint - firstPoint) + FirstChromosome.Substring(secondPoint);
        string SecondChild = SecondChromosome.Substring(0, firstPoint) + FirstChromosome.Substring(firstPoint, secondPoint - firstPoint) + SecondChromosome.Substring(secondPoint

        NewChromosomePair.Add(new Chromosome() { ChromosomeValue = FirstChild });
        NewChromosomePair.Add(new Chromosome() { ChromosomeValue = SecondChild });
    }
    return NewChromosomePair;
}
```

შეუღლება შეიძლება იყოს 1 წერტილიანიც, იმ შემთხვევაში თუკი საწყისი და საბოლოო წყვეტის წერტილის მნიშვნელობები ერთმანეთს დაემთხვევა.

მუტაციის ფუნქცია უტარდება სუსტი ქრომოსომებიდან ერთს, შემთხვევითად არჩეულს.

```
// მუტაციის ფუნქცია (რომელიმე ერთ ასოს ჩაანაცვლებს სხვა შემთხვევითით)
0 references
public static Chromosome Mutation(Chromosome chromosome)
{
    Random Rnd = new Random();
    int randSymbolNum = Rnd.Next(33);
    string symbol = Symbols.GeoSymbols[randSymbolNum].Name;

    int randGenNum = Rnd.Next(chromosome.ChromosomeValue.Length);
    string mutationText = chromosome.ChromosomeValue.Substring(0, randGenNum) + symbol + chromosome.ChromosomeValue.Substring(randGenNum + 1);

    return new Chromosome() { ChromosomeValue = mutationText };
}
```

რომელიმე გენი(ერთერთი სიმბოლო), რომელსაც ასევე შემთხვევითად ვირჩევთ, ჩანაცვლდება სხვა შემთხვევითად არჩეული ქართული სიმბოლით.

სელექციის მეთოდად ვიყენებ შემდეგს: ძველი და ახალი თაობებიდან ვტოვებ საუკეთესოების 10% შემდეგ ძველი თაობიდან შემთხვევითად ვტოვებ პოპულაციის 20% და ახალი თაობიდან შემთხვევითად 60%-ს. საბოლოო ჯამში ვიღებ იმავე რაოდენობის ახალ თაობას, სადაც ყველა ზემოთ ჩამოთვილილი მოქმედება თავიდან მეორდება, სანამ არ მიიღწევა დასრულების პირობა, პირობა კი არის, ან საუკეთესო ქრომოსომის პოვნა, რომელმაც მთელი ტექსტი სწორად გაშიფრა, ან ციკლების შესრულების რაოდენობა, რომელიც კომპიუტერის სიმძლავრიდან და ლოდინის დროის ხანგრძლივობიდან გამომდინარე მიეთითება.

4.3.4 შიფროტექსტის დეშიფრაცია, მიღებული გასაღების საშუალებით;

საბოლოოდ, მიღებული სავარაუდო გასაღებით, ვახდენთ დაშიფრული ტექსტის დეშიფრაციას და აღმოჩენილ სიტყვებს სხვებისგან გამოვყობთ, რათა იყოს თვალისთვის ადვილად შესამჩნევ. უნდა ითქვას რომ პასუხის სრული სიზუსტე არ არის მოსალოდნელი, ამ ალგორითმის მთავარი მიზანი ისაა, რომ, ტექსტი გახადოს ადამიანის თვალისთვის უფრო გასაგები, რათა მან შეძლოს გარკვეული აზრის გამოტანა, ადამიანის გონებისთვის კი, შემდეგი ეტაპი, ტექსტში „ბნელი ადგილების“ შევსება უფრო გამარტივდება.

დასკვნა

საბოლოო ჯამში, გენეტიკური ალგორითმები საკმაოდ კარგი მიდგომაა კრიპტოგრაფიული ალგორითმების კრიპტოანალიზისთვის, სისწრაფის და წარმადობის მხრივ, რათქმაუნდა საჭიროა თავისებური მეთოდების შექმნა.

ჩვენი ალგორითმი საკმაოდ კარგადაა შედგენილი ვიჟინერის შიფრისთვის, მაგრამ გვაქვს სემანტიკური პრობლემები. სიტყვების ბაზაში, რომელსაც ვიყენებთ დეშიფრირებული ტექსტის შემოწმებისას, თუ რამდენად შეესაბამება ის ქართულ ტექსტს გვხვდება დიდი რაოდენობით არაქართული სიტყვები. ეს ბაზა ჩვენ მოვიძეთ ინტერნეტიდან, რადგან სხვა საშვალება არ გვქონდა, თავიდან შედგენისა ან კორექტირებისთვის კი ძალიან დიდი დროა საჭირო.

გამოყენებული ლიტერატურა

- https://ka.wikipedia.org/wiki/კრიპტოგრაფია
- www.wikiwand.com/ka/ჩანაცვლებადი_შიფრი
- https://mathtasks.wordpress.com/2015/03/22/გენეტიკური-ალგორითმები-ga/
- https://ru.wikipedia.org/wiki/Метод_Касиски
- ზ.ქოჩლაძე - ინფორმაციული უსაფრთხოების ტექნოლოგიები
- რ.მეგრელიშვილი - ინფორმაციის დაცვის სისტემები
- Д.Рутковская, М.Пилиньский, Л.Рутковский - Нейронные сети, генетические алгоритмы и нечеткие системы