ლალი ბესელია, ინგა გაბისონია, სოფია შენგელია

კრიპტოგრაფიული ალგორითმები: მაგალითები და პროგრამული კოდები

გამოიცა „შოთა რუსთაველის საქართველოს ეროვნული სამეცნიერო ფონდის მხარდაჭერით გრანტი N SP-2-21-599, კრიპტოგრაფიული ალგორითმები: მაგალითები და პროგრამული კოდები“

თბილისი
2022

რეცენზენტი: ზურაბ ქოჩლაძე, ივ.ჯავახიშვილის სახ. თბილისის სახელმწიფო უნივერსიტეტის კომპიუტერული მეცნიერებების დეპარტამენტის გამოყენებითი ინფორმატიკის კათედრის ასოცირებული პროფესორი

ISBN 978-9941-8-4691-5

დაბეჭდილია შპს „პრინტჯეო“-ს მიერ
თბილისი, 2022წ.

ანოტაცია

წარმოდგენილი სახელმძღვანელო სტუდენტებსა და ინფორმაციული უსაფრთხოების საკითხებში დაინტერესებულ პირებს გააცნობს კრიპტოგრაფიულ ალგორითმებს პროგრამული რეალიზაციის თანხლებით.

Abstract

The presented textbook is intended for students and all persons interested in information security issues. The book will introduce them to cryptographic algorithms and their software implementations.

პროექტის ხელმძღვანელი: ლალი ბესელია, სოხუმის სახელმწიფო უნივერსიტეტის ინფორმატიკის მიმართულების ას. პროფესორი, თბილისის 128-ე საჯარო სკოლის კომპიუტერული ტექნოლოგიების მასწავლებელი.

პროექტის კოორდინატორი, წიგნის რედაქტორი: სოფია შენგელია**,** სოხუმის სახელმწიფო უნივერსიტეტის ინფორმატიკის მიმართულების ას. პროფესორი, თბილისის 12-ე საჯარო სკოლის კომპიუტერული ტექნოლოგიების მასწავლებელი.

პროექტის ძირითადი პერსონალი, წიგნის ტექნიკური რედაქტორი: ინგა გაბისონია, სოხუმის სახელმწიფო უნივერსიტეტის ინფორმატიკის მიმართულების ასოც. პროფესორი, ქართველ იუნკერთა სახელობის თბილისის 52-ე საჯარო სკოლის მათემატიკის მასწავლებელი.

წინასიტყვაობა

ინფორმაციული უსაფრთხოების დაცვა მნიშვნელოვან პრობლემას წარმოადგენს. კიბერშეტევების მსხვერპლი შეიძლება იყოს როგორც ფიზიკური პირი, ასევე სახელმწიფო და ეკონომიკური სტრუქტურები. ინფორმაციის უსაფრთხოების უზრუნველყოფა არის ერთ-ერთი უმნიშვნელოვანესი ამოცანა თანამედროვე საზოგადოებისთვის. ინფორმაციული უსაფრთხოების დაცვა კიდევ უფრო ღირებული გახდა პანდემიის პირობებში. საზოგადოება სულ უფრო და უფრო ხშირად იყენებს ინტერნეტ რესურსებს: დისტანციური სწავლება, ინტერნეტ მაღაზია, ონლაინ გადასახადების სისტემა და ასე შემდეგ. შესაბამისად იმატა კიბერთავდასხმების რაოდენობამ. ციფრული ტექნოლოგიების ეპოქაში გავრცელებულია ე.წ. ელექტრონული ხელმოწერაც, რომელიც გამოიყენება იმ პირის ავთენტურობის დასადგენად, რომელიც უფლებამოსილია ისარგებლოს გარკვეული ელექტრონული სერვისებით. ცნობილია, რომ კიბერუსაფრთხოების უზრუნველსაყოფად და თანამედროვე გამოწვევების საპასუხოდ

აუცილებელია კომპლექსური მიდგომა, აქედან ერთ-ერთ აუცილებელ პირობას წარმოადგენს კრიპტოგრაფიული უზრუნველყოფა. აქედან გამომდინარე, წარმოდგენილი წიგნი სტუდენტებსა და ინფორმაციული უსაფრთხოების საკითხებში დაინტერესებულ პირებს გააცნობს კრიპტოგრაფიულ ალგორითმებს პროგრამული რეალიზაციის თანხლებით.

ნებისმიერი კრიპტოსისტემა, რომელიც გამოიყენება ტექსტის დასაშიფრად, შიფროტექსტში უშვებს ინფორმაციის გაჟონვას გასაღებისა და ღია ტექსტის შესახებ, ამიტომ დროთა განმავლობაში ნებისმიერი კრიპტოალგორითმი „ბერდება“ და მისი გამოყენება შეუძლებელი ხდება. აქედან გამომდინარე, ახალი, სწრაფი და გამძლე კრიპტოალგორითმების აგება, რომლებიც გაუძლებენ თანამედროვე კიბერშეტევებს, ყოველთვის იყო და დღესაც ძალიან მნიშვნელოვანია. ამ მიზნით ნაშრომში წარმოდგენილია კრიპტოგრაფიული ალგორითმები. განხილულ ალგორითმებს გააჩნია პრაქტიკული მნიშვნელობა, ამიტომაც მნიშვნელოვანი ყურადღება ეთმობა მათი პროგრამული რეალიზაციის პრობლემებს, თავისებურებებს).

სახელმძღვანელო ძირითადად განკუთვნილია სამივე საფეხურის სტუდენტებისთვის (ბაკალავრიატი, მაგისტრატურა, დოქტორანტურა).

წიგნი შედგება სამი ნაწილისგან. პირველი ნაწილში მოცემულია კლასიკური კრიპტოალგორითმების აღწერა, მოკლე კრიპტოანალიზი, პროგრამული რეალიზაცია, მაგალითები და სავარჯიშოები.

მეორე ნაწილში განხილულია თანამედროვე სიმეტრიული ალგორითმები და ნაკადური შიფრები; მათი კრიპტოანალიზი, პროგრამული რეალიზაცია, მაგალითები და სავარჯიშოები, ინფორმაციის მთლიანობის პრობლემა და მისი გადაჭრის მეთოდები.

მესამე ნაწილში განხილულია ასიმეტრიული ალგორითმები, ციფრული ხელმოწერისა და ელიფსური კრიპტოგრაფიის ალგორითმები; მათი კრიპტოანალიზი, პროგრამული რეალიზაცია, მაგალითები და სავარჯიშოები.

ს ა რ ჩ ე ვ ი

შესავალი

კრიპტოგრაფია არის მეცნიერება, რომელიც შეისწავლის ინფორმაციის დაცვისა და აუტენტიფიკაციის მეთოდებს. ინფორმაციული უსაფრთხოების დაცვა მსოფლიოში ყოველთვის აქტუალური საკითხი იყო. ჯერ კიდევ პირველ მსოფლიო ომამდე პერიოდულად ჩნდებოდა პუბლიკაციები, კრიპტოგრაფიის, როგორც სპეციალიზებული დარგის განვითარების შესახებ. კრიპტოგრაფია თავდაპირველად, ძირითადად სამხედრო საქმიანობას ემსახურებოდა. ცნობილია, რომ ედვარდ ჰებერმა კალიფორნიიდან 1918 წელს როტორულ მანქანაზე მიიღო პატენტი. ეს მოწყობილობა დაახლოებით 50 წელი სამხედრო კრიპტოგრაფიას ემსახურებოდა. იგივე ტიპის „ლორენცის მანქანა“ მეორე მსოფლიო ომის დროს გერმანიის მიერ ფართოდ გამოიყენებოდა გასაიდუმლოებული შეტყობინების შიფრა-ციისთვის.

კრიპტოგრაფიის, როგორც მეცნიერების სწრაფი აღმავლობა გასული საუკუნის 30-40-ან წლებში მოხდა, თუმცა ამ პერიოდში დარგის ნაშრომები ღიად არ ქვეყნდებოდა. გამონაკლისს წარმოადგენდა კლოდ შენონის სტატია „The

Communication Theory of Secrecy Systems" (1949 წ). შემდგომში, 1967 წელს გამოვიდა დევიდ კანის წიგნი „The Codebreakers", რომლის მეშვეობითაც ამერიკულმა ფართო საზოგადოებამ გაიცნო კრიპტოგრაფიის დარგის საწყისები.

1975 წლიდან კრიპტოგრაფიაში დიფი და ჰელმანის მიერ ღია გასაღებების გამოყენების ეტაპი დადგა. ამ პერიოდიდან დაიწყო უკვე ისეთი კრიპტოსისტემების შექმნა, რომლებიც უკვე რიგ სერიოზულ ამოცანებს წყვეტდნენ. შემდგომ პერიოდში ბრიუს შრაინერის მიერ შეიქმნა და გამოიცა ფუნდამენტური ნაშრომი „გამოყენებითი კრიპტოგრაფიის" სახელწოდებით, სადაც ავტორ-მა განიხილა მრავალფეროვანი მაგალითები, დაწყებული ტელეფონის საუბრის კოდირებიდან დამთავრებული არჩევნების კრიპტოგრაფიული უზრუნველყოფით.

გასული საუკუნის 70-80 იან წლებში კრიპტოგრაფია კვლავ საზოგადოებრივი ყურადღების ქვეშ მოექცა, თუმცა ამერიკაში მოქმედი სახელმწიფო ორგანო NSA - National Security Agency ყველანაირად ცდილობდა გაენელებინა ეს ინტერესი, შემოეტანა შეზღუდვები კრიპტოგრაფიული მეთოდების გასაჯაროების

საქმეში, რადგან კრიპტოგრაფიის გამოყენება ეროვნულ უსაფრთხოებისთან არის დაკავშირებული. კრიპტოგრაფიული მეთოდების გამოქვეყნებაზე კონტროლის დაწესების გადაწყვეტილებები მიღებული იქნა როგორც ამერიკის კონგრესის მიერ, ასევე რიგი ქვეყნების სახელმწიფო უწყებების მიერ, ისეთ ქვეყნებში, როგორიცაა ინგლისი, საფრანგეთი, ისრაელი. იგივე პროცესები ხდებოდა, რა თქმა უნდა, იმ დროს ჯერ კიდევ არსებულ საბჭოთა კავშირში. ეს ქვეყნები მილიარდობით დოლარს ხარჯავდნენ უსაფრთხოების მიმართულებით, მაგალითად, საკუთარი კავშირგაბმულობის უსაფრთხოებაზე და ამასთან ერთად სხვადასხვა მეთოდებით ცდილობდნენ სხვა ქვეყნის უსაფრთხოების სისტემაში შეღწევას.

20 საუკუნის ბოლოს დაიწყო კრიპტოგრაფიული მეთოდების გამოყენება უკვე ცალკეული პიროვნებების ინტერესების უსაფრთხოების უზრუნველყოფის მიზნით. ასე მაგალითად, თუ ინფორმაციული უსაფრთხოება არ არის დაცული, პიროვნებამ შეიძლება სხვა ადამიანებს ზიანი მიაყენოს რაიმე პოლიტიკური კომპანიაში, საგადასახადო საქმიანობაში, ბიზ-

ნესის დაგეგმვისა და წარმოების დროს. სათანადო უსაფრთხოების სისტემის არქონის შემთხვევაში შესაძლებელია პიროვნების პირად ცხოვრებაში შეღწევა და უკანონო ქმედებების წარმოება. სწორედ ასეთი ქმედებებისგან გვიცავს კრიპტოგრაფია.

კრიპტოგრაფია ერთდროულად შიფრაციის მეცნიერებასა და ხელოვნებას წარმოადგენს, თუმცა თანამედროვე სახით, ის უკვე მოიცავს აუთენტიფიკაციას, ციფრულ ხელმოწერებს და უსაფრთხოების სხვა მეთოდებს. კრიპტოგრაფიის გამოიყენების სფეროების ძალიან ფართოა: კომპიუტერული უსაფრთხოება, უმაღლესი მათემატიკა, ეკონომიკა, კვანტური ფიზიკა, სამოქალაქო და სისხლის სამართალი, სტატისტიკა, მიკროსქემების პროექტირება, ექსტრემალური პროგრამირება, პოლიტიკა, სამომხმარებლო ინტერფეისი და სხვა.

კრიპტოგრაფია დიდი უსაფრთხოების სისტემის ერთი ნაწილია. თუ პირობითად, უსაფრთხოების სისტემას კედლებსა და კარებს შევადარებთ, მაშინ კრიპტოგრაფია კარის საკეტის ანალოგია. მხოლოდ კარის საკეტს არანაირი ფუნქციური დანიშნულება არ აქვს, ის ერთიანი

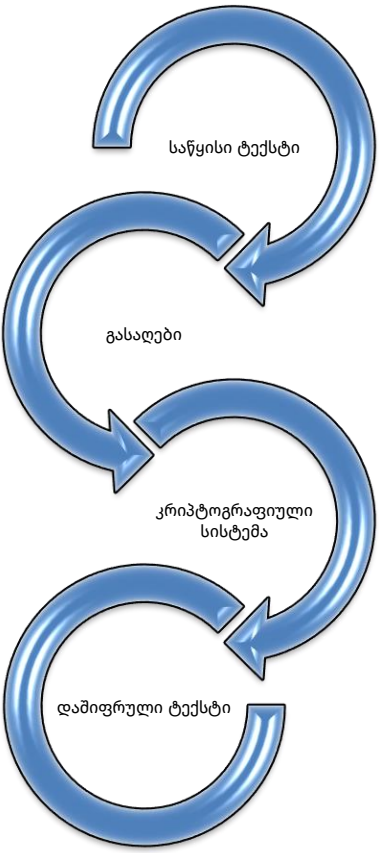
სისტემის ერთ-ერთი რგოლია გარკვეული თვისებით: გაატაროს ზოგიერთი და არ გაატაროს სხვები. კედლებისა და კარის ხარისხის მიუხედავად, საკეტი იდეალურად უნდა მუშაობდეს. პრაქტიკაში უსაფრთხოების სისტემის ყოველი რგოლი საიმედო უნდა იყოს და მაშინ კრიპტოგრაფიაც თავისი დანიშნულებისამებრ იმუშავებს. ასე მაგალითად, რომელიმე კრიპტოგრაფიული სისტემა გრძელი გასაღებით რაიმე საიტზე შეიძლება იდეალურად მუშაობდეს, მაგრამ ბოროტმოქმედი ეძებს საიტის მწყობრიდან გამოყვანის მარტივ გზას. კრიპტოგრაფიული სისტემის გატეხვის ნაცვლად მან შეიძლება web-სერვერის ბუფერი გადაავსოს. აქედან შეიძლება დავასკვნათ, რომ საიმედო კრიპტოგრაფიული სისტემის გამოყენებას მაშინ აქვს აზრი, როცა უსაფრთხოების სისტემის ყველა სხვა რგოლი (როგორც შენობის კედლები და კარი) საიმედოა. თუმცა ბოროტმოქმედის მიერ კრიპტოგრაფიული სისტემის გატეხვის შემთხვევაში გატეხვის ფაქტი შეიძლება შეუმჩნეველიც კი დარჩეს. წარმოიდგინეთ, შენობის კარი თუ იარაღით გატეხეს, ეს შესამჩნევია, მაგრამ თუ ბოროტმოქმედმა კარს გასაღები შეურჩია და

ჩვეულებრივ გახსნა, მაშინ ამ ფაქტს სანქციონირებული ვიზიტისაგან ვერ გავარჩევთ. გასაღების ქონა კი ბოროტმოქმედს აძლევს საშუალებას, არაერთხელ შემოიჭრას შენობაში და ზიანი მოგვაყენოს. ეს ფაქტი კრიპტოგრაფიული სისტემის გატეხვის ზუსტი ანალოგია.

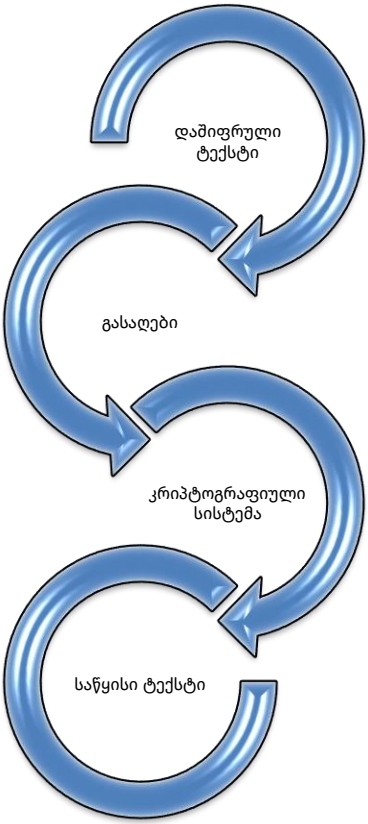
ზემოთ მოყვანილი მსჯელობიდან გამომდინარეობს, რომ მთელი სისტემის უსაფრთხოება დამოკიდებულია ამ სისტემის ყველაზე სუსტი რგოლის უსაფრთხოებაზე. შენობის დაცვის შემთხვევაში, რაც არ უნდა კარგი ხარისხის კარის საკეტი გვქონდეს, თუ შენობას გააჩნია, მაგალითად, ცუდად დაცული სავენტილაციო სისტემა, საიდანაც შეიძლება შემოძვრეს ბოროტმოქმედი, სხვა დაცვის საშუალებებს ყოველგვარი აზრი ეკარგებათ. უსაფრთხოების ნებისმიერი სისტემა მრავალი რგოლისგან შედგება. ყოველი რგოლი, თავის მხრივ, ცალკეულ კომპონენტებს მოიცავს. ასეთნაირად ელემენტარულ კომპონენტებად დაყოფილი უსაფრთხოების სისტემა ე.წ. ხისმაგვარ სტრუქტურას წარმოადგენს. ხისმაგვარ სტრუქტურაში ყველა კომპონენტის საიმედოობის დადგენა ხანგრძლივი და შრომატევადი პროცესია.

დაცვის სისტემაში ცალკეული კომპონენტების სისუსტის ანალიზი რეალური შეტევის დროს აუცილებელია. ითვლება, რომ მომხმარებლების პაროლები სანდოა, მაგრამ პრაქტიკაში ხშირად ხდება, რომ პაროლები ან ძალიან პრიმიტიულია, ან შესაძლებელია ადვილი მოსაპოვებელი იყოს, მაგალითად, ფურცელზე დაწერილი კომპიუტერის მონიტორზე იყოს მიკრული. ასეთ შემთხვევაში, შეტევის დროს დაცვის ეს რგოლი ყველაზე სუსტია. უნდა ითქვას ისიც, რომ შეტევის დროს თავდამსხმელმა არ იცის, დაცვის რომელი რგოლია სუსტი. უნდა აღინიშნოს ის ფაქტიც, რომ კრიპტოგრაფიულ სისტემაზე შეტევა შეიძლება ამ სისტემის შექმნიდან რამდენიმე წელში მოხდეს, შეტევის დროს შეიძლება გამოყენებული იქნეს უახლესი ტექნოლოგიები, სხვადასხვა უკანონო გზები, მაგრამ სისტემამ ნებისმიერ შეტევას უნდა გაუძლოს. თუ, ვთქვათ, იქმნება ელექტრონული გადახდების სისტემა, სისტემის შემმუშავებელმა ყველა მხარე - მყიდველი, გამყიდველი, მყიდველისა და გამყიდველის საბანკო სისტემა - უნდა ჩათვალოს ერთმანეთზე პოტენციური შეტევის განმახორციელებლად და ასეთი

პირობების გათვალისწინებით შექმნას სათანდო პროგრამული პროდუქტი.



ნახაზი- 1 მონაცემთა შიფრაციის პროცესი



ნახაზი- 2 მონაცემთა დეშიფრაციის პროცესი

საწყისი (ღია) ტექსტი - ტექსტური მონაცემები (ან სხვა ტიპის მონაცემები), რომელიც გადაიცემა კრიპტოგრაფიული ალგორითმის გამოყენების გარეშე.

დაშიფრული (დახურული) ტექსტი - მონაცემები რომლებიც მიიღება ღია ტექსტიდან საიდუმლო გასაღების მქონე კრიპტოგრაფიული სისტემის გამოყენებით.

კრიპტოგრაფიული სისტემა - წარმოადგენს შექცევადი გარდაქმნების ჯგუფს, რომელიც საწყის ტექსტს გარდაქმნის დაშიფრულ ტექსტად.

გასაღები - შიფრის კომპონენტი, რომელიც უზრუნველყოფს საწყისი ტექსტის კონკრეტული გარდაქმნის არჩევანს. თანამედროვე კრიპტოსისტემებში დაშიფვრის ალგორითმი ცნობილია და სიფრის კრიპტოგრაფიული მედეგობა სრულად დამოკიდებულია გასაღების საიმედოობაზე და ამას კერგოფის უწოდებენ.

კრიპტოანალიზი - მაცნიერების დარგია, რომლის მიზანია კრიპტოგრაფიული ალგორითმების გატეხვის მეთოდების ძიება და კვლევა, აგრეთვე ალგორითმის გატეხვის პროცესის უზრუნ-

ველყოფა.

კრიპტოანალიტიკოსი - სპეციალისტი, რომელიც შეიმუშავებს და იყენებს კრიპტოანალიზის მეთოდებს.

შიფრაცია - საწყისი ტექსტის გარდაქმნა გასაღების გამოყენებით დაშიფრულ ტექსტად.

დეშიფრაცია - შიფრაციის უკუპროცესი. დაშიფრული ტექსტის გარდაიქმნა გასაღების გამოყენებით საწყის ტექსტად.

კრიპტოლოგია - შეისწავლის კრიპტოგრაფიულ გარდაქმნებს. იგი შეიცავს ორ ძირითად მიმართულებას - კრიპტოგრაფიასა და კრიპტოანალიზს.

კრიპტოგრაფიული ალგორითმი - მონაცემთა გარდაქმნის ალგორითმი, რომელიც საიდუმლო გასაღებებს იყენებს.

კრიპტოგრაფიული თავდასხმა - კრიპტოანალიტიკოსის მიერ შეტევის განხორციელება დაცულ სისტემაში ინფორმაციის გაცვლისას. თუ კრიპტოგრაფიული თავდასხმა წარმატებულია, მაშინ ითვლება რომ მოხდა სისტემის გატეხვა.

კროპტოგრაფიული მედეგობა - კრიპტოგრაფიული ალგორითმის უნარი გაუძლოს კრიპტოანალიზს.

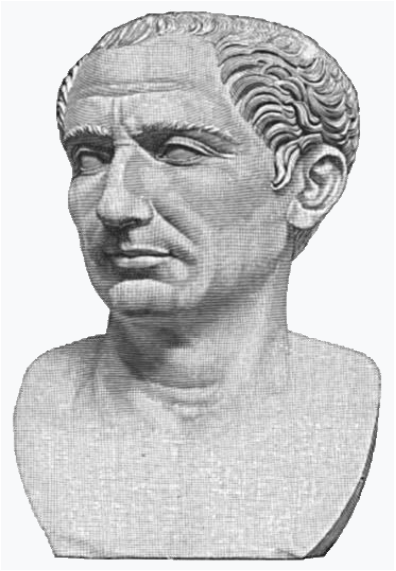
თავი I. კლასიკური შიფრები

ამ თავში მოცემულია კლასიკური კრიპტოგრაფიული ალგორითმების ცეზარის, ვიჟენერისა და ვერნამის ალგორითმების აღწერა, შესაბამისი პროგრამული კოდები, მაგალითები და მოკლე კრიპტოანალიზი.

ცეზარის ალგორითმი

ძველ შიფრებს შორის ყველაზე ცნობილს წარმოადგენს ცეზარის (კეისრის) შიფრი, რომელსაც იულიუს კეისრის (100-44 წწ. ჩვენს წელთაღრიცხვამდე) სახელი უწოდეს. ინფორ-მაციის დამალვის მეთოდი, რომელიც ამ მეთოდში გამოიყენება, ძალიან მარტივია. ყოველი ასო იცვლება სხვა ასოთი, რომელიც მიიღება ანბანში თავდაპირველი ასოს რამდენიმე პოზიციის წანაცვლებით. მიმღებისთვის საკმარისია იმის ცოდნა, თუ რამდენი პოზიციით ხდება წანაცვლება. ცეზარის ალგორითმი მიეკუთვნება ჩანაცვლების ტიპის შიფრს. უცნობია, კეისარმა თვითონ მოიგონა დაშიფვრის ეს მეთოდი, თუ უბრალოდ გამოიყენა სამხედრო ჩანაწე-

რებისათვის. ზოგიერთი ისტორიკოსი თვლის, რომ კეისარი ახორციელებდა ჩანაცვლებას 4-ის ტოლი ბიჯით (ანუ ანბანის ყოველი ასოს მისგან მარჯვნივ მეოთხე ასოთი შეცვლას), ზოგი კი - თვლის, რომ შიფრაცია ბიჯით 3 ხდებოდა. ზოგიერთი მეცნიერი თვლის რომ წანაცვლება არა მარჯვნივ, არამედ მარცხნივ ხდებოდა. აი, მაგალითად, კეისრის ცნობილი გამონათქვამი „Veni, Vidi, Vici!" (რაც ნიშნავს „მოვედი ვნახე, გავიმარჯვე!") 4 სიმბოლოთი მარჯვნივ წანაცვლებით ჩაიწერება, როგორც ZIRM ZMHM ZMGM, ხოლო 3 სიმბოლოთი მარცხნივ წანაცვლებით, როგორც SBKF SFAF SFZF. ასე რომ ცნობილი მხედართმთავარი ისტორიაში არა მარტო თავისი გამარჯვებებითაა ცნობილი, არამედ კლასიკური კრიპტოგრაფიული ალგორითმით. რადგანაც ომის დროს შიფრის გამოყენება არანაკლებ მნიშვნელოვანია, ვიდრე მრავალრიცხოვანი სამხედრო რეზერვები.

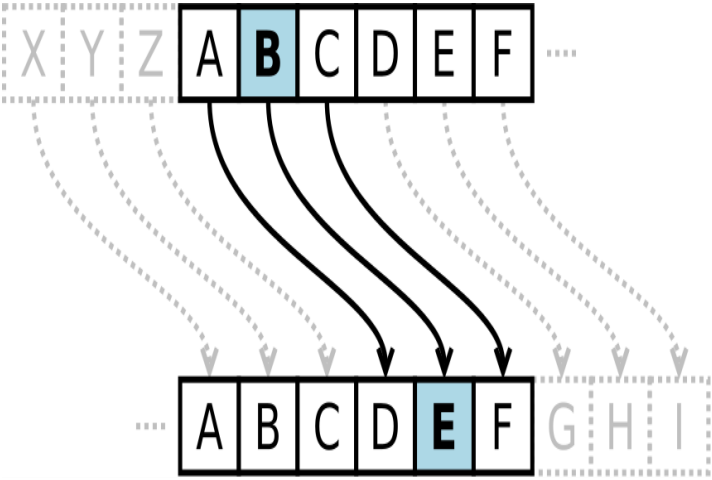


ნახაზი- 3 იულიუს კეისარი

ნახაზი- 4 მბრუნავი დისკები

შიფრაცია - დეშიფრაციისთვის გამოყენებული იქნა კონსტრუქცია შედგენილი ორი მბრუნავი დისკისაგან ცეზარის შიფრით.

წარმოვიდგინოთ, რომ იულიუს კეისარი (ცეზარი) თავის გზავნილში ლათინური ანბანის პირველ *A* ასოს შეცვლიდა, ვთქვათ, მეოთხე *D* ასო-ნიშნით, მეორე *B* ასოს მეხუთე - *E* ასო-ნიშნით და ა.შ. ბოლო *Z* ასოს მესამეთი. ამრიგად, თუ პირველ სტრიქონში ჩავწერთ ლათინური ანბანის ასო-ნიშნებს ჩვეულებრივი რიგის მიხედვით, ხოლო მეორე სტრიქონში დავალაგებთ შიფრის შესაბამის ასო-ნიშნებს, მივიღებთ დაშიფვრის ალგორითმს:



ნახაზი- 5 ცეზარის ალგორითმით დაშიფვრის სქემა

ი.ცეზარის ალგორითმში გასაღებს განსაზღვრავს გადანაცვლების წესი, კერძოდ, ის რომ ყოველი ასო-ნიშანი ანბანურ მწკრივში გადაინაცვლებს სამი პოზიციით, ანუ, საზოგადოდ, - k პოზიციით, სადაც $k \in \{1, 2,...,26\}$ (ი.ცეზარის ალგორითმში $k = 3$). ცხადია, რომ შესაძლებელია გადანაცვლების წესის გართულება, ვთქვათ სპეციალური ცხრილების (მატრიცების) შემოღება და სხვ.

ცეზარის ალგორითმის მათემატიკური ფორ-მულირება:

მათემატიკური ფორმულირება:

$$y = (x + k) mod\ m$$
$$x = (y - k) mod\ m$$

სადაც x არის ღია ტექსტი, k - გასაღები, y - დაშიფრული ტესტი, m - სიმბოლოთა რაოდენობა.

განვიხილოთ მაგალითი #1

საწყისი ტექსტი:	A	B	K	H	A	Z	I	A
დაშიფრული ტექსტი:	D	E	N	K	D	C	L	D

განვიხილოთ მაგალითი #2

საწყისი ტექსტი:	C	O	M	P	U	T	E	R
დაშიფრული ტექსტი:	F	R	P	S	X	W	H	U

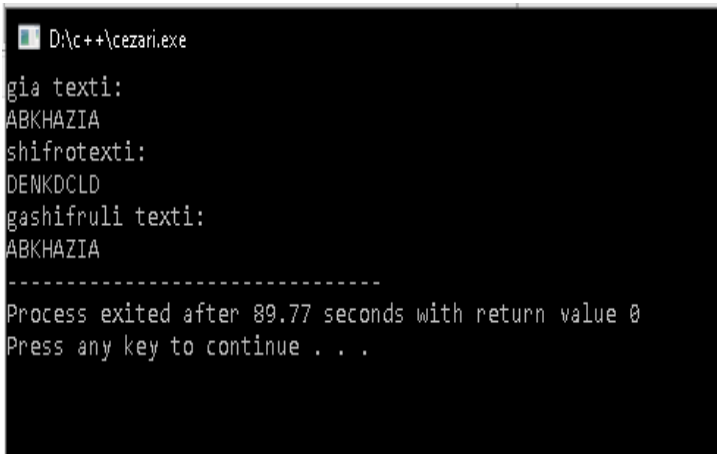
ცეზარის ალგორითმის პროგრამული რეალიზაცია:

```
#include <iostream>
#include <string>
using namespace std;
int main()
{string str;
char c;
 int k;
 k=3; //გასაღები
 cout<<"gia texti:"<<'\n';
getline(cin, str); //ღია ტექსტის შემოტანა
for(int i=0; i<str.length(); i++)       //დაშიფვრა
```

```
        if(str[i]>='A'&& str[i]<='Z')
        str[i]=(str[i]-'A'+k)%26+'A';
        else if(str[i]>='a'&& str[i]<='z')
        str[i]=(str[i]-'a'+k)%26+'a';
              else if(str[i]>='0'&& str[i]<='9')
        str[i]=(str[i]-'0'+k)%10+'0';
              cout<<"shifrotexti:"<<'\n';
for(int i=0; i<str.length(); i++) //შიფროტექსტის
//ბეჭდვა
        cout<<str[i];
        cout<<endl;
        for(int i=0; i<str.length(); i++) //გაშიფვრა
        if(str[i]>='A'&& str[i]<='Z')
         { c=(str[i]-'A'-k)%26+'A';
           if(c<'A') c+=26;
           str[i]=c;
        }
        else if(str[i]>='a'&& str[i]<='z')
         { c=(str[i]-'a'-k)%26+'a';
           if(c<'a')c+=26;
           str[i]=c;
        }
         else if(str[i]>='0'&& str[i]<='9')
{ c=(str[i]-'0'-k)%10+'0';
```

```
    if(c<'0') c+=10;
    str[i]=c;
}
cout<<"gashifruli texti:"<<'\n';
for(int i=0; i<str.length(); i++)
cout<<str[i];
}
```

შედეგი:



კრიპტოანალიზი:

ცეზარის შიფრში ხდება მხოლოდ სიმბოლოთა ჩანაცვლება, არ იცვლება სიმბოლოთა სიხშირე. გასაღების ვარიანტების სიმცირის გამო ძალისმიერი შეტევა არის ეფექტური და მარტივი. მის გასატეხად საკმარისია შიფრული ტექსტის თითოეული სიმბოლო ჩაანაცვლოს ანბანში მის

მარცხნივ მდგომმა სიმბოლომ, თუ ამის შედეგად შეუძლებელი გახდა წასაკითხი შეტყობინების მიღება, მაშინ აუცილებელია მოქმედების გამეორება, მაგრამ უკვე უნდა ჩაანაცვლოს ორით მარცხნივ მდგომმა სიმბოლომ. ასე გაგრძელდება მანამ, სანამ შედეგი არ იქნება საწყისი ტექსტი.

დავალება:

განახორციელეთ ტექსტის შიფრაცია - დე-შიფრაცია ცეზარის ალგორითმის გამოყენებით. საწყისი ტექსტი: CRYPTOGRAPHY

ვიჟენერის ალგორითმი

ვიჟენერის შიფრი პოლიალფაბეტური ჩანაც-ვლების მარტივი ფორმაა. ეს მეთოდი პირველად აღწერა ჯოვანი ბატისტა ბელასომ (Giovan Battista Bellaso) წიგნში La cifra del. Sig. Giovan Battista Bellaso 1553 წელს, მაგრამ მე-19 საუკუნეში მან მიიღო ფრანგი დიპლომატის ბლეზ ვიჟენერის სახელი. მეთოდი მარტივია როგორც გასაგებად, ასევე მისი რეალიზაციისთვის.

ცეზარის ალგორითმში ტექსტის ყოველი ასო-ნიშანი ანბანში გადაადგილდება პოზიციათა

ერთსა და იმავე რაოდენობით, რათა მივიღოთ დაშიფრული ტექსტი (k სიდიდე მუდმივია ყოველი ასო-ნიშნისათვის).

ცხადია, ჩნდება აზრი რომ გადანაცვლება მოვახდინოთ არა მუდმივი სიდიდით, არამედ განსხვავებული წესით. ვთქვათ, პირველი ტექსტური ასო-ნიშანი გადაადგილდეს k_1 პოზიციით, მე-2 - k_2 პოზიციით და ა.შ.

შეიძლება შემოვიღოთ l-სიგრძის $k = k_1, \dots, k_l$ გასაღები, ანუ l სიგრძის გარკვეული სიტყვა ან l სიგრძის მთელი წინადადება. მეორე მიმართულება გულისხმობს გარკვეული მატრიცის აგებას,რომელიც განახორციელებს დაშიფვრას. ასეთია ვიჟენერის მიერ შემოთავაზებული ალგორითმი, ანუ ცეზარის მოდიფიცირებული შიფრი.

მოსახერხებელია, რომ ასო-ნიშანთა განხილული გადანაცვლება ჩავწეროთ შემდეგი სახით. დავუშვათ, რომ ანბანის ასო-ნიშნებს შევუსაბამეთ რიცხვები ანბანში მათი რიგითი პოზიციის მიხედვით. ღია ტექსტი განვათავსოთ პირველ სტრიქონში, როგორც ცეზარის ალგორითმშია. მეორე სტრიქონში ჩავწეროთ ჩვენ მიერ შერჩეული „სიტყვა-გასაღები” განმეორებით

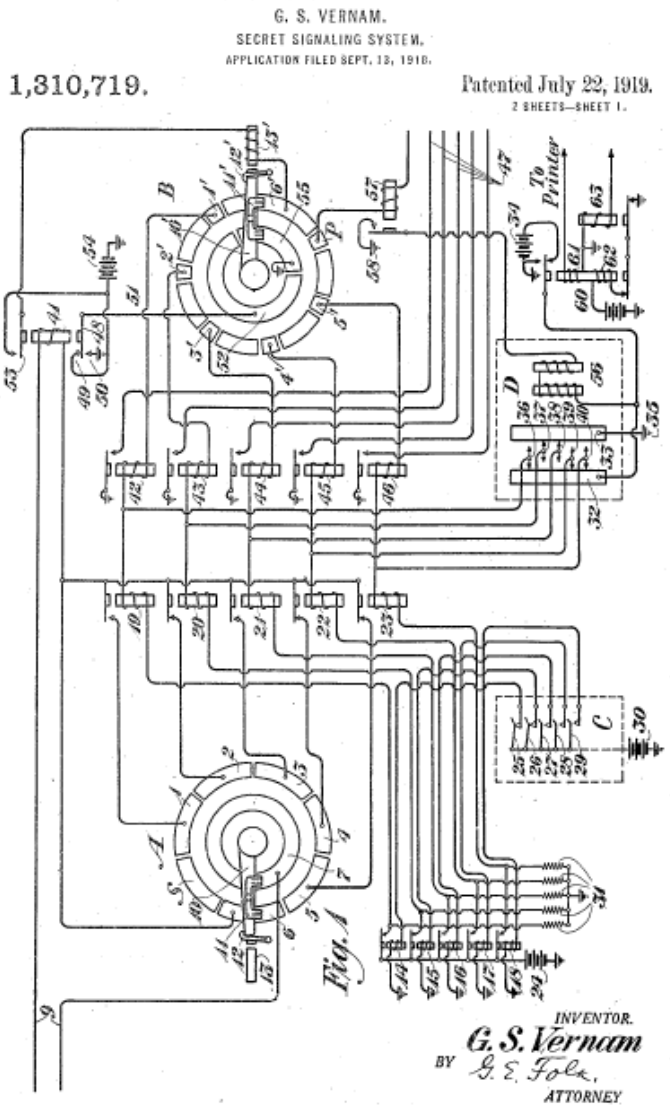
იმდენჯერ, რასაც მოითხოვს ტექსტი. საინტერესოა ალგორითმი, რომელიც მიიღო ვერნამმა.ზემოთ განხილულ ალგორითმებს აერთიანებს საერთო იდეა, რომლის მიხედვით ტექსტის გარდაქმნისათვის (ანუ, საზოგადოდ, ტექსტის სიმრავლის სხვა სიმრავლეზე ასახვისათვის) გამოყენებულია გადანაცვლების ან ჩანაცვლების ჩვეულებრივი მათემატიკური ოპერაციები.

გ. ვერნამის ალგორითმი

ეს ალგორითმი წარმოადგენს ცეზარის და ვიჟინერის ალგორითმების შემდგომ განვითარებას (1917). შესაძლოა გვეფიქრა, რომ სიტყვა "განვითარების" ხმარება აქ თავისი სრული მნიშვნელობით არც არის გამართლებული, იმდენად ორიგინალურია ალგორითმის მიღება. მაგრამ, თუ გავითვალისწინებთ, რომ ვერნამის ალგორითმი არის ერთადერთი სრულყოფილი ალგორითმი, რომელიც აკმაყოფილებს კ.შენონის მიერ შემუშავებულ კრიტერიუმებს, გასაგები გახდება ალგორითმის მნიშვნელობა.

ვერნამის შიფრის სიმარტივე იმაში მდგომარეობს, რომ, თუ ცეზარის ალგორითმში

გასაღების სიგრძეა $l = 1$, ხოლო ვიჟინერის ალგორითმში გასაღების სიგრძე გარკვეული სიდიდეა ($l > 1$), რომელიც, საზოგადოდ, გაცილებით ნაკლებია ტექსტის სიგრძეზე, და მეორდება შიფრაციის დროს, ვერნამის ალგორითმში გასაღების სიგრძე ტექსტის სიგრძის ტოლია და გამოიყენება მხოლოდ ერთჯერადად.



ნახაზი- 6 გილბერტ ვერნამის სისტემის პატენტი

ალგორითმი შეიძლება შემდეგი სახით ჩავწეროთ. ვთქვათ, ანბანის ასო-ნიშნების, ციფრების, სასვენი და სხვა ნიშნების ერთობლიობა შეადგენს $a_i \in A$ სიმბოლოების სიმრავლეს, რომლის სიმძლავრეა N. ამ სიმრავლის ყოველ ასო-ნიშანს შევუსაბამოთ m_i რიცხვი; $m_i \in \{0,..,N-1\}$. დავუშვათ, რომ მოცემული M ღია ტექსტური $a_1, a_2, a_3,..$ შეტყობინება, C სპეციალური $c_1, c_2, c_3, \ldots$ ტექსტით (გასაღებით) დაშიფვრის შემდეგ ღებულობს B შიფროტექსტის, ანუ $b_1, b_2, b_3,..$ მიმდევრობის სახეს. M ტექსტის a_{vi} სიმბოლოს შეესაბამება m_{vi} რიცხვითი მნიშვნელობა, C გასაღების c_{ui} სიმბოლოს m_{ui} მნიშვნელობა, ხოლო B შიფროტექსტის b_{wi} სიმბოლოს m_{wi} მნიშვნელობა, სადაც $v_i, u_i, w_i \in \{0, \ldots, N-1\}$. მაშინ თითოეული სიმბოლოს დაშიფვრა და გაშიფრვა.

$$m_{vi} + m_{ui} = m_{wi}(mod\ N)$$
$$m_{wi} - m_{ui} = m_{vi}(mod\ N)$$

შესაბამისობებით განხორციელდება.

ცხადია, რომ ყოველი დაშიფვრის და გაშიფვრის შემდეგ C სპეციალური სიმბოლოების ტექსტი, რომელიც მხოლოდ გადამცემ და მიმღებ მხარეებს გააჩნია, უნდა განადგურდეს.

XIX საუკუნეში ჰოლანდიელმა კერხოფმა

ჩამოაყალიბა კრიპტოგრაფიის ძირითადი წესი: შიფრის მედეგობა უნდა განისაზღვრებოდეს მხოლოდ გასაღების საიდუმლოობით. ინფორმაციის გამტაცებელს ან კრიპტოანალიტიკოსს შეუძლია, იცოდეს ყველა მონაცემი, გარდა გასაღებისა. ითვლება, რომ კრიპტოსისტემა გახსნილია, თუ გამტაცებელს დასაშვებზე მეტი ალბათობით შეუძლია შემდეგი ოპერაციების ჩატარება: საიდუმლო გასაღების პოვნა, გარდაქმნის ალგორითმის ეფექტური შესრულება, რომელიც ფუნქციონალურად ექვივალენტურია საწყისი კრიპტოალგორითმისა. იმისათვის, რომ კრიპტოსისტემა გახსნილად ჩაითვალოს, საჭიროა არა მხოლოდ გასაღების გახსნის (ანუ საიდუმლო გასაღების პარამეტრების მიღების) ალგორითმის ჩვენება, არამედ იმის ჩვენებაც, რომ ეს ალგორითმი შეიძლება შესრულდეს რეალურ დროში. ალგორითმის გახსნის სირთულე ითვლება კრიპტოსქემის ერთ-ერთ მთავარ მახასიათებლად და მას კრიპტომედეგობა ეწოდება.

ფორმალური კრიპტოგრაფიის ბოლო სიტყვა, რომელიც საკმაოდ მაღალი კრიპტომედეგობას იძლეოდა როტორული კრიპტოგრაფია (ავტომატიზირების მარტივი

მეთოდები). პირველი შესაბამისი სისტემა გამოიგონა 1790 წელს მომავალმა ამერიკის პრეზიდენტმა თომას ჯეფერსონმა.

მეოცე საუკუნის დასაწყისიდან კრიპტოგრაფიული პროცესების ავტომატიზაციის მიზნით გამოიგონეს მოწყობილობები (მანქანები), რომლებშიც გამოყენებულია მბრუნავი როტორები. როტორს აქვს დისკოს ფორმა. მასზე სათანადო პოზიციებში ასო-ნიშნების განთავსება (ანუ ტექტის ჩაწერა) და როტორების გარკვეული წესით შეერთება იწვევს ანბანის ასო-ნიშნების გადანაცვლება-ჩანაცვლებას, როგორც ეს ხდება ვიჟენერის ალგორითმში.

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

ნახაზი- 7 ვიჟენერის ცხრილი

განვიხილოთ მაგალითი #3

საწყისი	G	E	O	R	G	I	A
ტექსტი:							
გასაღები:	S	O	U	S	O	U	S
დაშიფრული	Y	S	I	J	U	C	S

ტექსტი:

საწყისი ტექსტი:

”G”- 6 ; “E”- 4 ; “O” – 14; “R” – 17; “G” – 6; “I” – 8; “A”- 0;

გასაღები:

“S” -18; “O” -14; “U”- 20;

დაშიფრული ტექსტი:

”G” +“S” $\rightarrow 6 + 18 = 24 \rightarrow$ “Y”;

“E” +”O” $\rightarrow 4 + 14 = 18 \rightarrow$ “S”

“O”+”U” $\rightarrow 14 + 20 = 34 \rightarrow 34(mod26) \equiv 8 \rightarrow$”I”

“R”+”S” $\rightarrow 17 + 18 = 35 \rightarrow 35(mod26) \equiv 9 \rightarrow$ “J”

“G”+”O” $\rightarrow 6 + 14 = 20 \rightarrow$ “U”

“I”+”U” $\rightarrow 8 + 20 = 28 \rightarrow 28(mod26) \equiv 2 \rightarrow$ “C”

“A”+”S” $\rightarrow 0 + 18 = 18 \rightarrow$ “S”

საწყისი ტექსტი:	6	4	14	17	6	8	0
გასაღები:	18	14	20	18	14	20	18
დაშიფრული ტექსტი:	24	18	34	35	20	28	18

განვიხილოთ მაგალითი #4

საწყისი ტექსტი:	A	L	G	O	R	I	T	H	M
გასაღები:	G	E	O	G	E	O	G	E	O
დაშიფრული ტექსტი:	G	P	U	U	V	W	Z	L	A

საწყისი ტექსტი:

“A”- 0; “L” – 11; “G” -6; “O” -14; “R” -17; “I”- 8; “T”- 19; “H”- 7 ; “M”- 12;

გასაღები:

”G”- 6 ; “E”- 4 ; “O” – 14;

დაშიფრული ტექსტი:

”A” +“G” $\rightarrow 0 + 6 = 6 \rightarrow$ “G”;

“L” + “E” $\rightarrow 11 + 4 = 15 \rightarrow$”P”

“G” + “O” $\rightarrow 6 + 14 = 20 \rightarrow$ “U”

“O” + “G” $\rightarrow 14 + 6 = 20 \rightarrow$ “U”

“R” + “E” $\rightarrow 17 + 4 = 21 \rightarrow$ “V”

“I” + “O” $\rightarrow 8 + 14 = 22 \rightarrow$ “W”

“T” + “G” $\rightarrow 19 + 6 = 25 \rightarrow$ “Z”

“H”+ “E” $\rightarrow 7 + 4 = 11 \rightarrow$ “L”

“M”+ “O” $\rightarrow 12 + 14 = 26 \rightarrow 26(mod26) \equiv 0 \rightarrow$ “A”

საწყისი ტექსტი:	0	11	6	14	17	8	19	7	12
გასაღები:	6	4	14	6	4	14	6	4	14
დაშიფრული ტექსტი:	6	15	20	20	21	22	25	11	26

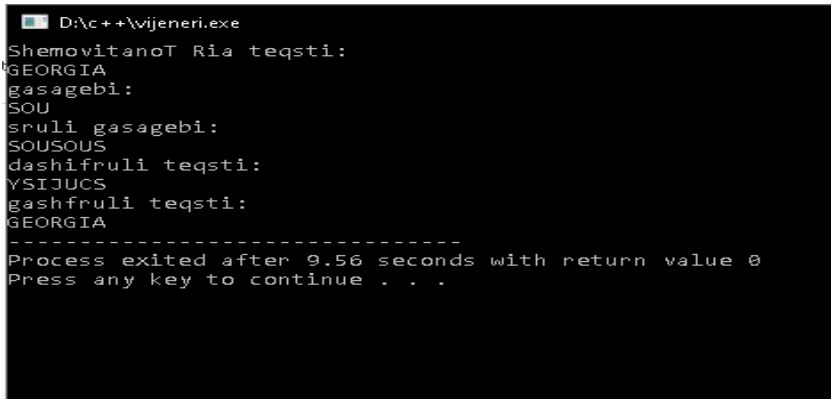
ვიჟენერის ალგორითმის პროგრამული რეალიზაცია:

```
#include<iostream>
#include<string>
#include<math.h>
using namespace std;
main()
{string gia_texti, shifro_texti, gashifruli_texti, kay, sruli_kay="";
cout<<"ShemovitanoT Ria teqsti:"<<'\n';
cin>>gia_texti;
cout<<"gasagebi:";
cin>>kay;
int n=gia_texti.length(); //ღია ტექსტის სიგრძე;
int l=kay.length(); //გასაღების სიგრძე;
if(n>l)
```

```
{
for(int i=0;i<n/l;i++)
//ღია ტექსტის სიგრძისა და
//გასაღების სიგრძის გაყოფის
   sruli_kay+=kay;
  //შედეგად მიღებულ რიცხვჯერ
 //ხდება გასაღების დამატება
        if(n%l!=0) for(int i=0;i<n%l;i++)
//თუ სიგრძეების გაყოფის შედეგად რჩება ნაშთი
        sruli_kay+=kay[i];
//სრულ გასაღებს ვამატებთ გასაღებიდან ნაშთის
//ტოლ სიმბოლოს.

}
else
if(n<l)  for(int i=0;i<n;i++)
//თუ ღია ტექსტის სიგრძე ნაკლებია საწყისი
//გასაღების სიგრძეზე
      sruli_kay[i]+=kay[i];
//სრული გასაღებისთვის საწყისი გასაღებიდან
//ვიღებთ ტექსტის სიგრძის
//(n) რაოდენობის სიმბოლოებს
 cout<<"sruli gasagebi:"<<'\n'<<sruli_kay<<'\n';
  cout<<"dashifruli teqsti:"<<'\n';
```

```
//დაშიფვრა
for(int i=0;i<n;i++)
{
shifro_texti[i]=(gia_texti[i]+sruli_kay[i]  -2*'A')%26  +
'A';
cout<<shifro_texti[i];}
cout<<'\n'<<"gashfruli teqsti:"<<'\n';
for(int i=0;i<n;i++)
  //გაშიფვრა
{
gashifruli_texti[i]=(shifro_texti[i]-sruli_kay[i])%26   +
'A';
if(gashifruli_texti[i]<'A') gashifruli_texti[i]+=26;
cout<<gashifruli_texti[i];}
}
```

შედეგი:



კრიპტოანალიზი:

ვიჟენერის შიფრის კრიპტოანალიზის დროს უპირველესი ამოცანაა დაშიფვრის დროს გამოყენებული გასაღების სიგრძის პოვნა. ამ მიზნით შეიძლება გამოვიყენოთ ე.წ. თანხვედრის ინდექსი. თანხვედრის ინდექსი - ეს არის ორი ტექსტიდან ნებისმიერად აღებული, ასოს დამთხვევის ალბათობის რიცხვი. ნებისმიერი ტექსტისათვის თანხვედრის რიცხვი გამოითვლება ფორმულით:

$$I(\vec{x}) = \sum_i f_i \frac{f_i - 1}{n(n-1)}$$

სადაც f_i წარმოადგენს ანბანის i-ური ასოს რაოდენობას ტექსტში, ხოლო n- ტექსტის ასოების რაოდენობაა. ინგლისური ენისათვის თანხვედრის

ინდექსი 0.0667-ის ტოლია, ხოლო შემთხვევითად აკრეფილი სიმბოლოებისათვის ის 0.038-ს უდრის. უფრო მეტიც, ერთი ანბანით (მაგ., ინგლისური ასოებით) დაშიფრული ტექსტისათვის თანხვედრის რიცხვი ისევ 0.0667-ის ტოლია. ეს იხსნება იმ ფაქტით, რომ სხვადასხვა ტექსტში განსხვავებული ასოების რაოდენობა ერთი და იგივეა.

სწორედ ეს თვისება გამოიყენება ვიჟენერის შიფრის გასაღების სიგრძის პოვნის შემთხვევაში. შიფრორტექსიდან თანმიმდევრულად უნდა შევარჩიოთ ყოველი მეორე ასო და მიღებული ტექსტისათვის უნდა დავითვალოთ თანხვედრის ინდექსი. თუ ამ ტექსტის დამთხვევების ინდექსი ახლოსაა ბუნებრივი ენის თანხვედრის ინდექსთან, მაშინ ეს ნიშნავს, რომ გასაღების სიგრძე 2-ის ტოლია. წინააღმდეგ შემთხვევაში, შიფრო-ტექსტიდან ისევ უნდა ავირჩიოთ უკვე ყოველი მესამე ასო და მისთვის დავთვალოთ თანხვედრის ინდექსი. ეს პროცესი გრძელდება მანამ, სანამ თანხვედრის ინდექსის რაიმე დიდი მნიშვნელობა გასაღების სიგრძეს არ მიგვითითებს.

ამ მეთოდის სისწორე განისაზღვრება იმ ფაქტით, რომ თუ გასაღების სიგრძე სწორადაა

ამოცნობილი, მაშინ შერჩეული ასოებისაგან შემდგარი შიფროტექსტი ცეზარის შიფრითაა დაშიფრული. მისი თანხვედრის ინდექსი დაახლოებით ბუნებრივი ენის (მაგ., ინგლისურის) თანხვედრის ინდექსის ტოლია. მას შემდეგ, რაც გასაღების სიგრძე ნაპოვნია, ვიჟენერის შიფრის გატეხვა დაიყვანება ცეზარის რამდენიმე შიფრის გატეხვაზე.

დავალება: განახორციელეთ ტექსტის შიფრაცია - დეშიფრაცია ვიჟენერის ალგორითმის გამოყენებით.

საწყისი ტექსტი: VIGENERECIPHER

გასაღები: EDU

საწყისი ტექსტი:	V	E	R	N	A	M	C	I	P	H	E	R
	21	4	17	13	0	12	2	8	15	7	4	17
გასაღები:	Z	Y	X	W	V	U	T	S	R	Q	P	O
	25	24	23	22	21	20	19	18	17	16	15	14
	46	28	40	35	21	32	21	26	32	23	19	31
დაშიფრული ტექსტი:	U	C	O	J	V	G	V	A	G	X	T	F

განვიხილოთ მაგალითი #5

განახორციელეთ ტექსტის შიფრაცია - დეშიფრაცია ვერნამის ალგორითმის გამოყენებით.

საწყისი ტექსტი: VERNAMCIPHER
გასაღები: ZYXWVUTSRQPO

$46(mod26) \equiv 20 \rightarrow$ U
$28(mod26) \equiv 2 \rightarrow$ C
$40(mod26) \equiv 14 \rightarrow$ O
$35(mod26) \equiv 9 \rightarrow$ J
$32(mod26) \equiv 6 \rightarrow$ G
$26(mod26) \equiv 0 \rightarrow$ A
$32(mod26) \equiv 6 \rightarrow$ G
$31(mod26) \equiv 5 \rightarrow$ F

განვიხილოთ მაგალითი #6
განახორციელეთ ტექსტის შიფრაცია - დეშიფრაცია

საწყისი ტექსტი:	I	N	F	O	R	M	A	T	I	O	N
	8	13	5	14	17	12	0	19	8	14	13
გასაღები:	S	E	C	U	R	I	T	Y	G	E	O
	18	4	2	20	17	8	19	24	6	4	14
	26	17	7	34	34	20	19	43	14	18	27
დაშიფრული ტექსტი:	A	R	H	I	I	U	T	R	O	S	B

ვერნამის ალგორითმის გამოყენებით.

საწყისი ტექსტი: INFORMATION
გასაღები: SECURITYGEO

$26(mod26) \equiv 0 \rightarrow$ A
$34(mod26) \equiv 8 \rightarrow$ I
$43(mod26) \equiv 17 \rightarrow$ R
$27(mod26) \equiv 1 \rightarrow$ B

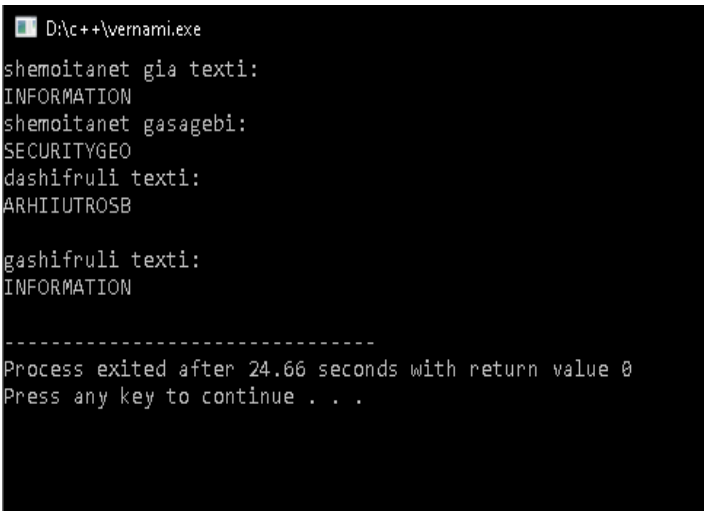
ვერნამის ალგორითმის პროგრამული რეალიზაცია:

```
#include<iostream>
using namespace std;
class vernami
{
        public:
           string str,kay;
           string enc, dec;
           vernami(string a, string b)
           {
                str = a;
                kay = b;
           }
           void encryption()
```

```
        {
            int i;
            for(i=0;i<str.size();i++)
                enc[i]=(str[i]+kay[i]-2*'A')%26+'A';
//დაშიფვრის მეთოდი
        }
        void decryption()
        {
            int i;
            for(i=0;i<str.size();i++)
                {
                dec[i]=(enc[i]-kay[i])%26+'A';
//გაშიფრვის მეთოდი
                if(dec[i]<'A') dec[i]+=26;}
        }
        void printenc()
        {
            int i;
            for(i=0;i<str.size();i++)
  // დაშიფრული ტექსტის ბეჭდვა
                cout<<enc[i];
            cout<<endl;
        }
        void printdec()
```

```
        {
            int i;
            for(i=0;i<str.size();i++)
  //გაშიფრული ტექსტის ბეჭდვა
                cout<<dec[i];
            cout<<endl;
        }
};
int main()
{
    string str,kay;
    cout<<"shemoitanet gia texti:"<<endl;
    getline(cin,str);
    cout<<"shemoitanet gasagebi:"<<endl;
    getline(cin,kay);

    vernami io(str,kay);
        io.encryption();
      cout<<"dashifruli texti:"<<endl;
    io.printenc();
    cout<<endl;
    io.decryption();
    cout<<"gashifruli texti:"<<endl;
    io.printdec();}
```

შედეგი:



ვერნამის ალგორითმი პროგრამული რეალიზაცია:

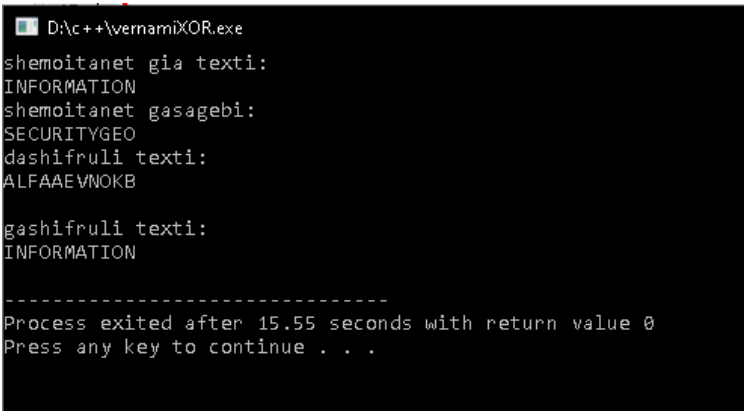
```
#include<iostream>
using namespace std;
class vernam
{
    public:
        string str,kay; //სტრიქონი, გასაღები
        string enc, dec;
//დაშიფრული და გაშიფრული სტრიქონები
        vernam(string a, string b)
        {
```

```
        str = a;
        kay = b;
    }
    void encryption() //დაშიფვრა
    {
        int i;
        for(i=0;i<str.size();i++)
            enc[i] = str[i]^kay[i];
    }
    void decryption()    //გაშიფვრა
    {
        int i;
        for(i=0;i<str.size();i++)
    dec[i] =enc[i]^kay[i];
    }
    void printenc()
//დაშიფრული ტექსტის ბეჭდვა
    {
        int i;
        for(i=0;i<str.size();i++)
            {
   char c =enc[i]%26+'A';
            cout<<c;}
            cout<<endl;
```

```
        }
        void printdec()
 // გაშიფრული ტექსტის ბეჭდვა
        {
            int i;
            for(i=0;i<str.size();i++)
                                cout<<dec[i];
                            cout<<endl;
        }
};
int main()
{
    string str,kay;
    cout<<"shemoitanet gia texti:"<<endl;
    getline(cin,str);
    cout<<"shemoitanet gasagebi:"<<endl;
    getline(cin,kay);
    vernam io(str,kay);
    io.encryption();
    cout<<"dashifruli texti:"<<endl;
    io.printenc();
    cout<<endl;
    io.decryption();
    cout<<"gashifruli texti:"<<endl;
```

```
    io.printdec();
}
```

შედეგი:



კრიპტოანალიზი: ვერნამის შიფრი არის სრულყოფილი შიფრი. თუ გასაღები არ მეორდება და გასაღების სიგრძე არის შეტყობინების ტოლი, მაშინ შიფრი არის გაუხსნადი. შემთხვევითი გასაღებისა და ღია ტექსტის xor-ით შეკრება ქმნის სრულიად შემთხვევით შიფროტექსტს, რომელსაც ვერანაირი ძალისმერი შეტევა ვერ გატეხავს. ამ სისტემის უარყოფითი მხარე არის ის, რომ გადაცემული ინფორმაციის ყოველი ბიტისთვის წინასწარ უნდა მომზადდეს გასაღების თითოეული შემთხვევითი ბიტი. დიდი

რაოდენობით მონაცემების დაშიფვრისას ეს ქმნის პრობლემას. ამიტომ, ეს სისტემა გამოიყენება მხოლოდ საიდუმლო შეტყობინებების გადასაცემად.

დავალება: განახორციელეთ ტექსტის შიფრაცია - დეშიფრაცია ვერნამის ალგორითმის გამოყენებით.
საწყისი ტექსტი: SOKHUMYSTATEUNIVERSITY
გასაღები: ZYXWVUTSRQPONMLKJIHGFE

თავი II. სიმეტრიული კრიპტოსისტემები

სიმეტრიული ეწოდება კრიპტოსისტემას, რომელშიც როგორც ინფორმაციის დასაშიფრად, ასევე გასაშიფრადაც გამოიყენება ერთი და იგივე გასაღები:

$$E_k(P) = C$$
$$D_k(C) = P,$$

სადაც E –დაშიფვრის ფუნქცია, k –გასაღები, P – ღია ტექსტი, C – შიფროტექსტი, D – გაშიფვრის ფუნქცია.
ჭეშმარიტია შემდეგი ტოლობა:

$$D_k(E_k(P)) = P.$$

სიმეტრიული ალგორითმები შედგება ორი ტიპის შიფრებისგან: ბლოკური და ნაკადური შიფრები.

ბლოკური შიფრები

ბლოკური შიფრები ეწოდებათ ისეთ შიფრებს, რომლებშიც ღია ტექსტი იყოფა ბლოკებად (გარკვეული სიგრძის სტრიქონებად) თანამედროვე შიფრებში ბლოკის სიგრძე ტოლია 2^n, სადაც $n>=8$. ერთდროულად ხდება თითოეული ბლოკის

დაშიფვრა და არხში გადაიცემა დაშიფრული ბლოკი:

$y = E_k(x)$ და $x = D_k(y)$

სადაც x , y – ღია და დაშიფრული ტექსტების ბლოკებია; k – შიფრის საიდუმლო გასაღებია;

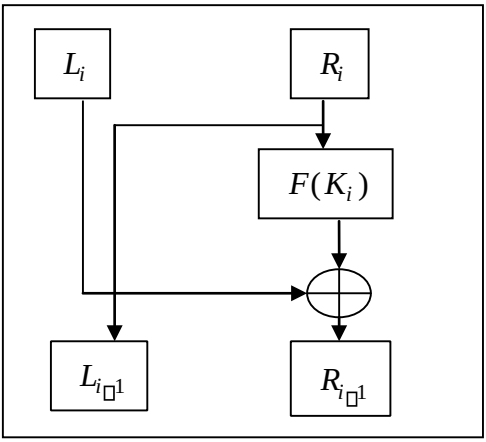
E, D – დაშიფვრისა და გაშიფვრის ფუნქციებია (ბლოკების დაშიფვრისა და გაშიფვრის ალგორითმებია).

კლოდ შენონმა თავის ფუნდამენტურ ნაშრომში ჩამოაყალიბა მედეგი ბლოკური შიფრე-ბის დამახასიათებელი ორი ძირითადი თვისება: მიმოფანტვა და დიფუზია. მინოფანტვა არის გარდაქმნა, რომლის მიზანია დამალოს კავშირი გასაღებსა და შიფროტექსტს შორის. დიფუზია არის გარდაქმნა, რომლის დროსაც შიფროტექსტის სიმბოლო დამოკიდებულია ღია ტექსტის რაც შეიძლება უფრო მეტ სიმბოლოზე, ღია ტექსტის სტრუქტურია დასამალად. თანამედროვე ბლოკურ შიფრებში გამოიყენება როგორც მიმოფანტვა, ასევე დიფუზიის ოპერაციები. თუ ღია ტექსტი წარმოადგენს n სიგრძის ბიტურ სტრიქონს, მაშინ ამ ოპერაციათა გამოყენების შედეგად მივიღებთ n სიგრძის ბიტურ ფსევდოშემთხვევით მიმდევრობას. ბლოკურ შიფრებში ერთი და იგივე ოპერაციების გამოყენება რამდენჯერმე ხდება, ამიტომაც თანამედროვე ბლოკურ შიფრებს იტერაციულ შიფრებსაც უწოდებენ, ხოლო თითოეულ იტერაციას კი - რაუნდს. ყველა

ბლოკური შიფრი შედგება ორი ძირითადი ნაწილისგან: ერთ რაუნდში ჩასატარებელი ოპერაციების კომბინაციებისგან და საწყისი გასაღებიდან რაუნდული გასაღების გამომუშავების სქემისგან. რაუნდული გასაღებების შექმნა დაშიფვრისა და დეშიფრაციის პარალელურად ამცირებს ალგორითმის მიერ გამოყენებული მეხსიერების მოცულობას. თანამედროვე კრიპტოგრაფიაში არსებობს დიმეტრიული შიფრების გამოყენების რამდენიმე ძირითადი კონსტრუქცია: ფეისტელის, ლაი-მესისის, SP(ჩანაცვლება-გადანაცვლების და კვადრატული არქიტექტურები).

- ფეისტელის სქემა:

ფეისტელის სქემაში დასაშიფრი ბლოკი იყოფა ორ ნაწილად: მარცხენა (L) და მარჯვენა (R) ნაწილებად. ხდება მარჯვენა ნაწილის გარკვეული ფუნქციით გარდაქმნა და შემდეგ XOR-ით შეკრება მარცხენა ნაწილთან. მიღებული შედეგი წარმოადგენს შემდეგი რაუნდის მარჯვენა ნაწილს, ხოლო შემდეგი რაუნდის მარცხენა ნაწილს წარმოადგენს წინა რაუნდის მარჯვენა ნაწილი (გარდაქმნამდე არსებული ნაწილი).



ნახაზი- 8 ჰ.ფეისტელის სქემა

მათემატიკურად ეს სქემა აღიწერება შემდეგი სახით:

$L_{i+1} = R_i;$

$R_{i+1} = L_i \oplus f_k(R_i).$

თითოეული რაუნდისათვის სქემა შებრუნებადია ისე, რომ საჭირო არ არის დამშიფრავი ფუნქციის შებრუნება

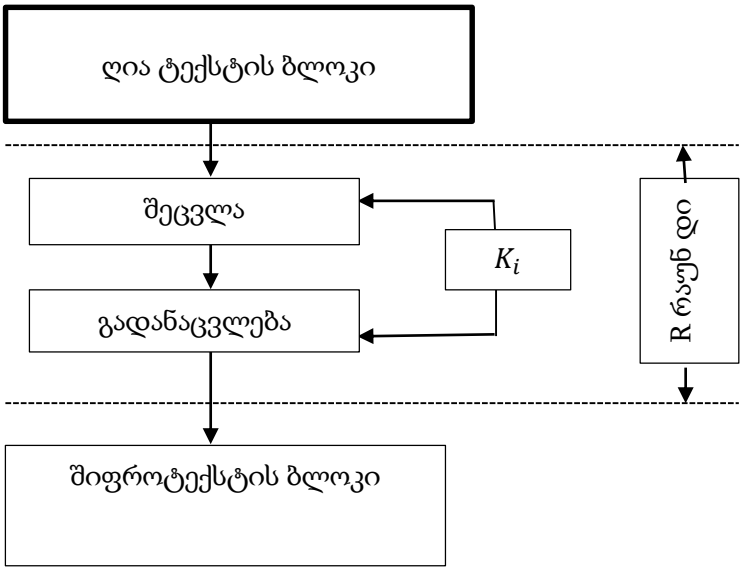
$R_i = L_{i+1};$

$L_i = R_{i+1} \oplus f_k(L_{i+1}).$

ფეისტელის სქემით ინფორმაციის დაშიფვრისა და გაშიფვრის რეალიზაცია მარტივია როგორც პროგრამულ, ასევე აპარატურულ დონეზე, ამან განაპირობა ამ სქემის ასე ფართო გამოყენება. ფეისტელის პრინციპზე დაყრდნობით შექმნილია მრავალი ბლოკური ალგორითმი: DES, ГОСТ 28147-89, RC5, BLOWFISH, TEA, CAST-128 და ა.შ.

- **გადანაცვლება-ჩანაცვლების სქემა (SP სქემა):**

ფეისტელის სქემისგან განსხვავებით SP სქემით ერთი რაუნდის განმავლობაში ხდება მთლიანი ბლოკის დაშიფვრა. ეს სქემა დაიყვანება მონაცემთა შეცვლასა და გადანაცვლებაზე, რომელიც დამოკიდებულია გასაღებზე.



ნახაზი- 9 SP სქემა

SP-სქემა უფრო ნაკლებადაა გავრცელებული ვიდრე ფეისტელის სქემა. SP-სქემაზე დაფუძნებული ალგორითმებია: SERPENT, SAFER+.

- კვადრატული არქიტექტურის შიფრე-ბისთვის დამახასიათებელია დაშიფრული მონაცემთა ბლოკის წარმოდგენა ორგან-ზომილებიანი ბაიტის მასივის სახით.

კრიპტოგრაფიული გარდაქმნები შეიძლება განხორციელდეს მასივის ცალკეულ ბაიტებზე, ასევე მის სტრიქონებზე ან სვეტებზე. ალგორითმის სტრუქტურამ სახელი მიიღო **კვადრატული ალგორითმიდან,** რომელიც 1996 წელს შეიმუშავეს ვინსენტ რაიჯმენმა და ჯოან დემენმა, Rijndael-ის ალგორითმის მომავალმა ავტორებმა, რომელსაც ასევე აქვს კვადრატის მსგავსი სტრუქტურა, რომელიც ლიტერატურაში იწოდება AES-ით. კვადრატული არქიტექტურის ალგორითმს წარმოადგენს ამავე ავტორების მიერ ადრეულ წლებში შექმნილი ალგორითმი SHARK და ალგორითმი CRYPTON, რომელიც 1998 წელს სამხრეთ კორეელი კრიპტოლოგის ჩე ჰონგ ლიმის მიერ შეიქმნა.

ნაკადური შიფრები

ნაკადური შიფრები ეწოდებათ ალგორითმებს, რომლებიც შიფრავენ ალგორითმში შემოსული ინფორმაციის თითოეულ ბიტს იმ მიმდევრობით, რა მიმდევრობითაც შემოდიან ალგორითმში და ამავე თანმიმდევრობით გადაიცემა არხში თითოეული ბიტი დაშიფვრის შემდეგ. ნაკადური შიფრები გამოიყენება სატელეკომუნიკაციო არხებში ინფორმაციის გადასაცემად. მორბენალი გასაღების გენერატორი (**running key generator)** გამოიმუშავებს ბიტების ნაკადს $k_1, k_2, \ldots, k_i, \ldots$, რომელსაც უწოდებენ გამას. ეს გამა და ღია ტექსტის $m_1, m_2, \ldots, m_i, \ldots$ ბიტები იკრიბება ოპერაცია **XOR**-ით და მიიღება შიფროტექსტი: $C_i = m_i + k_i$. დეშიფრაციის დროს კვლავ იკრიბება უკვე შიფროტექსტი გამასთან და მიიღება ღია ტექსტი.

ასეთი შიფრის საიმედოობა მთლიანად დამოკიდებულია გენერატორის მიერ გამომუშავებულ გამაზე, რაც უფრო ახლოს იქნება ეს გამა ნამდვილად შემთხვევით მიმდევრობასთან, მით უფრო გაუჭირდება კრიპტოანალიტიკოსს მისი გატეხვა. ასეთი შიფრებს გააჩნიათ განსაკუთრებული მგრძნობელობა ყოველი შეცდომის მიმართ. საკმარისია გადაცემის დროს დაიკარგოს ან ჩაემატოს ერთი ბიტი ინფორმაცია, რომ გაშიფრვა გახდება შეუძლებელი, ამიტომ აუცილებელია, გადამცემი და მიმღები

გენერატორების სინქრონიზაცია, რაც წარმოადგენს საკმაოდ რთულ პრობლემას. არსებობს ამ პრობლემის გადაჭრის ორი გზა. პირველი - ესაა ისეთი სპეციალური პროცედურების გათვალისწინება რეჟიმში, რომლებიც საშუალებას მოგვცემენ მოვახდინოთ დამშიფრავი და გამშიფრავი გენერატორების სინქრონიზაცია. ასეთ შიფრებს უწოდებენ სინქრონიზებად ნაკადურ შიფრებს. ასეთი შიფრების უპირატესობა მდგომარეობს იმაში, რომ არ ხდება შეცდომების გავრცელება, შეცდომა ერთ ბიტში (ბიტის შეცვლა და არა დაკარგვა, რაც უფრო ხშირადაა მოსალოდნელი) არ გამოიწვევს შეცდომას მომდევნო ბიტებში. ამ პრობლემის გადაჭრა შეიძლება თვითსინქრონიზებადი ნაკადური შიფრის საშუალებითაც. ამ დროს გამის თითოეული ბიტი წარმოადგენს რაიმე წინა ბიტების ფუნქციას. რადგანაც გენერატორის მდგომარეობა მთლიანად დამოკიდებულია წინა ბიტზე, გამშიფრავი გენერატორი მიიღებს რა n ბიტს ავტომატურად მოვა სინქრონიზაციაში დამშიფრავ გენერატორთან. ასეთი გენერატორების შემთხვევაში შეცდომა ერთ ბიტში გამოიწვევს შეცდომის გავრცელებას n ბიტზე, რის შემდეგ გენერატორები კვლავ მოდიან სინქრონიზაციაში.

კრიპტოსისტემა AES RIJNDAEL

განვიხილოთ ბლოკურ შიფრებს შორის ერთ-ერთი ყველაზე პოპულარული და თანამედროვე გამოყენებადი ალგორითმი AES, რომელიც ახალ სტანდარტად იქნა დამტკიცებული RIJNDAEL-is სახელწოდებით. RIJNDAEL-ი1998 წელს შეიქმნა ბელგიელი კრიპტგრაფების იოან დამენისა და ვინსენტ რაიჯმენის მიერ. მისი ოფიციალური სახელწოფებაა AES RIJNDAEL, შემდეგში მას უწოდებენ AES-ს (Advanctd Encription Standard).

AES-ში ყველა ოპერაცია, რომელიც სრულდება დასაშიფრ ბლოკზე, ეფუძნება არითმეტიკას $FG(2^8)$ ველში, AES-ში რიცხვების წარმოდგენა ხდება რვაბიტიანი (ბაიტიანი) ვექტორების სახით. რომელიღაც ბაიტი **b** შედგება $b_7, b_6, \ldots b_0$ ბიტებისაგან, ის წარმოადგენს პოლინომს კოეფიციენტებით {0,1} სიმრავლიდან, რომელსაც აქვს შემდეგი სახე:

$$b_7x^7 + b_6x^6 + \cdots + b_0$$

ამასთან, კომპიუტერში ინახება მხოლოდ ვექტორი, რომელიც შედგენილია ამ პოლინომის კოეფი-ციენტებისაგან. მისი აღქმა ხდება თექვსმეტობითი რიცხვის სახით. მაგალითად, ბაიტი, რომლის თექვსმეტობითი მნიშვნელობაა $(27)_{hex}$ და ორობითი მნიშვნელობა კი

1 1 100111, შეესაბამება პოლინომს

$$x^6 + x^4 + x^2 + x + 1.$$

ასეთი წარმოდგენის პირობებში ორი პოლინომის შეკრების ოპერაცია პრაქტიკულად წარმოადგენს მსგავსი წვერების კოეფიციენტების **XOR**-ით შეკრებას.

პოლინომურ წარმოდგენაში $FG(2^8)$ ველში რიცხვების გამრავლება შეესაბამება პოლინომების გამრავლებას რომელიმე მერვე ხარისხის დაუყვანად ბინარულ პოლინომზე მოდულით. **Rijndael**-ში ასეთ პოლინიმად აღებულია

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

რომელიც წარმოადგენს გამრავლების მიმართ ერთეულოვან ელემენტს $(01)_{hex}$. გამრავლების ოპერაცია სრულდება შემდეგნაირად: ორობით სისტემაში რიცხვის ორზე გამრავლება რიცხვის ერთი თანრიგით მარცხნივ წაძვრის ტოლფასია, საჭიროების შემთხვევაში, განვახორციელოთ XOR-ით შეკრება $(1B)_{hex}$-თან (ეს რიცხვი პოლინომურ წარმოდგენაში მოდულის ფუძის ტოლია).

Rijndael-ს საფუძვლად უდევს ე.წ. "კვადრატის" არქიტექტურა. AES- ის ალგორითმი მუშაობს 128,192 და 256 ბიტიან ბლოკებთან (თუმცა სტანდარტს წარმოადგენს 128 ბიტიანი ბლოკი). გასაღების სიგრძე ასევე შეიძლება იყოს 128,192 და 256 ბიტის სიგრძის, ამასთან შესაძლებელია სხვადასხვა სიგრძის ბლოკებთან

სხვადასხვა სიგრძის გასაღებების გამოყენება.

ალგორითმში რაუნდების (იტერაციების) რაოდენობა დამოკიდებულია დასაშიფრი ბლოკისა და გასაღების სიგრძეებზე. განვიხილოთ 128 ბიტიან ბლოკებთან მუშაობის ალგორითმი.

128 ბიტიანი გასაღების მქონე ალგორითმის აღწერა: საწყისი მონაცემი (ღია ტექსტი) იყოფა 16 ბაიტიან (128 ბიტიან) ბლოკებად, თუ ჯამური ბაიტის ზომა არ არის 16 ჯერადი, მაშინ მონაცემები ივსება 16-ის ჯერადი ბაიტის ზომამდე. ბლოკები წარმოდგენილია 4x4 მდგომარეობის[1] მატრიცის სახით.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

ამის შემდეგ ხორციელდება შემდეგი ნაბიჯები:

1. გასაღების გაფართოება- KeyExpansion;

[1] **Rijndael**-ში ყველა გარდაქმნა სრულდება ბაიტურ მატრიცაზე, რომელსაც ეწოდება მდგომარეობა. ეს მატრიცა შეიცავს ოთხ სტრიქონს, ხოლო სვეტების რაოდენო-ბა, რომელიც აღინიშნება Nb-თი, დამოკიდებულია ბლოკის სიგრძეზე და უდრის ბლოკის სიგრძე გაყოფილი 32-ზე, ამიტომ, **Rijndael**-ი შეიძლება მუშაობდეს 4×4, 4×6 და 4×8 ზომის მატრიცა.

2. საწყისი რაუნდი - შეკრება state (მდგომარეობთ მატრიცა) ძირითად გასაღებთან; დასაშიფრი ბლოკი **XOR** ოპერაციით იკრიბება გასაღებთან და ისე შედის მდგომარეობათა ბლოკში. ამასთან მიღებული ტექსტი (დავუშვათ 128 ბიტი) მდგომარეობათა მატრიცას ავსებს მარცხნიდან მარჯვნივ და ზევიდან ქვევით. ანუ მიმდევრობით:

3. $a_{0,0}, a_{1,0}, a_{2,0}, \dots a_{3,3}$.

3. დაშიფვრის 9 რაუნდი , სადაც თითოეულ რაუნდში ხდება შემდეგი გარდაქმნები:

- SubBytes
- ShiftRows
- MixColumns
- AddRoundKey

4. ბოლო რაუნდი შედგება შემდეგი გარ-დაქმნებისგან:

- SubBytes
- ShiftRows
- AddRoundKey

განვიხილოთ ზემოხსენებული ყველა გარდაქმნა:
SubBytes - ეს გარდაქმნა არის არაწრფივი ბაიტური გარდაქმნა, რომელიც ყოველი ბაიტისათვის სრულდება დამოუკიდებლად. ჩანაცვლების ცხრილი (ანუ **S**–ბლოკი) შებრუნებადია და კონსტრუირებულია ორი

გარდაქმნის კომპოზიციის სახით:

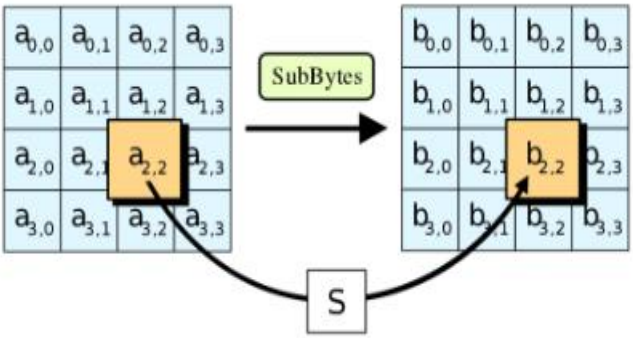
- ბაიტების შეცვლა S-box ცხრილის მიხედვით.
- თითოეული ბაიტის წარმოდგენა ხდება ორი ორობითი თექვსმეტობითი რიცხვის სახით b = (x,y), სადაც სადაც x განისაზღვრება b-ს 4 უფროსი თანრიგით და y- 4 უმცროსი თანრიგით. 16x16 ზომის S-box ცხრილი შეიცავს საწყისი ბაიტის ჩასანაცვლებელ მნიშვნელობებს: x სტრიქონის და y სვეტის გადაკვეთაზე არსებული b ბაიტი იცვლება b'-ით.

სხვანაირად ეს გარდაქმნა შეიძლება აღვწეროთ განტოლებით:

$$y_i = x_i \oplus x_{(i+4)\bmod 8} \oplus x_{(i+5)\bmod 8} \oplus x_{(i+6)\bmod 8} \oplus x_{(i+7)\bmod 8} \oplus c_i$$

სადაც x_i და y_i შესაბამისად i-ური ბიტის საწყისი და გარდაქმნილი მნიშვნელობებია $(i = 0,1,...,7)$, $c_0 = c_1 = c_5 = c_6 = 1$ და $c_2 = c_3 = c_4 = c_7 = 0$.

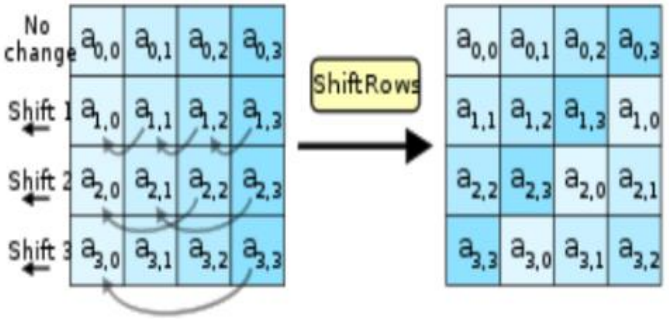
ამ გარდაქმნის შედეგად მიიღება მდგომარეობის ახალი მნიშვნელობა



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

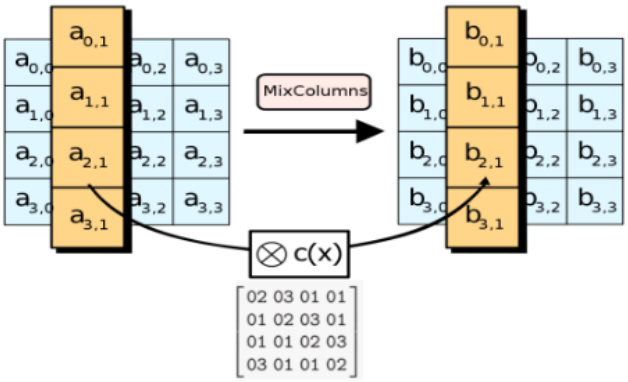
ნახაზი- 10 მდგომარეობათა მატრიცის გარდაქმნა S-box ცხრილის მიხედვით

- ShiftRows — მდგომარეობის მატრიცის სტრიქონების ციკლური ძვრა. ნულოვანი სტრიქონი რჩება თავის ადგილზე, პირველი სტრიქონის წაძვრა მოხდება ერთი ბაიტით მარცხნივ . მეორე სტრიქონის 2 ბაიტით მარცხნივ, მესამე სტრიქონის სამი ბაიტით მარცხნივ.



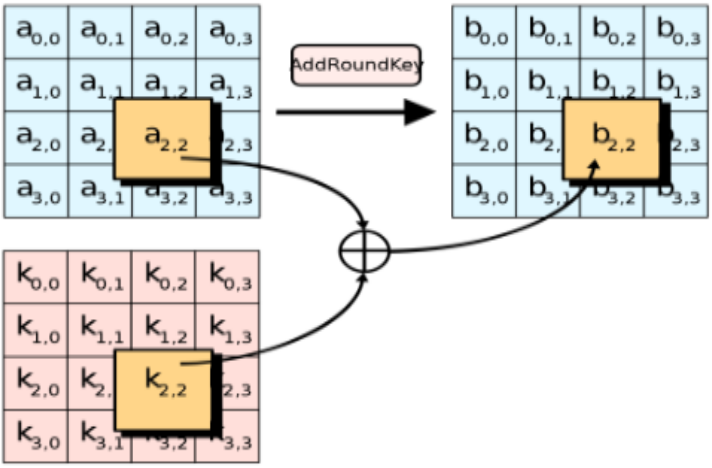
ნახაზი- 11 მდგომარეობის მატრიცის სტრიქონების ციკლური ძვრა

- MixColumns — მდგომარეობის მატრიცის თითოეული სვეტი მრავლდება ფიქსირებულ მატრიცაზე. ასეთი სახით ხდება მდგომარეობის მატრიცის სვეტების გარდაქმნა, სადაც გამრავლებისა და შეკრების ოპერაციები ხორციელდება ზემოთ აღწერილი წესების თანახმად.



ნახაზი- 12 მდგომარეობათა მატრიცის სვეტების გარდაქმნა

- AddRoundKey — რაუნდის გასაღებებისა და მდგომარეობის მატრიცის ელემენტების XOR-ით შეკრება.



ნახაზი- 13 XOR-ით შეკრების სქემა

- KeyExpansion — ძირითადი გასაღების გაფართოება რაუნდის გასაღებების შესაქმნელად, რომლებიც გამოიყენება რაუნდული დაშიფვრის დროს. გაფართოებული გასაღები შედგება 44 ბაიტური სიტყვისგან (Wi), აქედან გამომდინარე გასაღების მთლიანი ზომა არის 1408 ბიტი.
გაფართოებული გასაღების შექმნის ფუნქცია იყენებს Rcon მასივს და სრულდება შემდეგი მოქმედებები:

- გაფართოებული გასაღების პირველ 4 სიტყვაში ხდება ძირითადი გასაღების 4 სიტყვის გადატანა.
- თუ I რიცხვი უნაშთოდ იყოფა 4-ზე, მაშინ $SubBytes(RotByte(w_{i-1}))$ xor $Rcon_{i/4}$.
- წინააღმდეგ შემთხვევაში: $w_i = w_{i-4}$ xor w_{i-1}.

ფუნქცია RotByte ახორციელებს საწყისი სიტყვის ციკლურ გადაადგილებას: $\{ x_0, x_1, x_2, x_3 \} \rightarrow \{ x_3, x_0, x_1, x_2 \}$.

დეშიფრაცია ხდება შებრუნებული გარდაქმნებით, დაშიფვრის ინვერტირებული ოპერაციები სრულდება უკუმიმდევრობით:

1. გასაღების გაფართოება - KeyExpansion;
2. გაშიფვრის 9 რაუნდიდან თითოეული რაუნდი შედგება შემდეგი გარდაქმნებისგან:

· AddRoundKey —მდგომარეობათა მატრიცისა და რაუნდის გასაღების შეკრება;

· InverseMixColumns —მატრიცაში სვეტების გადანაცვლება;

· InverseShiftRows —მდგომარეობათა მატრიცის სვეტების შებრუნებული ციკლური წაძვრა;

· SubBytes —state მდგომარეობათა მატრიცის ბაიტების ჩანაცვლება შებრუნებული inverseS- box ცხრილის მიხედვით;

3. ბოლო რაუნდი:

· AddRoundKey

· InverseShiftRows

· InverseSubBytes

მაგალითი:

მოცემული გვაქვს

გასაღები - byte key[n]={0x45, 0x3f, 0x11, 0x13,
0x22, 0xef, 0xd4, 0xe6,
0xab, 0xf7, 0x21, 0x44,
0x09, 0xed, 0x8f, 0x3c};

საწყისი ტექსტი -byte plain[n] = {0x31, 0x44, 0x34, 0xe2,
0x43, 0x5a, 0x31, 0x37,
0xab, 0x30, 0x98, 0x08,
0xa7, 0x4d, 0xa4, 0x34};

პროგრამული რეალიზაცია:

```
#include <iostream>
#include <bitset>
#include <string>
```

```
#define n 16
#define nr 10   //რაუნდების რაოდენობა
#define nk 4    //შესატანი სიტყვების რაოდენობა
using namespace std;
typedef bitset<8> byte;
typedef bitset<32> word;

byte S[16][16] = {
   {0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5,
0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76},
   {0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0,
0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0},
   {0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC,
0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15},
   {0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A,
0x07, 0x12, 0x80, 0xE2, 0xEB, 0x27, 0xB2, 0x75},
   {0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0,
0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84},
   {0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B,
0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF},
   {0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85,
0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C, 0x9F, 0xA8},
   {0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5,
0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2},
   {0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17,
0xC4, 0xA7, 0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73},
   {0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88,
0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB},
```

```
  {0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C,
0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79},
  {0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9,
0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A, 0xAE, 0x08},
  {0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6,
0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A},
  {0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E,
0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E},
  {0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94,
0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF},
  {0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68,
0x41, 0x99, 0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16}
};

byte Inv_S[16][16] = {
  {0x52, 0x09, 0x6A, 0xD5, 0x30, 0x36, 0xA5, 0x38,
0xBF, 0x40, 0xA3, 0x9E, 0x81, 0xF3, 0xD7, 0xFB},
  {0x7C, 0xE3, 0x39, 0x82, 0x9B, 0x2F, 0xFF, 0x87,
0x34, 0x8E, 0x43, 0x44, 0xC4, 0xDE, 0xE9, 0xCB},
  {0x54, 0x7B, 0x94, 0x32, 0xA6, 0xC2, 0x23, 0x3D,
0xEE, 0x4C, 0x95, 0x0B, 0x42, 0xFA, 0xC3, 0x4E},
  {0x08, 0x2E, 0xA1, 0x66, 0x28, 0xD9, 0x24, 0xB2,
0x76, 0x5B, 0xA2, 0x49, 0x6D, 0x8B, 0xD1, 0x25},
  {0x72, 0xF8, 0xF6, 0x64, 0x86, 0x68, 0x98, 0x16,
0xD4, 0xA4, 0x5C, 0xCC, 0x5D, 0x65, 0xB6, 0x92},
  {0x6C, 0x70, 0x48, 0x50, 0xFD, 0xED, 0xB9, 0xDA,
0x5E, 0x15, 0x46, 0x57, 0xA7, 0x8D, 0x9D, 0x84},
```

```
  {0x90, 0xD8, 0xAB, 0x00, 0x8C, 0xBC, 0xD3, 0x0A,
0xF7, 0xE4, 0x58, 0x05, 0xB8, 0xB3, 0x45, 0x06},
  {0xD0, 0x2C, 0x1E, 0x8F, 0xCA, 0x3F, 0x0F, 0x02,
0xC1, 0xAF, 0xBD, 0x03, 0x01, 0x13, 0x8A, 0x6B},
  {0x3A, 0x91, 0x11, 0x41, 0x4F, 0x67, 0xDC, 0xEA,
0x97, 0xF2, 0xCF, 0xCE, 0xF0, 0xB4, 0xE6, 0x73},
  {0x96, 0xAC, 0x74, 0x22, 0xE7, 0xAD, 0x35, 0x85,
0xE2, 0xF9, 0x37, 0xE8, 0x1C, 0x75, 0xDF, 0x6E},
  {0x47, 0xF1, 0x1A, 0x71, 0x1D, 0x29, 0xC5, 0x89,
0x6F, 0xB7, 0x62, 0x0E, 0xAA, 0x18, 0xBE, 0x1B},
  {0xFC, 0x56, 0x3E, 0x4B, 0xC6, 0xD2, 0x79, 0x20,
0x9A, 0xDB, 0xC0, 0xFE, 0x78, 0xCD, 0x5A, 0xF4},
  {0x1F, 0xDD, 0xA8, 0x33, 0x88, 0x07, 0xC7, 0x31,
0xB1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xEC, 0x5F},
  {0x60, 0x51, 0x7F, 0xA9, 0x19, 0xB5, 0x4A, 0x0D,
0x2D, 0xE5, 0x7A, 0x9F, 0x93, 0xC9, 0x9C, 0xEF},
  {0xA0, 0xE0, 0x3B, 0x4D, 0xAE, 0x2A, 0xF5, 0xB0,
0xC8, 0xEB, 0xBB, 0x3C, 0x83, 0x53, 0x99, 0x61},
  {0x17, 0x2B, 0x04, 0x7E, 0xBA, 0x77, 0xD6, 0x26,
0xE1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0C, 0x7D}
};
//გასაღების გაფართოებისთვის გამოყენებადი
//კონსტანტა
word Rcon[10] = {0x01000000, 0x02000000,
0x04000000, 0x08000000, 0x10000000,
            0x20000000, 0x40000000, 0x80000000,
0x1b000000, 0x36000000};
```

```
void SubBytes(byte m[4*4])
// ბაიტების შეცვლა S-box ცხრილის მიხედვით
{
   for(int i=0; i<16; ++i)
   {
      int row = m[i][7]*8 + m[i][6]*4 + m[i][5]*2 + m[i][4];
      int col = m[i][3]*8 + m[i][2]*4 + m[i][1]*2 + m[i][0];
      m[i] = S[row][col];
   }
}

void ShiftRows(byte m[4*4]) //მდგომარეობის
//მატრიცის სტრიქონების ციკლური ძვრა
{
   byte t = m[4];
   for(int i=0; i<3; ++i)
      m[i+4] = m[i+5];
   m[7] = t;

   for(int i=0; i<2; ++i)
   {
      t = m[i+8];
      m[i+8] = m[i+10];
      m[i+10] = t;
   }

   t = m[15];
```

```
    for(int i=3; i>0; --i)
        m[i+12] = m[i+11];
    m[12] = t;
}

byte gamravleba(byte a, byte b) {//გამრავლების
//ფუნქცია
    byte p = 0;
    byte hi_bit_set;
    for (int counter = 0; counter < 8; counter++) {
        if ((b & byte(1)) != 0) {
            p ^= a;
        }
        hi_bit_set = (byte) (a & byte(0x80));
        a <<= 1;
        if (hi_bit_set != 0) {
            a ^= 0x1b; /* x^8 + x^4 + x^3 + x + 1 */
        }
        b >>= 1;
    }
    return p;
}
void MixColumns(byte m[4*4])   //მდგომარეობის
//მატრიცის თითოეული სვეტი მრავლდება
{
    byte arr[4];
    for(int i=0; i<4; ++i)
    {
```

```
        for(int j=0; j<4; ++j)
            arr[j] = m[i+j*4];

        m[i] = gamravleba(0x02, arr[0]) ^ gamravleba(0x03,
arr[1]) ^ arr[2] ^ arr[3];
        m[i+4] = arr[0] ^ gamravleba(0x02, arr[1]) ^
gamravleba(0x03, arr[2]) ^ arr[3];
        m[i+8] = arr[0] ^ arr[1] ^ gamravleba(0x02, arr[2])
^ gamravleba(0x03, arr[3]);
        m[i+12] = gamravleba(0x03, arr[0]) ^ arr[1] ^ arr[2]
^ gamravleba(0x02, arr[3]);
    }
}

void AddRoundKey(byte m[4*4], word k[4])
// მდგომარეობათა მატრიცის ელემენტებისა და
//რაუნდის გასაღებების XOR-ით შეკრება
{
    for(int i=0; i<4; ++i)
    {
        word k1 = k[i] >> 24;
        word k2 = (k[i] << 8) >> 24;
        word k3 = (k[i] << 16) >> 24;
        word k4 = (k[i] << 24) >> 24;

        m[i] = m[i] ^ byte(k1.to_ulong());
        m[i+4] = m[i+4] ^ byte(k2.to_ulong());
        m[i+8] = m[i+8] ^ byte(k3.to_ulong());
```

```
        m[i+12] = m[i+12] ^ byte(k4.to_ulong());
    }
}

//აქედან იწყება გაშიფრვის პროცესი
//   S_box-ის შებრუნებული გარდაქმნა
 void InvSubBytes(byte m[4*4])
{
    for(int i=0; i<16; ++i)
    {
        int row = m[i][7]*8 + m[i][6]*4 + m[i][5]*2 + m[i][4];
        int col = m[i][3]*8 + m[i][2]*4 + m[i][1]*2 + m[i][0];
        m[i] = Inv_S[row][col];
    }
}

//მარჯვნივ ბაიტების ციკლური წაძვრა
void InvShiftRows(byte m[4*4])
{
    //მეორე სტრიქონის ციკლური გადაადგილება
//ერთი ბიტით მარჯვნივ
    byte t = m[7];
    for(int i=3; i>0; --i)
        m[i+4] = m[i+3];
    m[4] = t;
    //მესამე სტრიქონის ციკლური გადაადგილება
//ორი ბიტით მარჯვნივ
    for(int i=0; i<2; ++i)
```

```
    {
        t = m[i+8];
        m[i+8] = m[i+10];
        m[i+10] = t;
    }
//მეოთხე სტრიწონის ციკლური გადაადგილება 3
//ბიტით მარჯვნივ
    t = m[12];
    for(int i=0; i<3; ++i)
        m[i+12] = m[i+13];
    m[15] = t;
}
//მატრიცის სვეტების გადანაცვლება
void InvMixColumns(byte m[4*4])
{
    byte arr[4];
    for(int i=0; i<4; ++i)
    {
        for(int j=0; j<4; ++j)
            arr[j] = m[i+j*4];

        m[i] = gamravleba(0x0e, arr[0]) ^ gamravleba(0x0b,
arr[1]) ^ gamravleba(0x0d, arr[2]) ^ gamravleba(0x09,
arr[3]);
        m[i+4] = gamravleba(0x09, arr[0]) ^
gamravleba(0x0e, arr[1]) ^ gamravleba(0x0b, arr[2]) ^
gamravleba(0x0d, arr[3]);
```

```
        m[i+8] = gamravleba(0x0d, arr[0]) ^
gamravleba(0x09, arr[1]) ^ gamravleba(0x0e, arr[2]) ^
gamravleba(0x0b, arr[3]);
        m[i+12] = gamravleba(0x0b, arr[0]) ^
gamravleba(0x0d, arr[1]) ^ gamravleba(0x09, arr[2]) ^
gamravleba(0x0e, arr[3]);
    }
}

// გასაღების  გაფართოება
//4 ბაიტის ერთ სიტყვაში გაერთიანება

word Word(byte& k1, byte& k2, byte& k3, byte& k4)
{
   word shedegi(0x00000000);
   word t;
   t = k1.to_ulong();
   t <<= 24;
   shedegi |= t;
   t = k2.to_ulong();
   t <<= 16;
   shedegi |= t;
   t = k3.to_ulong();
   t <<= 8;
   shedegi |= t;
   t = k4.to_ulong();
   shedegi |= t;
   return shedegi;
```

```
}


word RotWord(word& rw)
{
   word high = rw << 8;
   word low = rw >> 24;
   return high | low;
}

/**
 *      S-box -ის გარდაქმნასიტყვასი შემავალი თითოეული ბაიტისთვის
 */
word SubWord(word& sw)
{
   word t;
   for(int i=0; i<32; i+=8)
   {
      int row = sw[i+7]*8 + sw[i+6]*4 + sw[i+5]*2 + sw[i+4];
      int col = sw[i+3]*8 + sw[i+2]*4 + sw[i+1]*2 + sw[i];
      byte val = S[row][col];
      for(int j=0; j<8; ++j)
         t[i+j] = val[j];
   }
   return t;
}
```

```
//გასაღების გაფართოების ფუნქცია
void KeyExpansion(byte key[4*nk], word w[4*(nr+1)])
{
   word t;
   int i = 0;
   while(i < nk)
   {
      w[i] = Word(key[4*i], key[4*i+1], key[4*i+2],
key[4*i+3]);
      ++i;
   }

   i = nk;

   while(i < 4*(nr+1))
   {
      //წინა სიტყვის ჩაწერა
      if(i % nk == 0)
         w[i] = w[i-nk] ^ SubWord(w[i-1]) ^ Rcon[i/nk-
1];
      else
         w[i] = w[i-nk] ^ t;
      ++i;
   }
}

//დაშიფრვის ფუნქცია
```

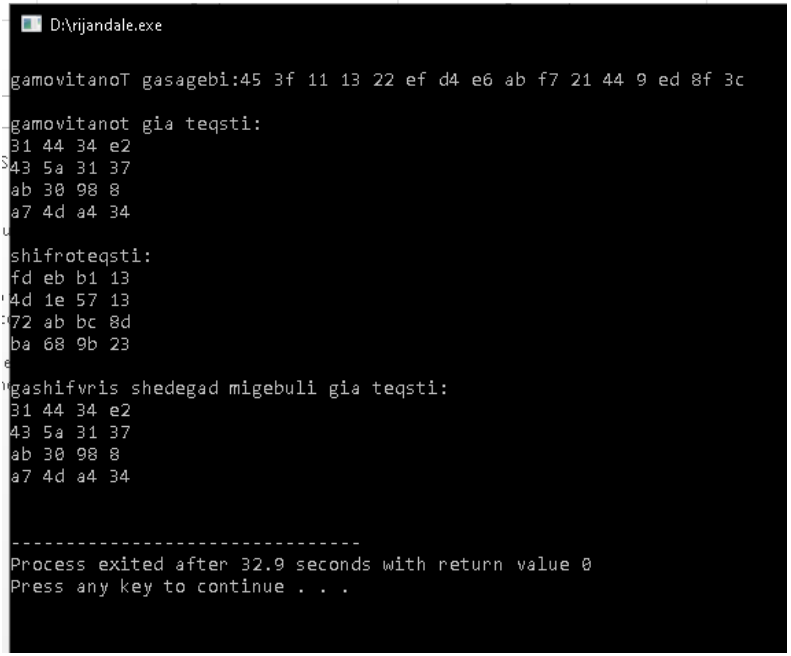
```
void encrypt(byte in[4*4], word w[4*(nr+1)])
{
   word key[4];
   for(int i=0; i<4; ++i)
      key[i] = w[i];
   AddRoundKey(in, key);

   for(int round=1; round<nr; ++round)
   {
      SubBytes(in);
      ShiftRows(in);
      MixColumns(in);
      for(int i=0; i<4; ++i)
         key[i] = w[4*round+i];
      AddRoundKey(in, key);
   }

   SubBytes(in);
   ShiftRows(in);
   for(int i=0; i<4; ++i)
      key[i] = w[4*nr+i];
   AddRoundKey(in, key);
}

//გაშიფრვის ფუნქცია
void decrypt(byte in[4*4], word w[4*(nr+1)])
{
   word key[4];
```

```
    for(int i=0; i<4; ++i)
        key[i] = w[4*nr+i];
    AddRoundKey(in, key);

    for(int round=nr-1; round>0; --round)
    {
        InvShiftRows(in);
        InvSubBytes(in);
        for(int i=0; i<4; ++i)
            key[i] = w[4*round+i];
        AddRoundKey(in, key);
        InvMixColumns(in);
    }

    InvShiftRows(in);
    InvSubBytes(in);
    for(int i=0; i<4; ++i)
        key[i] = w[i];
    AddRoundKey(in, key);
}


int main()
{


        byte key[n]={0x45, 0x3f, 0x11, 0x13,
             0x22, 0xef, 0xd4, 0xe6,
```

```
                0xab, 0xf7, 0x21, 0x44,
                0x09, 0xed, 0x8f, 0x3c};
                cout<<endl;


byte plain[n] = {0x31, 0x44, 0x34, 0xe2,
                0x43, 0x5a, 0x31, 0x37,
                0xab, 0x30, 0x98, 0x08,
                0xa7, 0x4d, 0xa4, 0x34};

  cout << "gamovitanoT gasagebi:";
  for(int i=0; i<n; ++i)
     cout << hex << key[i].to_ulong() << " ";
  cout << endl;

  word w[4*(nr+1)];
  KeyExpansion(key, w);


  cout << endl << "gamovitanot gia teqsti:"<<endl;
  for(int i=0; i<n; ++i)
  {
     cout << hex << plain[i].to_ulong() << " ";
     if((i+1)%4 == 0)
        cout << endl;
  }
  cout << endl;
```

```
    //დაშიფრვა
    encrypt(plain, w);
    cout << "shifroteqsti:"<<endl;
    for(int i=0; i<n; ++i)
    {
        cout << hex << plain[i].to_ulong() << " ";
        if((i+1)%4 == 0)
            cout << endl;
    }
    cout << endl;

    //გაშიფვრა
    decrypt(plain, w);
    cout << "gashifvris shedegad migebuli gia
teqsti:"<<endl;
    for(int i=0; i<n; ++i)
    {
        cout << hex << plain[i].to_ulong() << " ";
        if((i+1)%4 == 0)
            cout << endl;
    }
    cout << endl;
    return 0;
}
```

შედეგი:



კრიპტოანალიზი: AES არის სანდო და უსაფრთხო დაშიფვრის მეთოდი, რომელიც უკვე 20 წელიწადია რაც გამოიყენება და მაღალ დონეზე უზრუნველყოფს ინფორმაციის დაცვას.

AES-ის უპირატესობები:

- იყენებს გასაღების სიმეტრიულ ალგორითმებს, რაც ნიშნავს, რომ ერთიდაიგივე გასაღები გამოიყენება მონაცემთა დაშიფვრასა და

გაშიფვრაში. ეს ამცირებს გასაღებების ამოცნობის რისკს მესამე პირის მიერ გასაღებების გენერირების პროცესის შესახებ ცოდნისა და კონტროლის ნაკლებობის გამო.

- AES-ი გვთავაზობს 256 ბიტიან დაცვას, რომლის გატეხვაც დიდი სიმძლავრის მქონე გამოთვლითი საშუალებებითაც კი შეუძლებელია.
- უცნობია AES-ის სუსტი მხარეები,რომლებიც შეიძლება გამოიყენონ თავდამსხმელებმა.
- AES-ის 128 ბიტიანი ბლოკის ზომა უზრუნველყოფს ნაკლები მეხსიერების გამოყენებით მონაცემთა სწრაფ დამუშავებას და ასევე მაქსიმალურ უსაფრთხოებას.

დავალება: განახორციელეთ ტექსტის შიფრაცია - დეშიფრაცია AES-ის ალგორითმის გამოყენებით.

ინფორმაციის მთლიანობის პრობლემა, დაშიფრვა აუთენტიფიკაციით.

ჩვენს დროში ყველა სფეროში საინფორმაციო ტექნოლოგიების შეღწევამ გამოიწვია რიგი ცვლილებები ნებისმიერი ორგანიზაციისა და დაწესებულების დოკუმენტბრუნვის ტექნოლოგიებში, აგრეთვე ორგანიზაციასა და მომხმარებელს შორის დიალოგში. უკანასკნელ წლებში უფრო და უფრო იზრდება ელექტრონული დოკუმენტბრუნვის როლი, ხოლო ქაღალდზე წარმოდგენილ დოკუმენტაციასთან მუშაობის წილი ნელ-ნელა კლებულობს. 2020 წლიდან, პანდემიის პირობებში, მთელს მსოფლიოში განსაკუთრებით აქტუალურია სუბიექტებს შორის დოკუმენტაციის გაცვლა ელექტრონული სახით. ასეთი მიდგომის უპირატესობები ცხადია: დოკუმენტაციის შენახვის და დამუშავების ხარჯების შემცირება, დოკუმენტის სწრაფად ძიების შესაძლებლობა და ა.შ. დასამუშავებელი ინფორმაციის მოცულობის ექსპონენციალური ზრდის პირობებში ზემოთ აღნიშნული მიდგომა ერთადერთ გამოსავალს წარმოადგენს. თუმცა დოკუმენტბრუნვის ქაღალდიდან ელექტრონულ სახეზე გადაყვანა მომხმარებლების მიმართ მთელ რიგ პრობლემებს აჩენს, მათ შორის უმთავრესია, გადასაცემი დოკუმენტის მთლიანობის (ორიგინალობის) უზრუნველყოფისა და მისი ავტორის

აუთენტიფიკაციის პრობლემა. ელექტრონული შეტყობინების როგორც გამგზავნისათვის, ასევე მიმღებისათვის აუცილებელია შეტყობინების გაგზავნისას მისი უცვლელობის უზრუნველყოფა. დოკუმენტბრუნვისას შეტყობინება უნდა იყოს ტექნოლოგიებით დაცული გარეშე პირის მიერ ამ შეტყობინებაში რაიმე სახის ცვლილებების შეტანისაგან. იმ შემთხვევაში, თუ ბოროტმოქმედმა მაინც მოახერხა გადაცემულ შეტყობინებაში რაიმე ცვლილებების შეტანა, მიმღებს თანამედროვე ტექნოლოგიების მეშვეობით უნდა შეეძლოს თითქმის 100%-ს სიზუსტით ამ ფაქტის დადგენა. შეტყობინების ავტორის ნამდვილობის აუთენ-ტიფიკაციის პრობლემა მდგომარეობს იმაში, რომ სუბიექტმა ვერ შეძლოს თავისი საკუთარი შეტყობინების გარდა ვერც ერთი სხვა შეტყობინების ხელმოწერა. ხოლო თუ სუბიექტმა მაინც მოაწერა ხელი სხვის შეტყობინებას, მიმღებს ტექნოლოგიების გამოყენებით ასევე უნდა შეეძლოს ამ ფაქტის მაღალი სიზუსტით ამოცნობა. ჩვეულებრივ, ქაღალდზე დოკუმენტის წარდგე-ნისას ინფორმაცია და ავტორის ხელმოწერა ფიზიკური მატარებლით - ქაღალდითაა დაცული. ქაღალდის დოკუმენტის დაცვისთვის (ანუ ინფორმაციის მთლიანობის უზრუნველყო-ფისთვის) აგრეთვე გამოიყენება ფიზიკური ხელმოწერები, ბეჭდები, დამცავი (წყლის) ნიშნები ქაღალდზე, ჰოლოგრამები და ა.შ. ელექტრონულ

დოკუმენტში ინფორმაციისა და მისი ფიზიკური მატარებლის მყარი კავშირი არ არსებობს. ამიტომაც პრობლემის გადაწყვეტა სხვა საშუალებებით უნდა მოხდეს.

ტერმინი - ინფორმაციის მთლიანობა ინფორმატიკაში, კრიპტოგრაფიაში, ტელეკომუნიკაციების თეორიაში, საინფორმაციო უსაფრთხოების თეორიაში გამოიყენება და აღნიშნავს იმ ფაქტს, რომ მონაცემები არ იყო შეცვლილი ამ მონაცემებზე ჩატარებული ოპერაციების დროს. მონაცემებზე ჩატარებულ ოპერაციებში იგულისხმება მონაცემების გადაცემა, შენახვა ან გამოსახვა ინფორმაციის რაიმე მატარებელზე. ტელეკომუნიკაციებში მონაცემთა მთლიანობა ხშირად ამოწმებენ შეტყობინების ჰეშ-ჯამის გამოთვლით, ამ შემთხვევაში MAC ("message authentication code") ალგორითმით სარგებლობენ. ხოლო კრიპტოგრაფიასა და საინფორმაციო უსაფრთხოებაში ინფორმაციის მთლიანობაში გულისხმობენ, რომ ინფორმაციაში რაიმე სახის ცვლილების შეტანა მხოლოდ იმ სუბიექტებს შეუძლიათ, ვისაც აღნიშნული ცვლილებების შეტანის უფლება აქვთ. ინფორმაციის მთლიანობის დარღვევის მაგალითებია:

- ბოროტმოქმედის მიერ საბანკო ტრანზაქციის დროს აქაუნტის ნომრის ცვლილების

მცდელობა ან დოკუმენტის გაყალბების მცდელობა.

- ინფორმაციის უნებლიე (შემთხვევითი) ცვლილება გადაცემის დროს ან მყარი დისკის გაუმართავობის შემთხვევაში.
- მასობრივი ინფორმაციის წყაროების მიერ ფაქტების დამახინჯება საზოგადოებრივი აზრის მანიპულაციის მიზნით.

მონაცემთა ბაზების თეორიაში ინფორმაციის მთლიანობა მონაცემთა კორექტულობას და მათ არაწინააღმდეგობრიობას აღნიშნავს. ასე მაგალითად, მონაცემთა ბაზაში „ობოლი“ ჩანაწერები (ჩანაწერები, რომლებსაც გაწყვეტილი აქვთ მემკვიდრეობითი კავშირები სხვა ჩანაწერებთან) ან ჩანაწერების გასაღებების (“primary kay”) უნიკალურობის თვისების დაკარგვა ინფორმაციის მთლიანობის დარღვევას წარმოადგენს.

კრიპტოგრაფიაში მონაცემთა მთლიანობის შემოწმების მიზნით გამოიყენება ჰეშ-ფუნქციები, მაგალითად MD5. ჰეშ-ფუნქცია გარდაქმნის ნებისმიერი ზომის ბაიტების მიმდევრობას ფიქსირებული ზომის ბაიტების მიმდევრობაში (ანუ რიცხვად). თუ მონაცემები იცვლება, მაშინ ჰეშ-ფუნქციის მიერ გენერირებული რიცხვიც იცვლება. მონაცემთა მთლიანობაში ამ მონა-

ცემების წინასწარ ცნობილი სახისა და ხარისხის შენარჩუნება იგულისხმება.

ფართოდ გამოიყენება აგრეთვე ტერმინი - ობიექტის მთლიანობა („integrity”), რომელიც ინფორმაციული უსაფრთხოების თეორიაში გვხვდება. ამ შემთხვევაში ობიექტს უწოდებენ სპეციალიზებულ მონაცემებს ან ავტომატიზებული სისტემის რესურსებს. ობიექტს გააჩნია რიგი მახასიათებლები: ხელმისაწვდომობა („availability“), მთლიანობა („integrity“), კონფიდენციალობა („confidentiality“); აგრეთვე, შეიძლება გამოიყოს სხვა თვისებებიც: არაუარყოფა („non-repudiation“), ანაგარიშვალდებულება („accountability“), ავთენტურობა („authenticity“), საიმედოობა („reliability“). აღნიშნული ტერმინოლოგია ძალიან აქტუალურია და ხშირად გამოიყენება. თანამედროვე მოთხოვნების შესაბამისად, ინფორმაციის ცალკეული ნაწილების დაზიანების ან დაკარგვის შემთხვევაში მონაცემები მაინც უნდა ინარჩუნებდნენ მთლიანობის თვისებას და ექვემდებარებოდნენ უსაფრთხოდ აღდგენას სარეზერვო კოპირების და არქივირების მეთოდებით. ინფორმაციის მთლიანობის დაცვას ემსახურება კრიპტოგრაფიული მეთოდები: დაშიფრვა, ჰეშირება და ელექტრონული ციფრული ხელმოწერა.

როგორც ადრე შევამჩნიეთ, მონაცემთა დაშიფრვა, ცხადია, ვერ უზრუნველყოფს მონაცემთა მთლიანობის დაცვას. ინფორმაციის მთლიანობის დაცვას ემსახურებას ზოგიერთი კრიპტოგრაფიული მეთოდი. მონაცემთა მთლიანობის დარღვევისას შეიძლება მოხდეს ბიტების ინვერსია, ახალი ბიტების (და შესაბამისად ახალი მონაცემების) დამატება მესამე პირის მიერ, ბიტების თანმიმდევრობის ამოშლა, ბიტების ან ბაიტების თანმიმდევრობის შეცვლა. შეცდომების ამოცნობის და შესწორების მეთოდით შეიძლება ინფორმაციის შემთხვევითი დაზიანებების აღმოფხვრა. ხოლო კრიპტოანალიტიკოსის მიერ ინფორმაციის შეცვლის აღმოჩენის მიზნით ინფორმაციაში ჩასმულია ე.წ. ჭარბი ინფორმაცია, ანუ შეტყობინებაში ჩასმულია შესამოწმებელი ფრაგმენტი. ეს ფრაგმენტი გამოითვლება სპეციალური ალგორითმით და წარმოადგენს ინფორმაციის მთლიანობის ინდიკატორს. ამ მომენტში დგინდება, იყო თუ არა მესამე პირის მიერ მონაცემები შეცვლილი. შეტყობინებაში ჩასმული **იმიტოჩანართი MAC** („message authentication code“— შეტყობინების ავთენტურობის კოდი) თავსდება შეტყობინებაში დაშიფრვის მომენტში ან უფრო ადრე. **MAC** ჩანართი წარმოადგენს იმიტოდაცვის ინსტრუმენტს, შეტყობინებების აუთენტიფიკაციის პროტოკოლებში, ის წარმოადგენს

შეტყობინებაში ჩასმულ სიმბოლოების ერთობლიობას და ემსახურება ინფორმაციის მთლიანობის უზრუნველყოფის ამოცანას და მონაცემთა წყაროს აუთნტიფიკაციას, **MAC** დაცვა უზრუნველყოფს ინფორმაციის ფალსიფიკაციის დაცვას. თუ გაგზავნილ შეტყობინებაში ჰეშ-ფუნქციის მნიშვნელობა არის დამატებული, მაშინ შეტყობინების მიმღები აგრეთვე ითვლის მიღებული შეტყობინების ჰეშ-ფუნქციას. თუ გამგზავნისა და მიმღების მიერ გამოთვლილი ჰეშ ფუნქციის მნიშვნელობები ერთმანეთს დაემთხვა, მაშინ ითვლება, რომ შეტყობინება ცვლილებების გარეშე იქნა მიღებული. აქედან ვასკვნით, რომ ჰეშ-ფუნქცია უზრუნველყოფს შეტყობინების მთლიანობის საკითხს, მაგრამ არ წყვეტს აუთენტობის საკითხს.

როგორც ზემოთ აღვნიშნეთ, ფალსი-ფიკაციის (ანუ იმიტაციის) საკითხს მხოლოდ **იმიტოჩანართი** წყვეტს. **MAC** დაცვისთვის გამოიყენება საიდუმლო გასაღები, რომელიც მხოლოდ გამგზავნი და მიმღები პირებისთვისაა ცნობილი. იმიტოჩანართში ორობითი თანრიგების რაოდენობა განსაზღვრულია კრიპტოგრაფიულად, ფალსიფიცირებული მონაცემების მიღების ალბათობა $1/2^p$ რიცხვის ტოლია, სადაც p - იმიტოჩანართის ორობითი თანრიგების რიცხვია. იმიტოჩანართი წარმოადგენს რაიმე x შეტყობინების ფუნქციას $x = f(x)$. როგორც ზემოთ

აღვნიშნეთ, ის ერთდროულად ემსახურება როგორც შეტყობინების აუთენტიფიკაციის საკითხს, ასევე ამ შეტყობინების მთლიანობის საკითხსაც. ამიტომ იმიტოჩანართებს ორ კლასად ყოფენ:

- შეტყობინების მთლიანობის კოდი (MDC,“modification detection code“), რომელიც გამოითვლება შეტყობინების ჰეშირების გზით. ჰეშირების დროს ნებისმიერი სიგრძის მონაცემთა მასივი გარდაიქმნება ფიქსირებული სიგრძის ბიტურ სტრიქონად. ჰეშირების განმახორციელებელ ჰეშ-ფუნქციებს ხანდახან ნახევევის ფუნქციებსაც ეძახიან. ჰეშირების შედეგად მიღებულ სტრიქონს კი - ჰეშ-კოდს ან შეტყობინების მესიჯს (”message digest“) უწოდებენ. თუ ორ შეტყობინებას ერთნაირი ჰეშ-კოდები აქვთ, მაშინ შეტყობინებები ერთმანეთს ემთხვევა, ხოლო თუ ჰეშ-კოდები სხვადასხვაა, შეტყობინებებიც სხვადასხვა იქნება. ჰეშირებას იყენებენ ასოციური მასივების ასაგებად, მონაცემებში დუბლირების ძებნის დროს, უნიკალური იდენტიფიკატორების აგების დროს, უნებლიე და სხვა სახის შეცდომების აღმოჩენის მიზნით საკონტროლო ჯამის მეთოდის გამოყენებისას, პაროლების შენახვის სისტემის დაცვისთვის, ელექტრონული ხელმოწერების გამოყენების დროს და სხვა. ცხადია, რომ ჰეშ-ფუნქციების მნიშვნელობების რაოდენობა ბევრად უფრო მცირეა, ვიდრე სხვადასხვა სახის საწყისი

მონაცემების რაოდენობა. ამიტომ შეიძლება ისეთი განსხვავებულ მონაცემთა ორი მასივის შერჩევა, რომ მათი ჰეშ-კოდები ერთმანეთს ემთხვევა. მათ **კოლიზიებს** უწოდებენ. ამიტომ ჰეშ-ფუნქციის შეფასებისას კოლიზიების არსებობის ალბათობა მცირე უნდა იყოს. ჰეშ-ფუნქციების უმარტივესი მაგალითია საკონტროლო ჯამის გამოთვლა.

- შეტყობინების აუთენტიფიკაციის კოდი MAC ("message authentication code"), იცავს შეტყობინებას ფალსიფიკაციისაგან, გამოითვლება შეტყობინების ჰეშირებით, საიდუმლო გასაღების გამოყენებით.

თავი III. ასიმეტრიული (ღია გასაღებიანი) კრიპტოსისტემები

1976 წელს უიტფილდ დიფმა და მარტინ ჰელმანმა და მათგან დამოუკიდებლად რალფ მერკლმა წამოაყენეს ღია გასაღებიანი ანუ ასიმეტრიული კრიპტოგრაფიის იდეა, რომელმაც გამოიწვია რევოლუციური ცვლილებები კრიპტოგრაფიაში. იდეის არსი მდგომარეობს იმაში, რომ შესაძლებელია შეტყობინების დასაშიფრად გამოვიყენოთ ერთი გასაღები, ხოლო დეშიფრაციისათვის კი – მეორე, პირველისაგან განსხვავებული გასაღები. ასეთმა სისტემებმა რადიკალურად გააფართოვა კრიპტოგრაფიის შესაძლებლობები.

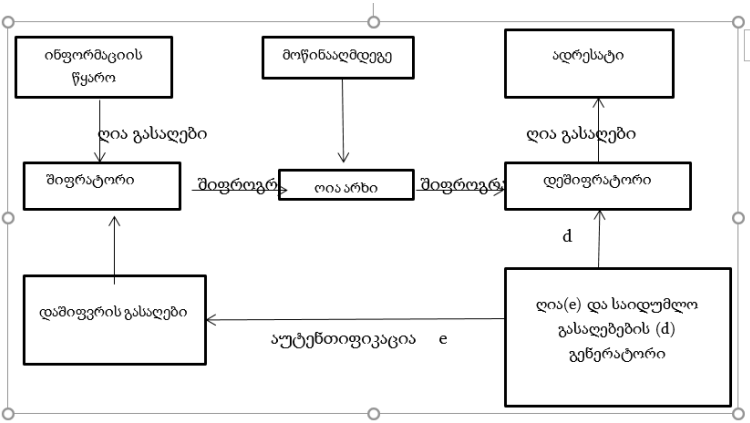
ასეთი კრიპტოგრაფიული სისტემის ასაგებად, აუცილებელია, რომ ეს გასაღებები დაკავშირებულები იყვნენ ერთმანეთთან, რადგან წინააღმდეგ შემთხვევაში შეუძლებელი იქნება დაშიფრული ტექსტის დეშიფრაცია. ამასთან, ეს კავშირი აუცილებლად უნდა იყოს ცალსახა (კვლავაც გამომდინარე დეშიფრაციის მოთხოვნებიდან), და ისე დამალული, რომ

მოწინააღმდეგემ, რომლისთვისაც ყოველთვის ცნობილია ერთ-ერთი გასაღები, ვერ შეძლოს მისი საშუალებით მეორის გამოთვლა. ასეთი თვისებების მქონე ნებისმიერ ფუნქციას უწოდებენ ცალმხრივმიმართულ ფუნქციას.

ღია გასაღებიანი დაშიფრვის ალგორითმების იდეა ეფუძნება ცალმხრივმიმართული ხაფანგიანი ფუნქციის ცნებას. იდეა კი მდგომარეობს იმაში, რომ რომელიმე ასეთი ფუნქციის საშუალებით მყარდება კავშირი ორ სიდიდეს შორის $y = f(x)$, სადაც x წარმოადგენს საიდუმლო გასაღებს, y - კი ღიას. f ცალმხრივმიმართული ხაფანგიანი ფუნქცია უზრუნველყოფს იმას, რომ მოწინააღმდეგეს არ შეუძლია ღია გასაღებიდან საიდუმლო გასაღების გამოთვლა. ასეთ პრინციპზე აგებული კრიპტოალგორითმები შეიძლება გამოვიყენოთ როგორც ინფორმაციის კონფიდენციალურობის, ასევე ინფორმაციის მთლიანობის და გასაღებების განაწილების ამოცანებში. ღია გასაღებიანი კრიპტოსისტემით ინფორმაციის გადაცემის სქემას აქვს ნახ. 14–ში მოცემული სახე. როგორც ამ სქემიდან ჩანს, აღარ არის საჭირო დახურული არხი გასაღების

გადასაცემად. საჭიროა მხოლოდ გასაღებების აუთენტიფიკაცია და სერთიფიცირება. ის ვინც შიფრავს ინფორმაციას ღია გასაღებით, დარწმუნებული უნდა იყოს, რომ ეს გასაღები ნამდვილად ეკუთვნის ადრესატს. თუ სქემაში მონაწილე სუბიექტებმა უნდა შეცვალონ როლები, მაშინ ასეთ კრიპტოსისტემაში მონაწილე ყველა სუბიექტს უნდა ჰქონდეს ორი გასაღები, ღია და საიდუმლო, ერთი ინფორმაციის დასაშიფრად და მეორე ინფორმაციის დეშიფრაციისათვის. როდესაც საქმე ეხება კონფიდენციალურობის დაცვას, როგორც წესი ღია გასაღები გამოიყენება ინფორმაციის დასაშიფრად, საიდუმლო კი – ინფორმაციის დეშიფრაციისათვის. სისტემის ნებისმიერ მომხმარებელს შეუძლია დაშიფროს ინფორმაცია, მაგრამ დეშიფრაცია შეუძლია მხოლოდ საიდუმლო გასაღების პატრონს, რადგანაც მხოლოდ მან იცის ის ინფორმაცია (საიდუმლო პარამეტრი), რომელიც საშუალებას იძლევა საიდუმლო გასაღებიდან გამოთვალოს ღია გასაღები. ორი გასაღების გამოყენება იწვევს იმას, რომ ხშირად დაშიფრვისა და დეშიფრაციის ალგორითმები სხვადასხვაა, ამიტომ ღია გასაღებიან სისტემებს მეორენაირად ხშირად უწოდებენ

აგრეთვე ასიმეტრიულ კრიპტოგრაფიულ ალგორითმებსაც.



ნახაზი- 14 ინფორმაციის გადაცემის სქემა ორი გასაღების საშუალებით

ღია გასაღებიანი კრიპტოსისტემა აღიწერება $(M, C, K_p, K_s, E_p, D_s)$ პარამეტრებით, რომლებიც აკმაყოფილებენ შემდეგ პირობებს:

1. M არის ღია ტექსტების სასრული სიმრავლე;

2. C არის შიფროტექსტების სასრული სიმრავლე;

3. K_p არის ღია გასაღებების სივრცე, ღია გასაღებების სასრული სიმრავლე;

4. K_s არის საიდუმლო გასაღებების სივრცე, საიდუმლო გასაღებების სიმრავლე, ამასთან

$K_p = f(K_s)$, სადაც f არის ცალმხრივმიმართული ფუნქცია საიდუმლო პარამეტრით;

5. E_p არის დაშიფრვის ალგორითმების სიმრავლე;

6. D_s არის დეშიფრაციის ალგორითმების სიმრავლე;

7. ნებისმიერი $m \in M$ - თვის არსებობს ისეთი $e_{k_p} \in E_p$ და $d_{k_s} \in D_s$, რომ სრულდება ტოლობა $d_{k_s}(e_{k_p}(m)) = m$.

ღიაგასაღებიანი კრიპტოსისტემა- რივესტი - ადელმანი -შამირი (RSA)

ერთ-ერთი ყველაზე პოპულარული ღიაგასაღებიანი კრიპტოსისტემა RSA შეიქმა 1977 წელს რ. რივესტის, ა. შამირისა და ლ. ადელმანის მიერ. RSA არის პირველი სრულფასოვანი ალგორითმი ღია გასაღებით, რომელიც გამოიყენება როგორც შიფრაციისთვის, ასევე ციფრული ხელმოწერებისთვის. ამ ალგორითმის გაგება და რეალიზაცია დიდ სირთულეს არ წარმოადგენს. მათემატიკის პოპულარიზატორმა მარტინ გარდნერმა ის „მათემატიკური თამაშის“ სახელითაც კი გამოაქვეყნა. მრავალი წლების

მანძილძე კრიპტოანალიტიკოსები სწავლობენ ამ ალგორითმის მედეგობას და მისი საიმედოობა ვერც დადასტურებული და ვერ უარყოფილია. RSA ალგორითმის უსაფრთხოება ეფუძნება დიდი რიცხვების თანამამრავლებად წარმოდგენის სირთულეს. ღია და დახურული გასაღებები წარმოადგენენ ორი დიდი (100-200 ან ზოგჯერ მეტ თანრიგიანი) მარტივი რიცხვის ფუნქციებს. იგულისხმება, რომ შიფროტექსტისა და ღია გასაღების მეშვეობით ღია ტექსტის მიღება ორი დიდი რიცხვის ნამრავლის მამრავლებად დაშლის ამოცანის ექვივალენტურია. ორი გასაღების გენერაციის მიზნით გამოიყენება ორი დიდი შემთხვევითი მარტივი p და q რიცხვი. მაქსიმალური უსაფრთხოებისათვის ეს რიცხვები ერთნაირი სიგრძის უნდა იყოს (თანრიგთა რაოდენობა ერთნაირია). უნდა გამოვთვალოთ $n = p \cdot q$ ნამრავლი. ამის შემდეგ შემთხვევითად უნდა შეირჩეს დაშიფრვის e გასაღები, ისეთი რომ e და $(p-1)(q-1)$ ურთიერთ მარტივი რიცხვები იყოს. დეშიფრაციის გასაღებს წარმოადგენს d რიცხვი, რომლის პოვნისთვის გამოიყენება ევკლიდეს განზოგადებული ალგორითმი. d რიცხვისთვის სრულდება პირობა:

$$ed = 1\ mod\ ((p-1)(q-1))$$

სხვანაირად რომ გადავწეროთ,

$$d = e^{-1}\ mod\ ((p-1)(q-1)).$$

შევნიშნოთ, რომ d და n აგრეთვე ურთიერთმარტივი რიცხვებია. e და n ღია გასაღებებია, ხოლო d რიცხვი - დახურული გასაღებია. p და q რიცხვები აღარ გვჭირდება, მათ აღარ გამოვიყენებთ, თუმცა ვერ გამოვაცხადებთ.

რაიმე m ციფრული შეტყობინების დასაშიფრად ის თავდაპირველად უნდა დაიყოს რაიმე ციფრული ბლოკების სახით, რომელთა სიგრძე არ აღემატება n-ს (ორობითი სახით წარმოდგენილი მონაცემებისათვის უნდა შევარჩიოთ 2-ის ყველაზე მაღალი ხარისხი, რომელიც n-ზე ნაკლებია). ე.ი. თუ მარტივი p და q რიცხვები 100 თანრიგისგან შედგება, მაშინ n რიცხვი შედგება 200 თანრიგისაგან. აქედან გამომდინარე გადასაცემი შეტყობინების ყოველი m_i ბლოკი უნდა იყოს 200 თანრიგიანი. თუ ფიქსირებული რაოდენობის ბლოკების შიფრაცია, ისინი შეიძლება შევავსოთ მარცხნიდან საჭირო რაოდენობის ნულებით, იმ მოსაზრებებით, რომ ყოველი ბლოკი n-ზე ნაკლები სიგრძის იქნება. c დაშიფრული შეტყობინება შედგება იმავე სიგრძის c_i ბლოკებისაგან.

შიფრაციის ფორმულა შემდეგნაირად გამოიყურება:

$$c_i = m_i^e \ mod \ n$$

ხოლო შეტყობინების დეშიფრაციისათვის ავიღოთ ყოველი c_i ბლოკი და გამოვთვალოთ

$$m_i = c_i^d \ mod \ n$$

რადგანაც

$$c_i^d = (m_i^e)^d = m_i^{ed} = m_i^{k(p-1)(q-1)+1}$$
$$= m_i m_i^{k(p-1)(q-1)} = m_i * 1 = m_i$$

ყველა გამოთვლა $mod\ n$-ით ხდება (აქ აღწერილია პირდაპირი მეთოდი, პრაქტიკაში გამოიყენება სპეციალური სქემა, რომელიც ტექსტს გარდაქმნის რიცხვად - OAEP სქემა).

ანუ სქემატურად ჩამოვაყალიბოთ ალგორითმი:

ღია გასაღები:

n - ორი მარტივი p და q რიცხვის ნამრავლი, ეს რიცხვები საიდუმლოა.

e არის ურთიერთმარტივი რიცხვი $(p-1)(q-1)$ რიცხვთან.

დახურული გასაღები: $d = e^{-1} \ mod \ ((p-1)(q-1))$.

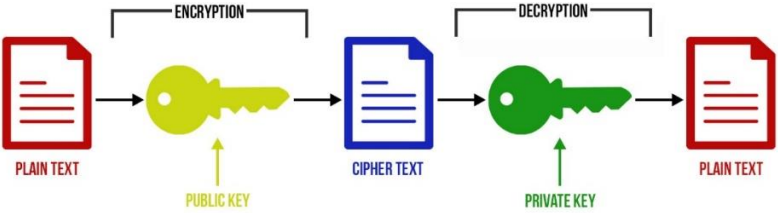
შიფრაცია $c = m^e \ mod \ n$

დეშიფრაცია $m = c^d \ mod \ n$

შენიშვნა: ზუსტად ასევე შესაძლებელია შიფრაცია d რიცხვის მეშვეობით, ხოლო დეშიფრაცია - e რიცხვის მეშვეობით, ანუ ჩვენ არჩევნში თავისუფალი ვართ. ალგორითმის მუშაობა რიცხვთა თეორიაზეა დამყარებული.

არსებობს RSA ალგორითმის უამრავი აპარატული რეალიზაცია მიკროსქემის სახით, თუმცა მათი თავისუფლად შეძენა შეუძლებელია. აპარატულად RSA ალგორითმი დაახლოებით 100-1000-ჯერ (დამოკიდებულია შიფრაციის ბიტების რაოდენობაზე) უფრო ნელია, ვიდრე DES ალგორითმის მიკროსქემები. ეს კი კიდევ ერთხელ ადასტურებს, რომ RSA ვერასოდეს ვერ იმუშავებს სიმეტრიული ალგორითმების სიჩქარით.

ალგორითმის შექმნის პარიოდში მარტივი რიცხვების ზომა იყო 500 ბიტი, დღეს კი უკვე აღარ ყოფნის 2048 ბიტიანი რიცხვები და გადავიდნენ 3072 ბიტიან რიცხვებზე, ამიტომ პრაქტიკულად, ამ ალგორითმს შეტყობინების დასაშიფრად აღარ იყენებენ.



ნახაზი- 15 კრიპტოსისტემა RSA

სავარჯიშო N1

I გასაღების შექმნა

RSA სისტემისათვის დავუშვათ, რომ $p = 3;\ q = 7;\ M1 = 11;\ M2 = 5;\ M3 = 25,$ სიდიდეთა მნიშვნელობები. განვახორციელეთ შიფრაციისა და დეშიფრაციის ოპერაციები, მაშინ

$$N = pq;\quad N = 3 * 7 = 21;$$
$$\varphi(N) = (p-1)(q-1);\ \ \varphi(N) = (3-1)(7-1) = 12;$$

$$ed = 1\big(mod\varphi(N)\big);\ \ ed = 1(mod12);\ \ e = 5; d = 5;$$

ანუ

მივიღეთ: ღია გასაღები (n,e) და დახურული გასაღები (n,d) ე.ი. ღია გასაღები (21,5) და დახურული გასაღები (21,5)

II ტექსტის დაშიფრვა

X *მხარე გამოთვლის* $C = M^e \equiv M^e(modn);$

$C1 \equiv M1^e(modn) \equiv 11^5(mod21) \equiv 2(mod21);$

$C2 \equiv M2^e(modn) \equiv 5^5(mod21) \equiv 17(mod21);$

$C3 \equiv M3^e(modn) \equiv 25^5(mod21) \equiv 16(mod21);$

III ტექსტის გაშიფრვა (დეშიფრაცია)

Y *მხარე გამოთვლის* $M = C^e \equiv C^e(modn);$

$M1 \equiv C1^d(modn) \equiv 2^5(mod21) \equiv 11(mod21);$

$M2 \equiv C2^d(modn) \equiv 17^5(mod21) \equiv 5(mod21);$

$M3 \equiv C3^d(modn) \equiv$

$16^5(mod21) \equiv 4(mod21); \rightarrow$

ტექსტის დეშიფრაციის დროს მივიღეთ მცდარი მნიშვნელობა, რადგან M<n პირობა ვერ სრულდება.

სავარჯიშო N2

I გასაღების შექმნა

RSA სისტემისათვის დავუშვათ, რომ

$p = 3;\ q = 11;\ M1 = 3;\ M2 = 18;\ M3 = 15,$

სიდიდეთა მნიშვნელობები. განვახორციელეთ შიფრაციისა და დეშიფრაციის ოპერაციები, მაშინ

$N = pq;\quad N = 3 * 11 = 33;$

$\varphi(N) = (p-1)(q-1);\ \varphi(N) = (3-1)(11-1) = 20;$

$$ed = 1\big(mod\varphi(N)\big);\ \ ed = 1(mod20);\ \ e = 3;\ d = 7;$$

ანუ

მივიღეთ: ღია გასაღები (n,e) და დახურული გასაღები (n,d) ე.ი. ღია გასაღები (33, 3) და დახურული გასაღები (33, 7)

II ტექსტის დაშიფრვა

X მხარე გამოთვლის $C = M^e \equiv M^e(modn);$

$$C1 \equiv\ M1^e(modn) \equiv 3^3(mod33) \equiv 27(mod33);$$

$$C2 \equiv M2^e(modn)\ \equiv 18^3(mod33) \equiv 24(mod33);$$

$$C3 \equiv M3^e(modn)\ \equiv 15^3(mod33) \equiv 9(mod33);$$

Y მხარე გამოთვლის $M = C^e \equiv C^e(modn);$

$$M1 \equiv\ C1^d(modn) \equiv 27^7(mod33) \equiv 3(mod33);$$

$$M2 \equiv\ C2^d(modn) \equiv 24^7(mod33) \equiv 18(mod33);$$

$$M3 \equiv\ C3^d(modn) \equiv$$

$$9^7(mod33) \equiv 15(mod33);$$

ალგორითმის პროგრამული რეალიზაცია:

```
#include<iostream>
#include<stdlib.h>
#include<math.h>
```

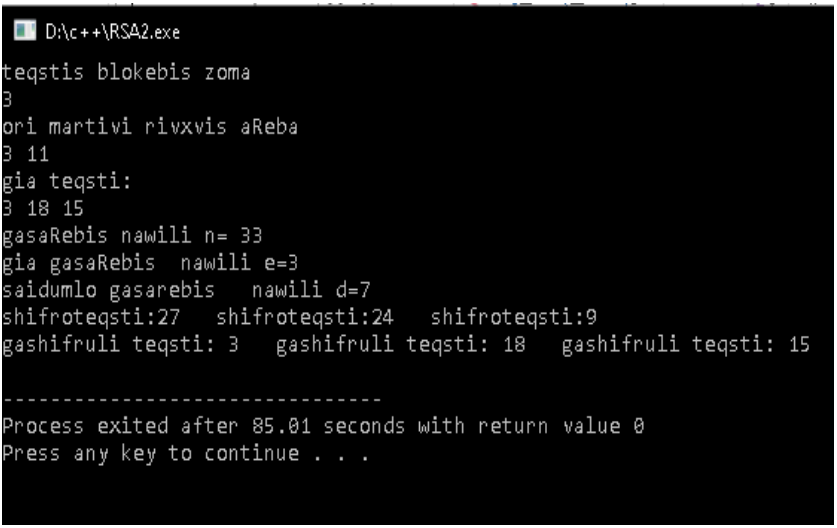
```
#include<time.h>
using namespace std;
 long long int  usg(long long int x, long long int y)
//უდიდესი საერთო გამყოფის პოვნა
        {while(x!=y)
        if(x>y) x-=y;
        else y-=x;
        return x;
                        }
         long long e_urtiertmartivi (long long int f1)
         {
// ღია გასაღების  e-სიდიდის  არჩევა
         long long int e;
                          bool f=true;
                        for(int i=2;i<f1 && f;i++ )
                                if(usg(i,f1)==1){
                                  f=false; e=i;}
                         return e;
                         }
        int d_mod(long long int a, long long int b)
{
//ევკლიდეს გაფართოებული ალგორითმით
//საიდუმლო გასაღების d სიდიდის პოვნა
//ed=1(mod f)
```

```
 long long int q, r, x1, x2, y1, y2,x,y,d;
 if (b == 0) {
   d=a; x=1; y = 0;
   return x;
 }
 x2=1; x1=0; y2=0; y1=1;
 while (b > 0) {
   q=a/b; r=a-q*b;
   x=x2-q*x1; y=y2-q*y1;
   a=b; b=r;
   x2=x1; x1=x; y2=y1; y1=y;
 }
 d = a; x = x2; y = y2;
if (d == 1) return x;}
long long int gamotvla(long long int  a1, long long int
x1, long long int p1)
            {
//დაშიფვრისა და გაშიფრის ფუნქცია
                long long int t;
                  if(x1==1)
                       return a1;
                  t=gamotvla(a1,x1/2,p1);
                  if(x1%2==0)
                       return (t*t)%p1;
```

```
                else
                    return (((t*t)%p1)*a1)%p1;}
int main()
{ int m;
cout<<"teqstis blokebis zoma"<<endl;
cin>>m;
      long long int p,q,x,n,d,e,text[m],C[m],D[m];
      cout<<"ori martivi rivxvis aReba "<<endl;
      cin>>p>>q;
      cout<<"gia teqsti:"<<endl;
     for(int i=0;i<m;i++)
       cin>>text[i];
            n=p*q;
      cout<<"gasaRebis nawili n= "<<n<<endl;
       long long int f=(p-1)*(q-1);
      e=e_urtiertmartivi(f);
      cout<<"gia gasaRebis  nawili e="<<e<<endl;
      d=d_mod(e,f);
      cout<<"saidumlo gasarebis   nawili d="<<d<<endl;
      for(int i=0;i<m;i++)
      C[i]=gamotvla(text[i],e,n);
      for(int i=0;i<m;i++)
    cout<<"shifroteqsti:"<<C[i]<<"    ";
    cout<<endl;
```

```
 for(int i=0;i<m;i++)
  D[i]=gamotvla(C[i],d,n);
 for(int i=0;i<m;i++)
 cout<<"gashifruli teqsti: "<<D[i]<<"   ";
 cout<<endl;
}
```

შედეგი:



სავარჯიშო N3

I გასაღების შექმნა

RSA სისტემისათვის დავუშვათ, რომ

$p = 7;\ q = 13;\ M = 3;$

სიდიდეთა მნიშვნელობები. განახორციელეთ შიფრაციისა და დეშიფრაციის ოპერაციები, მაშინ

$$N = pq; \quad N = 13 * 7 = 91;$$

$$\varphi(N) = (p-1)(q-1); \quad \varphi(N) = (13-1)(7-1) = 72;$$

$$ed = 1(mod\varphi(N)); \quad ed = 1(mod72); \quad e = 29; \quad d = 5;$$

ანუ

მივიღეთ: ღია გასაღები (n,e) და დახურული გასაღები (n,d) ე.ი. ღია გასაღები (91, 29) და დახურული გასაღები (91, 5)

II ტექსტის დაშიფვრა

$$X \text{ მხარე გამოთვლის } C = M^e \equiv M^e(modn);$$

$$C \equiv M^e(modn) \equiv 3^{29}(mod91) \equiv 61(mod91);$$

III ტექსტის გაშიფვრა (დეშიფრაცია)

$$Y \text{ მხარე გამოთვლის } M = C^e \equiv C^e(modn);$$

$$M \equiv C^d(modn) \equiv 61^5(mod91) \equiv 3(mod91);$$

კრიპტოანალიზი: RSA ალგორითმის უსაფრთხოება მთლიანად დამოკიდებულია დიდი რიცხვების თანამამრავლებად დაყოფის საკითხზე. თუმცა მკაცრად არასოდეს დამტკიცებულა ფაქტი,

რომ n რიცხვი უნდა წარმოვადგინოთ თანამარავლების სახით იმისათვის, რომ m რიცხვი აღვადგინოთ c და e რიცხვებით. კრიპტოანალიზისთვის სხვა გზებით მოქმედებენ. ცხადია, თუ რაიმე მეთოდით კრიპ-ტოანალიტიკოსი იპოვის d რიცხვს, იგივე მეთოდით შესაძლებელია დიდი რიცხვების თანამამრავლებად წარმოდგენაც. RSA ალგორითმის გატეხვა, ცხადია, შესაძლებელია $(p-1)(q-1)$ რიცხვის შერჩევითაც, მაგრამ ესეც სამაოდ რთული და შრომატევადია.

დავალება: RSA სისტემისათვის დავუშვათ, რომ

$$p = 3;\ q = 11;\ M1 = 19;\ M2 = 20;\ M3 = 12;$$

სიდიდეთა მნიშვნელობები. განახორციელეთ შიფრაციისა და დეშიფრაციის ოპერაციები.

დიფი-ჰელმანის ალგორითმი

დიფი ჰელმანის ალგორითმით, რომელიც შეიქმნა გასაღებების განაწილების პრობლემის გადასაჭრელად, დაიწყო სწორედ ღიაგასაღებიანი კრიპტოგრაფია. 1976 წელს გამოქვეყნდა ეს ალგორითმი აუიტლდ დიფისა და მარტინ ჰელმანის სახელით, მაგრამ თვით ჰელმანი, აღნიშნავდა რამერკლის ღვაწლს , მოითხოვდა რომ ალგორითმისთვის დაერქვათ სამივეს სახელით, თუმცა ალგორითმის სახელი არ შეცვლილა. დიფი-ჰელმანის ალგორითმში ხდება არა წინასწარ გამზადებული გასაღების გადაცემა, არამედ საერთო საიდუმლო გასაღების გამომუშავება მოწინააღმდეგის მიერ სრულად კონტროლირებად ღია არხში.

ალგორითმის შესასრულებლად ორივე პიროვნება გამგზავნი და მიმღები(ადრესატი) უნდა შეთანხმდნებ p მარტივ და $\mathbf{g}\epsilon F_p^*$ რიცხვს. რაც ყველაზე მთავარია, არ არის საჭირო ამ რიცხვების დამალვა მესამე პირისგან. არ არის საჭირო დახურული არხი.

საერთო საიდუმლო გასაღების გამომუშავების ეტაპი.

1. პირველი პიროვნება (გამგზავნი) შეარჩევს საიდუმლო a რიცხვს $a\epsilon\{2,\dots,p-2\}$, და გამოთვლის $A_1= g^a \bmod p$ რიცხვს და მიღებულ

შედეგს გაუგზავნის მიმღებს. ხოლო a რიცხვს ინახავს საიდუმლოდ.

2. მეორე პიროვნება (მიმღები) შეარჩევს საიდუმლო y რიცხვს $y \in \{2, \dots, p-2\}$, და გამოთვლის $B = g^b \bmod p$ რიცხვს და მიღებულ შედეგს გაუგზავნის პირველ პიროვნებას. ხოლო b რიცხვს ინახავს საიდუმლოდ.
3. პირველი პიროვნება გამოთვლის $K = B^a \bmod p$. მას უკვე შეუძლია წაშალოს თავისი საიდუმლო რიცხვი, რომ ის მესამე პირმა არ აღმოაჩინოს.
4. მეორე პიროვნებაც მიღებული გამოთვლის $K = A^b \bmod p$. მასაც შეუძლია წაშალოს თავისი საიდუმლო გასაღები.
5. შემდგომი ეტაპი იმითაა საინტერესო, რომ ორივე პიროვნების შედეგი ახარისხების მიუხედავად ერთი და იგივე რიცხვია.

$$K = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p$$
$$K = (g^b \bmod p)^a \bmod p = g^{ba} \bmod p$$

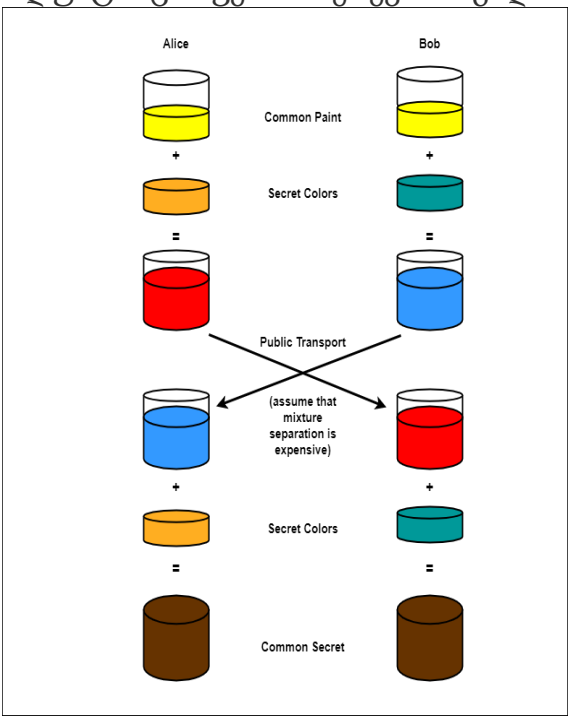
აღნიშნული რიცხვი წარმოადგენს საერთო საიდუმლო გასაღებს.

როგორც ზემოთ ჩამოყალიბებულიდან ჩანს, დიფი-ჰელმანის ალგორითმი გამოიყენება ღია გასაღებების გენერირების მიზნით. გარდა ამისა, ამ ალგორითმს იყენებენ „ინტეგრებულ დაშიფრვის სქემებში“; პაროლით აუთენტიფიკაციის შემთხვევაში, როცა ორი პიროვნება ერთობლივად (სხვადასხვა ადგილიდან) იყენებს ერთი და იგივე

პაროლს; პირდაპირი გასაიდუმლოების პროტოკოლებში, როცა ყოველი ახალი სეანსისათვის ახალი გასაღებების გენერირება ხდება საჭირო. ამ შემთხვევაში სეანსის დასრულების შემდეგ პაროლები უბრალოდ იშლება.

ზემოთ ჩამოყალიბებულიდან ჩანს, რომ გამგზავნი და მიმღები ერთმანეთს შეიძლება საერთოდ არ იცნობდნენ და კონტაქტს ამყარებდნენ მხოლოდ დაუცველი არხით, გასაღებების გაცვლაც ღია არხით ხდება. ხოლო საიდუმლო გასაღების ერთობლივი მოხმარება უსაფრთხოა. თუმცა დიფი ჰელმანის ალგორითმი არ გამოიყენება ყველანაირ ასიმეტრიულ ალგორითმებში. ეს ალგორითმი არ გამოიყენება არც ციფრული ხელმოწერებისათვის. დიფი-ჰელმანის ალგორითმის გატეხვა შესაძლებელია მესამე პირის მიერ თუნდაც იმ მიზეზით, რომ ის არ გულისხმობს გამგზავნი-მიმღები მხარეების აუთენტიფიკაციას.

დიფი-ჰელმანის ალგორითმის ცნობილი ილუსტრაცია ფერთა შერევის მაგალითზე.



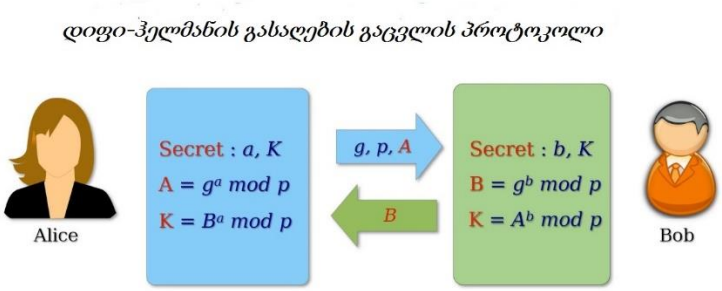
ნახაზი 16

კრიპტოანალიზი:

დიფი-ჰელმანის ალგორითმის უსაფრთხოება, ეფუძნება დისკრეტული ლოგარითმის პრობლემას და პარამეტრების სწორად შერჩევის შემთხვევაში, თუ მიმტაცებელი განახორციელებს სისტემაზე მხოლოდ პასიურ შეტევას, მაშინ ალგორითმის გატეხვა შეუძლებელია, მაგრამ მიმტაცებელს

ყოველთვის აქვს შესაძლებლობა როგორც გამგზავნთან , ასევე მიმღებთანაც ფარულად შექმნას საიდუმლო გასაღები და შიფროგრამის მიტაცების შემთხვევაში გაშიფროს ეს შეტყობინება. აქედან გამომდინარეობს, რომ დამატებითი პროცედურების გარეშე დიფი-ჰელმანის ალგორითმის გამოყენება არ შეიძლება, ამიტომ საჭიროა პროტოკოლში გათვალისწინებული იქნას დამატებითი პროცედურა, რომლის საშუალებითაც მოხდება შეტყობინების ავტორის აუტენთიფიკაცია.

დიფი - ჰელმანის გასაღების გაცვლის პროტოკოლი



სავარჯიშო N1

გასაღების ფორმირება დიფი - ჰელმანის ასიმეტრიული მეთოდის გამოყენება

$$p = 11;\ a = 3;\ x = 2;\ y = 6;$$

$$X \text{ მხარე გამოთვლის } C_1 = a^x \equiv C_1(modp)$$

$$\equiv 3^2(mod11) \equiv 9(mod11);$$

Y *მხარე გამოთვლის* $C_2 = a^y \equiv C_2(modp)$

$\equiv 3^6(mod11) \equiv 729(mod11)$

$\equiv 3(mod11);$

Y *მხარე გამოთვლის* $K_1 = (C_1)^y$

$\equiv a^{xy} \equiv K_1(modp) \equiv 9^6(mod11)$

$\equiv 9(mod11);$

X *მხარე გამოთვლის* $K_2 = (C_2)^x \equiv a^{yx} \equiv K_2(modp)$

$\equiv 3^2(mod11) \equiv 9(mod11);$

$K_1 = K_2 = K;$

ე. ი. K= 9;

განვახორციელოთ შიფრაცია - დეშიფრაცია $M = 7$ და K' შერჩევით:

$KK' \equiv 1(mod(p-1))$

$KK' \equiv 1(mod10)$

$9K' \equiv 1(mod10)$

$K' \equiv 9;$

X მხარე დაშიფრავს K გასაღებით. მიღებულ C შიფრო ტექსტს გადაუგზავნის Y მხარეს, რომელიც თავისი $K' = 9$, გასაღებით გაშიფრავს .

$C = M^k = 7^9 \equiv 40353607(mod11) \equiv 8(mod11);$

$M = C^{K'} \equiv 8^9 \equiv 134217728(mod11) \equiv 7(mod11);$

$M = 7.$

სავარჯიშო N2

გასაღების ფორმირება დიფი - ჰელმანის ასიმეტრიული მეთოდის გამოყენება.

$p = 11;\ a = 3;\ x = 3;\ y = 5;$

X მხარე გამოთვლის $C_1 = a^x \equiv C_1(modp)$
$\equiv 3^3(mod11) \equiv 27(mod11)$
$\equiv 5(mod11);$
Y მხარე გამოთვლის $C_2 = a^y \equiv C_2(modp)$
$\equiv 3^5(mod11) \equiv 243(mod11)$
$\equiv 1(mod11);$
Y მხარე გამოთვლის $K_1 = (C_1)^y$
$\equiv a^{xy} \equiv K_1(modp) \equiv 5^5(mod11)$
$\equiv 3125(mod11) \equiv 1(mod11);$
X მხარე გამოთვლის $K_2 = (C_2)^x \equiv a^{yx} \equiv K_2(modp)$
$\equiv 1^3(mod11) \equiv 1(mod11);$
$K_1 = K_2 = K;$
ე. ი. K= 1.

სავარჯიშო N3

გასაღების ფორმირება დიფი - ჰელმანის ასიმეტრიული მეთოდის გამოყენებ

$p = 21;\ a = 3;\ x = 4;\ y = 5;$
X მხარე გამოთვლის $C_1 = a^x \equiv C_1(modp)$
$\equiv 3^4(mod21) \equiv 81(mod21)$
$\equiv 18(mod21);$
Y მხარე გამოთვლის $C_2 = a^y \equiv C_2(modp)$
$\equiv 3^5(mod21) \equiv 243(mod21)$
$\equiv 12(mod21);$
Y მხარე გამოთვლის $K_1 = (C_1)^y$
$\equiv a^{xy} \equiv K_1(modp) \equiv 18^5(mod21)$
$\equiv 1889568(mod21) \equiv 9(mod21);$
X მხარე გამოთვლის $K_2 = (C_2)^x \equiv a^{yx} \equiv K_2(modp)$
$\equiv 12^4(mod21) \equiv 20736(mod21)$
$\equiv 9(mod21);$
$K_1 = K_2 = K;$
ე. ი. K= 9.

განახორციელეთ მონაცემის შიფრაცია-დეშიფრაცია დიფი - ჰელმანის ასიმეტრიული მეთოდით .

გასაღების ფორმირება დიფი - ჰელმანის ასიმეტრიული მეთოდის გამოყენებ

$$p = 17;\ a = 2;\ x = 3;\ y = 2;$$

$$X\ \text{მხარე გამოთვლის}\ C_1 = a^x \equiv C_1(modp) \equiv 2^3(mod17) \equiv 8(mod17);$$

$$Y\ \text{მხარე გამოთვლის}\ C_2 = a^y \equiv C_2(modp) \equiv 2^2(mod17) \equiv 4(mod17);$$

$$Y\ \text{მხარე გამოთვლის}\ K_1 = (C_1)^y \equiv a^{xy} \equiv K_1(modp) \equiv 8^2(mod17) \equiv 64(mod17) \equiv 13(mod17);$$

$$X\ \text{მხარე გამოთვლის}\ K_2 = (C_2)^x \equiv a^{yx} \equiv K_2(modp) \equiv 4^3(mod17) \equiv 64(mod17) \equiv 13(mod17);$$

$K_1 = K_2 = K;$

ე. ი. K= 13.

განვახორციელოთ შიფრაცია - დეშიფრაცია $M = 3$ და K' შერჩევით:

$$KK' \equiv 1(mod(p-1))$$

$$KK' \equiv 1(mod16)$$

$$13K' \equiv 1(mod16)$$

$K' \equiv 5;$

X მხარე დაშიფრავს K გასაღებით. მიღებულ C შიფროტექსტს გად

რომელიც თავისი $K' = 5$,

გასაღებით გაშიფრავს .

$$C = M^k = 3^{13} \equiv 1594323(\mathrm{mod}17) \equiv 12(\mathrm{mod}17);$$
$$M = C^{K'} \equiv 12^5 \equiv 248832(\mathrm{mod}17) \equiv 3(\mathrm{mod}17);$$
$$M = 3.$$

პროგრამული რეალიზაცია:

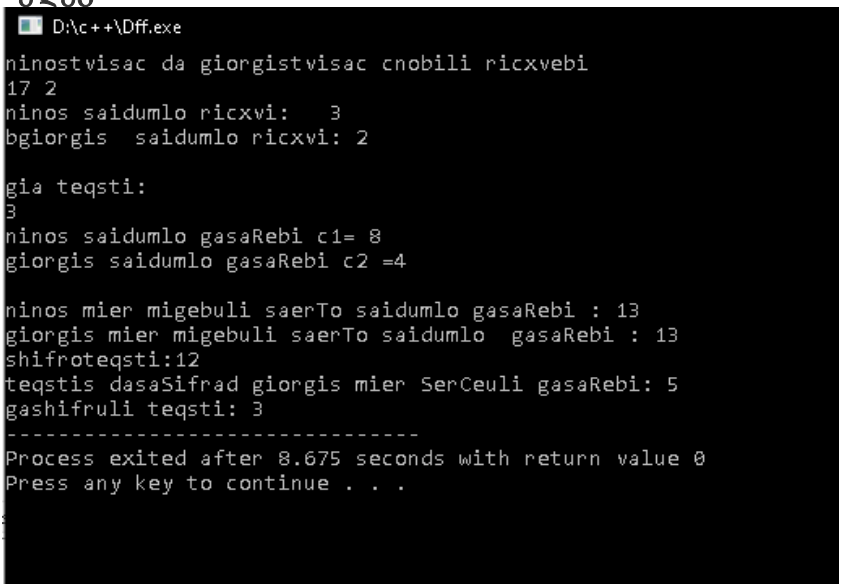
```
#include<iostream>
#include<stdlib.h>
#include<math.h>
#include<time.h>

using namespace std;

class Dh
{
    public:
        long long int p,a,x,y,c1,c2,
k1,k2,K,C,w,m,D,kk;
        bool f;

        Dh(long long int p1,long long int a1,long long
int x1,long long int y1, long long int m1)
        {
            p = p1;
            a = a1;
            x = x1;
            y = y1;
            m=m1;
            f=true;
            c1=gamotvla(a,x,p);
```

```
                c2=gamotvla(a,y,p);
                k1 = gamotvla(c2,x,p);
                k2 = gamotvla(c1,y,p);
               C=gamotvla(m,k1,p);
              for(int i=1;i<=p-1 && f;i++)

if(gamotvla(C,i,p)==m){D=gamotvla(C,i,p); w=i;f=false;}
                                          }



        long long int gamotvla(long long int  a1, long
long int x1, long long int p1)

        {

            long long int t;
            if(x1==1)
                return a1;
            t=gamotvla(a1,x1/2,p1);
            if(x1%2==0)
                return (t*t)%p1;
            else
                return (((t*t)%p1)*a1)%p1;
        } };

int main()
```

```
{
    long long int p,a,x,y,c1,c2,C,D,K,w,m;


    cout<<"ninostvisac da giorgistvisac cnobili ricxvebi
"<<endl;
    cin>>p>>a;

    cout<<"ninos saidumlo ricxvi:   ";
    cin>>x;
    cout<<"bgiorgis  saidumlo ricxvi: ";
    cin>>y;
    cout<<endl;
    cout<<"gia teqsti:"<<endl;
    cin>>m;
   Dh kay(p,a,x,y,m);

    cout<<"ninos saidumlo gasaRebi c1=
"<<kay.c1<<endl;
    cout<<"giorgis saidumlo gasaRebi c2
="<<kay.c2<<endl;
    cout<<endl;
    cout<<"ninos mier migebuli saerTo saidumlo
gasaRebi : "<<kay.k1<<endl;
    cout<<"giorgis mier migebuli saerTo saidumlo
gasaRebi : "<<kay.k2<<endl;
  cout<<"shifroteqsti:"<<kay.C<<endl;
  cout<<kay.w<<endl;
```

```
    cout<<"gashifruli teqsti: "<<kay.D;
}
```

შედეგი:



დავალება 1: განახორციელეთ გასაღების ფორმირება და მონაცემის შიფრაცია- დეშიფრაცია დიფი - ჰელმანის ასიმეტრიული მეთოდით.

$$p = 13;\ \ a = 6;\ \ x = 4;\ \ y = 5;\ M = 5.$$

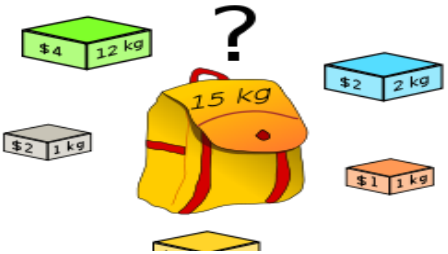
დავალება 2: განახორციელეთ გასაღების ფორმირება და მონაცემის შიფრაცია-დეშიფრაცია დიფი - ჰელმანის ასიმეტრიული მეთოდით.

$$p = 17;\ \ a = 5;\ \ x = 7;\ \ y = 3;\ M = 4.$$

დავალება 3: განახორციელეთ გასაღების ფორმირება და მონაცემის შიფრაცია- დეშიფრაცია დიფი - ჰელმანის ასიმეტრიული მეთოდით.
$p = 47;\ \ a = 23;\ \ x = 12;\ \ y = 33;\ M = 2.$

მერკლი-ჰელმანის ალგორითმი

1978 წელს გამოქვეყნდა მერკლი - ჰელმანის ცნობილი სტატია, რომელშიც აღწერილი იყო ღია გასაღებიანი (ასიმეტრიული) კრიპტოსისტემა, დაფუძნებული ზურგჩანთის ამოცანის ერთერთ კერძო შემთხვევაზე, რომელიც შეიძლება ჩამოვაყალიბოთ შემდეგი სახით: დავუშვათ, მოცემული გვაქვს რაიმე ობიექტების სასრული სიმრავლე, რომლებიც შეიძლება დავახასიათოთ ორი პარამეტრით: ზომა და ფასი. გვაქვს აგრეთვე ზურგჩანთა, რომლის ზომა რაიმე მუდმივი რიცხვია. ჩვენი მიზანია ჩავალაგოთ ზურგჩანთაში ნივთები ისე, რომ შევავსოთ ზურგჩანთა და ამავე დროს ამ ნივთების ჯამური ფასი იყოს მაქსიმალური.



ნახაზი- 17 ზურგჩანთა

დავუშვათ არის ვექტორი, $b = \{b_1, b_2, \dots\ b_n\}$ რომლის ელემენტები მოცემული ნივთების ზომებია და $p = \{p_1,\ p_2, \dots\ p_n\}$ ვექტორი წარმოადგენს ამ ნივთების ფასებს.

შემოვიღოთ კიდეც ერთი ვექტორი $X = (x_1, x_2, \dots, x_n)$ სადაც

მაშინ ჩვენი ამოცანა მიიღებს შემდეგ სახეს:

$$\begin{cases} \sum_{i=1}^{n} x_i w_i \le V \\ max \sum_{i=1}^{n} x_i p_i \end{cases}$$

არსებობს ამ ამოცანის სხვადასხვა ვარიანტები. მერკლი–ჰელმანის კრიპტოსისტემა დაფუძნებულია ზურგჩანთის ამოცანის ერთ-ერთ კერძო შემთხვევაზე, რომელიც შეიძლება ჩამოვაყალიბოთ შემდეგი სახით: მოცემულია V მოცულობის ზურგჩანთა და $B = \{b_1, b_2, \dots b_n\}$ საგანთა სიმრავლე, რომელთაც გააჩნიათ გარკვეული მოცულობები. ჩვენი ამოცანაა ვიპოვოთ B სიმრავლის ისეთი $B_i \subseteq B$ ქვესიმრავლე, რომლის ელემენტებისთვისაც შესრულდება ტოლობა

$$V = \sum_{i=1}^{n} b_i \cdot x_i \ ,$$

სადაც $x_i \in \{0,1\}, i = 1,2,...,n$. თუ $x_i = 1$, ეს ნიშნავს, რომ i-ური საგანი უნდა მოვათავსოთ ზურგჩანთაში, ხოლო თუ $x_i = 0$ - საგანი არ ჩავდოთ ზურგჩანთში. როგორც ცნობილია, ზურგჩანთის ამოცანა მიეკუთვნება NP სირთულის ამოცანათა კლასს, მაგრამ ამ კერძო შემთხვევისთვის, თუ B სიმრავლე წარმოადგენს ზეზრდად მიმდევრობას, ანუ მიმდევრობის ყოველი b_i წევრი აკმაყოფილებს პირობას

$$b_i > \sum_{j=1}^{i-1} b_j \ , \qquad (1)$$

მაშინ არსებობს ამოცანის ამოხსნის წრფივი სირთულის ალგორითმი

მერკლიმ და ჰელმანმა ამ თვისების გამოყენებით ააგეს ღიაგასაღებიანი კრიპტოსისტემა, რომელშიც ღია, დაშიფვრის გასაღებს წარმოადგენს $A = \{a_1, a_2, ...a_n\}$ არაზეზრდადი მიმდევრობა, სადაც A მიმდევრობის თითოეული a_i წევრი მიიღება შემდეგი წესით:

$a_i = b_i \cdot t (\mathrm{mod}\, m)$,

სადაც $m,t \in Z$ და დაკმაყოფილებულია შემდეგი პირობები: $m > \sum_{i=1}^{n} b_i$, $(t,m) = 1$. თუ პირველ პირობას შევცვლით უფრო მარტივი პირობით $m > \max b_i$, მაშინ გვქნება, რომ A მიმდევრობა მიიღება B მიმდევრობისგან მოდულარული გამრავლებით (t,m) წყვილის მიმართ. მაგრამ ეს ნიშნავს, რომ B მიმდევრობა ასევე მიიღება A მიმდევრობისაგან (u,m) წყვილის მიმართ მოდულარული გამრავლებით, სადაც $u = t^{-1}$ (ასეთი u აუცილებლად არსებობს მეორე პირობის თანახმად). თუ ჩვენ ავიღებთ $m > \sum_{i=1}^{n} b_i$ პირობას, ამბობენ, რომ A მიმდევრობა მიიღება B მიმდევრობისგან ძლიერი მოდულარული გამრავლებით (t,m) წყვილის მიმართ. ამ შემთხვევაში აღარ შეიძლება ვთქვათ, რომ B მიმდევრობა ასევე მიიღება A მიმდევრობისაგან (u,m) წყვილის მიმართ ძლიერი მოდულარული გამრავლებით, რადგანაც $m > \sum_{i=1}^{n} a_i$ უტოლობა ზოგადად აღარ სრულდება.

საიდუმლო, დეშიფრაციის გასაღებია (B, m, t) სამეული. ღია ტექსტი, რომელიც წარმოდგენს ნულების და ერთების მიმდევრობას, დაშიფვრის დროს დაიყოფა n სიგრძის L რაოდენობის ბლოკად და ასრულებს $x_i \in \{0,1\}$ სიმრავლის როლს. დაშიფრულ ტექსტს წარმოადგენს $S_1, S_2, ..., S_L$ ჯამები, რომლებიც გამოითვლება ფორმულით:

$$S_j = \sum_{i=1}^{n} x_{ij} \cdot a_i \quad . \qquad (2)$$

ღია ტექსტის აღსადგენად საჭიროა ამოიხსნას ზურგჩანთის ამოცანის ზემოთ მოყვანილი ვარიანტი წრფივი სირთულის ალგორითმით, როდესაც ცნობილია B ზეზრდადი მიმდევრობა და m და t პარამეტრები. ამისათვის თითოეული ჯამი მრავლდება t^{-1} მოდულით m

$$S_j^{/} = S_j \cdot t^{-1} (\mathrm{mod}\, m). \qquad (3)$$

და ხდება ზურგჩანთის ამოცანის ამოხსნა ცალ-ცალკე თითოეული $S_j^{/}$ ჯამისთვის ნახსენები წრფივი სირთულის ალგორითმით, როდესაც ცნობილია B ზეზრდადი მიმდევრობა.

კრიპტოანალიზი:

მოწინააღმდეგემ, რომლისთვისაც უცნობია დეშიფრავიის გასაღები, გატეხოს სისტემა და იპოვოს ღია ტექსტი, მოუწევს ამოხსნას NP სირთულის ამოცანა, რაც პრაქტიკულად შეუძლებელია, როდესაც *B* მიმდევრობაში წევრების რაოდენობა იცვლება ორასიდან სამას ელემენტამდე, რაც იმას ნიშნავს, რომ შეტევა კრიპტოალგორითმზე არ მოგვცემს შედეგს. როგორც, ცნობილმა კრიპტოლოგმა ა.შამირმა აჩვენა,რომ ამ სისტემის სუსტი წერტილია ღია გასაღების მიღების პროცედურა. ეს პროცედურა არ იყენებს ცალმხრივმიართულ ფუნქციას, როგორც ეს არის სხვა ღიაგასაღებიან სისტემებში, რადგანაც მოდულით გამრავლების ოპერაცია არ წარმოადგენს ცალმხრივმიმართულ ფუნქციას. ა. შამირმა შექმნა ღია გასაღებიდან საიდუმლო გასაღების აღდგენის პოლინომური სირთულის ალგორითმი და გატეხა სისტემა. ამას ხელი შეუწყო იმანაც, რომ, როგორც შამირმა აჩვენა, სულაც არაა საჭირო ვიპოვოთ ზუსტად ის (t_0,m_0) წყვილი, რომლითაც *B* ზეზრდადი მიმდევრობიდან მიღებული იქნა ღია გასაღები - *A* არაზეზრდადი მიმდევრობა. ნებისმიერი ზეზრდადი მიმდევრობა,

რომლიდანაც შესძლებელია მოცემული A არაზეზრდადი მიმდევრობის მიღება შეიძლება გამოვიყენოთ საიდუმლო გასაღებად. ამ სისუსტეების გამოყენებით ა. შამირმა შექმნა კრიპტოსისტემზე შეტევის ალგორითმი, რომელიც შედგება ორი ნაწილისაგან. პირველ ნაწილში ალგორითმი ეძებს ისეთ მთელ რიცხვებს, რომლებისთვისაც კმაყოფილდება პირობა, რომ u/m სიდიდე რამდენიმე a_i-თვის მდებარეობს ამ ფუნქციათა საერთო მინიმალურ ინტერვალში. ასეთი რიცხვების მოძებნის შემდეგ, ალგორითმი დიოფანტეს აპროქსიმაციის მეთოდით ეძებს (u,m) წყვილს, რომლის საშუალებითაც ღია გასაღებიდან შესაძლებელი იქნება საიდუმლო გასაღების გამოთვლა.

მაგალითი: მოცემული გვაქვს:
ღია ტექსტი:“luka”;
საიდუმლო გასაღები: 1, 2, 5, 9, 19, 45, 91,174;
m და t პარამეტრები: t=5, m=347;
მერკლი ჰელმანის ალგორითმის გამოყენებით დავშიფროთ მოცემული ღია ტექსტი და შემდეგ გავშიფროთ.

ამოხსნა:

საიდუმლო გასაღებიდან ღია გასაღების გამოთვლა.

საიდუმლო გასაღები	1	2	5	9	19	45	91	174
ღია გასაღები $b_i = k\, a_i (mod\ m)$ $=5\ a_i (mod\ 347)$	5	10	25	45	95	225	108	176

ტექსტი, რომელიც კომპიუტერულ სისტემაში წარმოიდგინება როგორც ბიტური სტრიქონი, დაიყოფა L რაოდენობის ბლოკად. ყოველ ბლოკში არის n ბიტი. ვითვლით

$$s_j = \sum_{i=1}^{n} x_{ij}\, b_j$$

სადაც x_{ij} არის ღია ტექსტის i ბლოკის j ელემენტი.

$S1, S2, \cdots, SL$ არის დაშიფრული ტექსტი.

ღია ტექსტი		*წონების ჯამი*	*შიფროგრამა*
სიმბოლო	ბინარ. კოდი		
L	0110110	10+25+95+225	355

	0		
U	01110101	10+25+45+225+176	481
K	01101011	10+25+95+108+76	414
A	01100001	10+25+76	211

5	10	25	45	95	225	108	176

(ღია გასღები)

დეშიფრაციისთვის ყოველი S_i მრავლდება k^{-1}-ზე მოდულით m

$$S_j' = S_j\, k^{-1} (mod\ m);$$

შიფროგრამა	$S_j\ k^{-1}(mod\ m)= S_j 139 (mod\ 247)$	წონების ჯამი	ღია ტექსტი	
			ბინარ.კოდი	სიმბოლოები
355	*71*	2+5+19+45	01101100	L
481	*235*	2+5+9+45+174	01110101	U
414	*291*	5+9+45+91+174	01101011	K

211			*181*		2+5+174		01100001	A
1	2	5	9	19	45	91	174	

(საიდუმლო გასაღები)

პროგრამული რეალიზაცია:

```
#include <cstdlib>
#include <iostream>
#include<time.h>
#include<vector>
#include<math.h>
#define n 8
using namespace std;

 int      y[]={1,2,5,9,19,45,91,174};          //საიდუმლო
//გასაღები
class zurgchanta
{
     public:
      zurgchanta();
      void ecoding();
      void decoding();
   ~zurgchanta();
    private:
```

```
        int **a, b[n], *s,*s1,k;
        int m,t;

      char c,*T;
      unsigned char x;
      vector<int>v,v1;
      vector<vector<int> >A;
     };

 zurgchanta::zurgchanta()  //ღია ტექსტის შემოტანა
//და ბინარული სახით გარდაქმნა
{ int i;
     c='a';

while((c=getchar())!='\n')
{x=0x80;
for(i=0; i<8; i++)
{if((c&x)==0) v.push_back(0);
 else  v.push_back(1);

   x>>=1;}
   }
  cout<<"gia texti binaruli saxit:"<<endl;
   for(int i=0; i<v.size(); i++)
```

```
   cout<<v[i];
   cout<<endl;
     k=v.size()/n;
   int l=0 ;
    a=new int*[k];
for(i=0;i<k;i++)
a[i]=new int[n];
cout<<"blokebad dayofili gia teqsti:"<<endl;// ღია
//ტექსტის ბლოკებად დაყოფა, მატრიცის
//სახით წარმოდგენა
   for( i=0; i<k; i++)
   {
   for(int j=0; j<n; j++)
   {
    a[i][j]=v[l];
    cout<<a[i][j];
    l++;}
    v.clear();
    cout<<endl;}
    m=347;t=5;
    cout<<"gia gasagebi:"<<endl;
    for(i=0;  i<n;i++) //საიდუმლო გასაღებიდან ღია
//გასაღების მიღება.
{
```

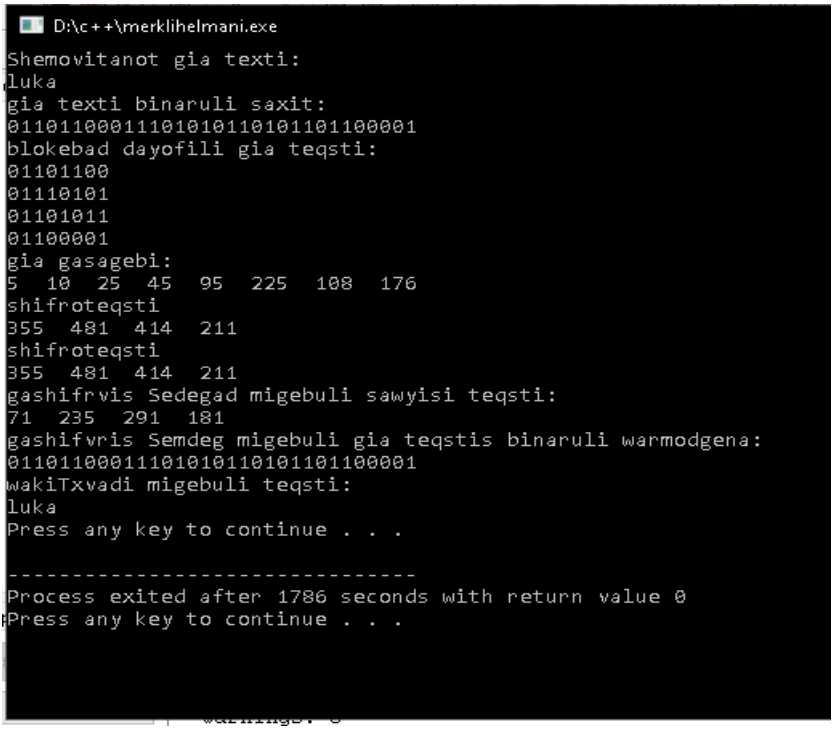
```
 b[i]=(t*y[i])%m;
 cout<<b[i]<<"  ";

}
cout<<endl;}
void zurgchanta::ecoding()//დაშიფვრა
{int i,j;
   s=new int[k];
   cout<<"shifroteqsti"<<endl;
   for(i=0;i<k;i++)
{s[i]=0;
for(j=0; j<n; j++)
s[i]=s[i]+a[i][j]*b[j];
cout<<s[i]<<"  ";
}
cout<<endl;
}
void zurgchanta::decoding()//გაშიფვრა
{int i;
s1=new int[k];
ecoding();
cout<<"gashifrvis Sedegad migebuli sawyisi
teqsti:"<<endl;
for( i=0;i<k; i++)
```

```
{
s1[i]=(s[i]*139)%m;
cout<<s1[i]<<"  ";}
cout<<endl;
cout<<"gashifvris Semdeg migebuli gia teqstis binaruli warmodgena:"<<endl;
for(i=0;i<k; i++)
{int j=n-1;
while(j>=0)
     {  if(s1[i]<y[j]) v.push_back(0);
       else { v.push_back(1);
       s1[i]=s1[i]-y[j];}
       j--;
       }
for(int l=n-1;l>=0;l--)
 v1.push_back(v[l]);

A.push_back(v1);
v1.clear();
v.clear();
       }
for(int i=0; i<k; i++)
for (int j=0; j<n; j++)
 cout<<A[i][j];
```

```
   cout<<endl;
   T=new char[k];
   cout<<"wakiTxvadi migebuli teqsti:"<<endl;
 for(int i=0;i<k;i++)
{T[i]=0;
for(int l=0;l<n;l++)
T[i]|=A[i][l]<<(7-l);  }
    for(int i=0; i<k; i++)
       cout<<T[i];
   cout<<endl;
}
zurgchanta::~zurgchanta()
{
     A.clear();
                    }

 main()
{int i;
cout<<"Shemovitanot gia texti:"<<endl;
      zurgchanta io;
      io.ecoding();
      io. decoding();
   system("PAUSE");
   return EXIT_SUCCESS;
```

}

შედეგი:



დავალება:

მოცემული გვაქვს:

ღია ტექსტი:“Welcome to Georgia”;

საიდუმლო გასაღები: 1, 3, 6, 11, 22, 46, 101 ,225;

m და t პარამეტრები: t=5, m=231;

მერკლი ჰელმანის ალგორითმის გამოყენებით

დავშიფროთ მოცემული ღია ტექსტი და შემდეგ გავშიფროთ.

ელგამალის კრიპტოსისტემა

ელგამალის (Elgamal) სქემა წარმოადგენს ღია გასაღებიან კრიპტოსისტემას, რომელიც დაფუძნებულია დისკრეტული ლოგარითმების გამოთვლის სირთულეზე სასრულ ველებში. მოცემული g და a-სთვის, a ელემენტის g ფუძით დისკრეტულ ლოგარითმს უწოდებენ $g^x = a$ სახის განტოლების x ამონახსნს. ვთქვათ, აღნიშნული განტოლება განიხილება რაიმე სასრულ მულტიპლიკატიურ სასრულ აბელის G ჯგუფში. ამოვხსნათ ეს განტოლება ნიშნავს, ვიპოვოთ ისეთი არაუარყოფითი მთელი x რიცხვი, რომელიც აკმაყოფილებს მოცემულ $g^x = a$ სახის განტოლებას. თუ განტოლება ამოხსნადია, მაშინ მას აუცილებლად უნდა ჰქონდეს ერთი მაინც ნატურალური ამონახსნი, რომელიც ჯგუფის რიგს არ აღემატება. აქედან ამონახსნის პოვნის ალგორითმის უხეში შეფასება გამომდინარეობს: სრული გადარჩევის ალგორითმის მეშვეობით შესაძლებელია ამონახსნის პოვნა იმ რაოდენობის

ბიჯების მეშვეობით, რომელიც არ აღემატება მოცემული ჯგუფის რიგს. ყველაზე ხშირად განიხილავენ შემთხვევას, როცა G ციკლური ჯგუფია, რომელსაც g ელემენტი წარმოშობს. სწორედ ამ შემთხვევაში აღნიშნულ განტოლებას ყოველთვის აქვს ამონახსნი. სხვა დანარჩენ შემთხვევაში ზემოთ მოყვანილი განტოლების ამონახსნების პოვნა დამატებით კვლევებს ითხოვს.

ელგამალის კრიპტოსისტმე შეიცავს შიფრაციის ალგორითმს და ციფრული ხელმოწერის ალგორითმს. ელგამალის სქემას წლების განმავლობაში ამერიკაში (DSA) და რუსეთში (ГОСТ Р 34.10-94) იყენებდნენ ელექტრონული ციფრული ხელმოწერისათვის. აღნიშნული სქემა შემოთავაზებული იყო 1985 წელს ტაჰერ ელგამალის მიერ. ელგამალმა შეიმუშავა დიფი-ჰელმანის ალგორითმის ერთ-ერთი ვარიანტი. მან მიიღო დიფი-ჰელმანის ალგორითმის მოდიფიკაციით ორი ალგორითმი: ერთი შიფრაციისთვის, ხოლო მეორე აუთენტიფიკაციისთვის. RSA ალგორითმისგან განსხვავებით, ელგამალის ალგორითმი არ იყო

დაპატენტებული და მის გამოყენებას არ ესაჭროებოდა ლიცენზიის თანხის გადახდა. ამ მიზეზით ის RSA-ს უფრო იაფიან ანალოგს წარმოადგენდა. ამჟამად ითვლება, რომ ელგამალის ალგორითმი დიფი-ჰელმანის ალგორითმის მოდიფიკაციას წარმოადგენს.

გასაღების გენერაციის სქემა:

1. გენერირდება შემთხვევითი მარტივი P რიცხვი .
2. შეირჩევა მთელი g რიცხვი, P რიცხვის პირველადი ფესვი.
3. შეირჩევა შემთხვევითი მთელი x რიცხვი, ისეთი რომ $1 < x < p - 1$
4. გამოითვლება $y = g^x \bmod P$
5. ღია გასაღებს წარმოადგენს (y, g, P), ხოლო დახურულ გასაღებს კი - x რიცხვი.

დაშიფრვის დროს M შეტყობინება უნდა იყოს P რიცხვზე ნაკლები. შეირჩევა სესიის გასაღები - k შემთხვევითი მთელი რიცხვი, რომელიც ურთიერთმარტივია $(p - 1)$-თან, ისეთი რომ $1 < k < p - 1$. შემდეგ გამოითვლება ორი რიცხვი $a = g^k \bmod P$ და $b = y^k M \bmod P$. (a, b) რიცხვების წყვილი წარმოადგენს შიფროტექსტს. რთული

შესამჩნევი არაა, რომ შიფროტექსი ორჯერ უფრო გრძელია, ვიდრე საწყისი ტექსტი.

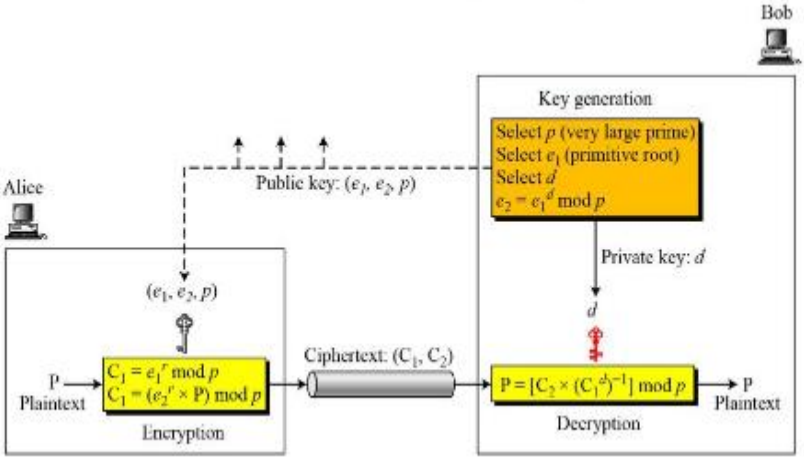
დეშიფრაციის დროს მოქმედებათა თანმიმდევრობა შემდეგია:

თუ ვიცით x დახურული გასაღები, თავდაპირველი შეტყობინების მიღება შესაძლებელია (a, b) წყვილის მეშვეობით. კერძოდ, უნდა გამოვიყენოთ ფორმულა

$M = b(a^x)^{-1} \bmod P$. ამასთან ერთად, $(a^x)^{-1} = g^{-kx} \bmod P$. ამიტომაც, $b(a^x)^{-1} = \left(y^k M\right)g^{-xk} \equiv \left(g^{kh} M\right)g^{-xk} \equiv M (\bmod P)$.

პრაქტიკული გამოთვლებისათვის ყველაზე კომფორტულია ფორმულა:

$M = b(a^x)^{-1} = ba^{(p-1-x)} (\bmod P)$.



ნახაზი- 18 შიფრაცია-დეშიფრაცია ელგამალის ალგორითმით

განვიხილოთ მაგალითი :

განახორციელეთ მონაცემების შიფრაცია - დეშიფრაცია, როდესაც

$$p = 11;\ g = 2;\ x = 8;\ k = 9;\ M = 5.$$

I გასაღების გენერირება

$$y = g^x \left(\mathrm{mod}(p)\right) \equiv 2^8 \mathrm{mod} 11 = 256 \mathrm{mod} 11$$
$$\equiv 3 \mathrm{mod} 11$$

მივიღეთ: ღია გასაღები: $\{p, g, y\} \rightarrow \{11, 2, 3\}$

დახურული გასაღები: $\{x\} \rightarrow \{8\}$

II მონაცემის შიფრაცია

$$a = g^k \left(\mathrm{mod}(p)\right);\ b = y^k M \left(\mathrm{mod}(p)\right);$$
$$a = 2^9 \mathrm{mod} 11 \equiv 512 \mathrm{mod} 11 \equiv 6 \mathrm{mod} 11;$$
$$b = 3^9 * 5 \mathrm{mod} 11 \equiv 98415 \mathrm{mod} 11 \equiv 9 \mathrm{mod} 11;$$

III მონაცემის დეშიფრაცია

$$M = b(a^x)^{-1} mod p;$$

$$M = 9 * 6^2 mod 11 \equiv 324 mod 11 \equiv 5$$

განვიხილოთ მაგალითი :

განახორციელეთ მონაცემების შიფრაცია - დეშიფრაცია, როდესაც

$$p = 11;\ g = 3;\ x = 2;\ K = 2;\ \ M = 7.$$

I გასაღების გენერირება

$$y = \ g^x \left(mod(p)\right) \equiv 3^2 mod 11 = 9 mod 11$$

მივიღეთ: ღია გასაღები: $\{p, g, y\} \rightarrow \{11, 3, 9\}$

დახურული გასაღები: $\{x\} \rightarrow \{2\}$

II მონაცემის შიფრაცია

$$a = g^k\big(mod(p)\big);\ \ b = y^k M\big(mod(p)\big);$$

$$a = 3^2 mod 11 \equiv 9 mod 11;$$

$$b = 9^2 * 7 mod 11 \equiv 567 mod 11 \equiv 6 mod 11;$$

III მონაცემის დეშიფრაცია

$$M = b(a^x)^{-1} mod p;$$

$$M = 6 * 9^8 mod 11 \equiv 258280326 mod 11 \equiv 7.$$

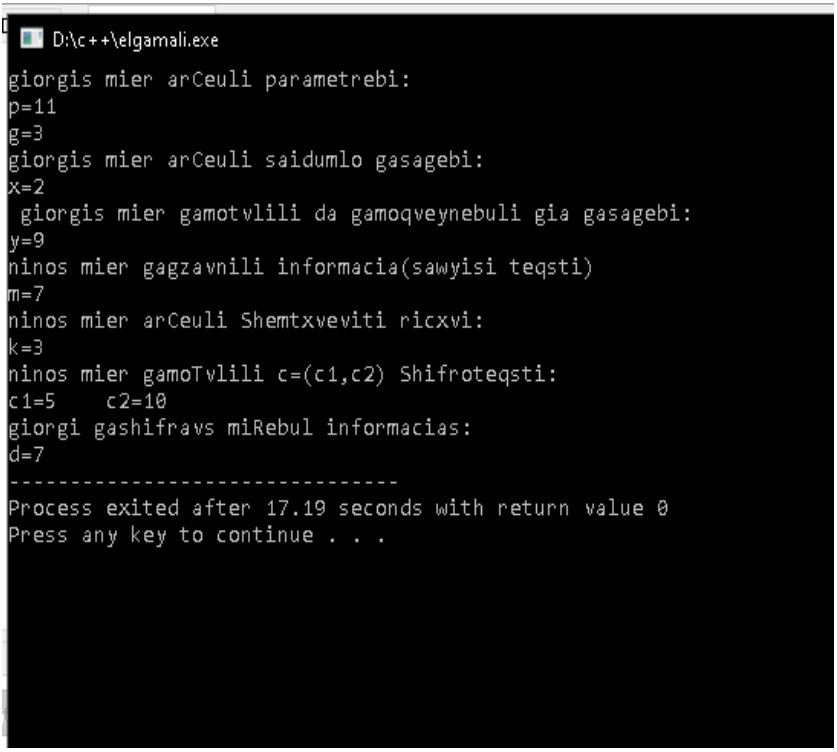
პროგრამული რეალიზაცია:

```
#include<iostream>
#include<stdlib.h>
#include<math.h>
#include<time.h>
using namespace std;
long long int  usg(long long int x, long long int y)
//უდიდესი საერთო გამყოფის  პოვნა
        {while(x!=y)
        if(x>y) x-=y;
        else y-=x;
                        return x;
                        }
         long long int  k_urtiertmartivi(long long int p1)
        {  long long int k;
                         bool f=true;
                        for(int i=2;i<p1 && f;i++ )
 //k-ს გამოთვლა
                                if(usg(i,p1)==1){
                                  f=false; k=i;}

                        return k;
                         }
```

```
        long long int gamotvla(long long int  a1, long long int x1, long long int p1)

        {   long long int t,t1;
                        t=pow(a1,x1);
                        t1=t%p1;
                        return t1;
        }
int main()
{
     long long int m,g,p,x,y,k,c1,c2,t,d1,d,c;
     cout<<"giorgis mier arCeuli parametrebi:"<<endl;
     cout<<"p="; cin>>p;
     cout<<"g="; cin>>g;
     cout<<"giorgis mier arCeuli saidumlo gasagebi:"<<endl;
     cout<<"x=";cin>>x;
     cout<<" giorgis mier gamotvlili da gamoqveynebuli gia gasagebi:"<<endl;
    y=gamotvla(g,x,p);
    cout<<"y="<<y<<endl;
    cout<<"ninos mier gagzavnili informacia(sawyisi teqsti)"<<endl;
    cout<<"m="; cin>>m;
```

```
 cout<<"ninos mier arCeuli Shemtxveviti ricxvi:"<<endl;
     k=k_urtiertmartivi(p-1);
     cout<<"k="<<k<<endl;
 cout<<"ninos mier gamoTvlili c=(c1,c2)
Shifroteqsti:"<<endl;
     c1=gamotvla(g,k,p);
     t=m*pow(y,k);
          c2=gamotvla(t,1,p);
          cout<<"c1="<<c1<<"    "<<"c2="<<c2<<endl;;
          cout<<"giorgi gashifravs miRebul
informacias:"<<endl;
         //long long int c11=gamotvla(c1,x,p);
          //cout<<c11<<endl;
          d=c2*gamotvla(c1,p-1-x,p)%p;
          cout<<"d="<<d;
}
```

შედეგი:



კრიპტოანალიზი:

ელგამალის სისტემას გააჩნია დაცვის გაცილებით მაღალი ხარისხი, ვიდრე RSA ალგორითმს ერთი და იგივე N-ის შემთხვევაში, რაც შიფრაციისა და დეშიფრაციის სიჩქარის საგრძნობ გაზრდას იწვევს. ელგამალის სისტემის კრიპტო მედეგობა ემყარება შემდეგ ფაქტს: საკმაოდ

მარტივად შეიძლება ნებისმიერი მთელი რიცხვის საჭირო ხარისხში ახარისხება. ამისათვის ეს რიცხვი შეგვიძლია გავამრავლოთ თავის თავზე იმდენჯერ, რამდენჯერაც დავჭირდება, და ეს ოპერაცია განხორციელდება ჩვეულებრივად რიცხვების გამრავლების წესებით. სირთულე კი მდგომარეობს იმაში, რომ ძალიან ძნელია იპოვო ის ხარისხის მაჩვენებელი, რომელშიც უნდა აახარისხო მოცემული პირველი რიცხვი, რომ ახარისხების შედეგად მიიღო მეორე მოცემული რიცხვი. ზოგადად, ეს დისკრეტული გალოგარითმების ამოცანა გაცილებით უფრო რთულია, ვიდრე დიდი რიცხვის მარტივ მამრავლებად დაშლის ამოცანა. აქედან შეგვიძლია დავასკვნათ, რომ RSA და ელგამალის სისტემების გატეხვა დაახლოებით ერთი და იგივე სირთულის ამოცანაა. პროგრამული და აპარატული რეალიზაციის შემთხვევაში რეალიზაციის სირთულე დაახლოებით ერთი და იგივეა. მაგრამ კრიპტომედეგობის თვალსაზრისით ეს ორი ალგორითმი საგრძნობლად განსხვავდება. თუ ჩვენ განვიხილავთ 512 ბიტიანი ნებისმიერი მთელი რიცხვის დაშლას მარტივ მამრავლებად და 512 ბიტიანი მთელი რიცხვების გალოგარითმების

ამოცანას, მათემატიკოსების აზრით, მეორე ამოცანა საგრძნობლად უფრო რთულია, ვიდრე პირველი.

აქვე უნდა აღინიშნოს ერთი საკითხი. თუ კრიპტოანალიტიკოსმა RSA ალგორითმით აგებულ სისტემაში შეძლო რომელიმე აბონენტის ღია N გასაღების დაშლა ორ მარტივ მამრავლად, მაშინ გატყდება მხოლოდ ამ აბონენტის მონაცემები. ხოლო ელგამალის ალგორითმით აგებული სისტემის შემთხვევაში ერთი აბონენტის მონაცემების გატეხვით კრიპტოგრაფიული ქსელის ყველა აბონენტის მონაცემებს ემუქრება გატეხვა. ცნობილია აგრეთვე, რომ მკვლევარებმა ლენსტრამ და მანასიმ შეძლეს RSA ალგორითმით აგებულ სისტემაზე შეტევის განხორციელება, მათ ფერმას მეცხრე რიცხვი მართივ მამრავლებად დაშალეს. გარდა ამისა, ამ კვლევარებმა ელგამალის ალგორითმის სისუსტეზეც მიუთითეს, რადგან ფერმას მეცხრე რიცხვის მარტივ მამრავლებად წაარმოდგენის მათი მეთოდი საკმაოდ ამარტივებს ზოგიერთი მარტივი რიცხვის დისკრეტული გალოგარითმების საკითხს. ანუ მარტივი *P* შეიძლება შერჩეული იყოს საგანგებოდ ისეთი, რომლისთვისაც დისკრეტული გალოგარითმების ამოცანა ჩვეულებრივი კომპიუტერის მეშვეობით

ადვილად გადაწდება. თუმცა ელგამალის ალგორითმისათვის ასეთი რიცხვების არსებობა ფატალური არ არის, მათი რაოდენობა ძალიან ცოტაა, და ალგორითმის მარტივად გატეხვის საშიშროება მოიხსნება, თუ ალგორითმს დავუმატებთ P მარტივი რიცხვის გარანტირებულად შემთხვევითად შერჩევის პროცედურას.

დავალება:

განახორციელეთ მონაცემების შიფრაცია - დეშიფრაცია , როდესაც

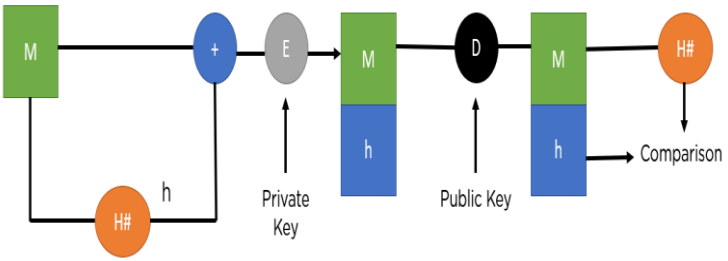
$$p = 11; g = 2; x = 4; k = 6;\ M = 5.$$

ციფრული ხელმოწერა (Digital Signature–DS)

ციფრული ხელმოწერა ერთდროულად გადაჭრის გადაცემული შეტყობინების მთლიანობის, შეტყობინების ავტორის აუთენტიფიკაციისა და ავტორობის ვერუარყოფის პრობლემებს. ციფრული ხელმოწერის დროს სპეციალური განაწესის საშუალებით გადაცემულ ინფორმაციას ემატება ჩანაწერი, რომელიც გამომუშავდება ინფორმაციის ავტორის გასაღების და გადასაცემი ინფორმაციის საშუალებით.

ციფრული ხელმოწერა საშუალებას იძლევა მოვახდინოთ გადაცემული შეტყობინების მთლიანობის და ავტორის აუთენტიფიკაცია. ამიტომ შეუძლებელი უნდა იყოს, რომ ვინმემ, ვისაც არა აქვს წვდომა ხელმოწერის საიდუმლო გასაღებთან, გააყალბოს ხელმოწერა. ამ თვისებას უწოდებენ **ხელმოწერის გაუყალბებლობის** თვისებას. ხელმოწერის ავტორმა ვერ უნდა შეძლოს უარყოს თავისი ხელმოწერა. ამ თვისებას უწოდებენ **ხელმოწერის ვერუარყოფის** თვისებას.
ციფრული ხელმოწერის შესასრულებლად საჭიროა ქსელში ჩართულ სუბიექტებს გააჩნდეთ ღია და საიდუმლო გასაღებების წყვილი, იცოდნენ სისტემის ღია პარამეტრები და ზუსტად შეასრულონ განაწესით გათვალისწინებული პროცედურები. განსხვავებით ღიაგასაღებიანი დაშიფვრის სისტემებისგან, ხელმოწერის სისტემებში

საიდუმლო გასაღები გამოიყენება შეტყობინების ხელმოსაწერად, ღია გასაღები კი - ხელმოწერის ნამდვილობის შესამოწმებლად (ხელმოწერის ვერიფიკაციისთვის).



ნახაზი- 19 DSA ალგორითმი

ციფრული ხელმოწერა ნიშნავს განსაკუთრებული პროცედურების საშუალებით ციფრული სახით წარმოდგენილი ინფორმაციის ჭეშმარიტობის დასაბუთებას. ანალოგიურად იმისა, როდესაც ქაღალდზე დაწერილი ინფორმაციის ჭეშმარიტებას ადასტურებენ ხელმოწერისა და ბეჭდის საშუალებით. აქედან გამომდინარე, შეგვიძლია ჩამოვთვალოთ ის კრიტერიუმები, რომლებსაც უნდა აკმაყოფილებდეს ციფული ხელმოწერის ალგორითმი:

- ციფრული ხელმოწერის საშუალებით ნებისმიერ პირს, რომელიც ფლობს ინფორმაციის ავტორის ღია გასაღებს, ცალსახად უნდა შეეძლოს გადმოცემული ინფორმაციის ავტორის დადგენა;

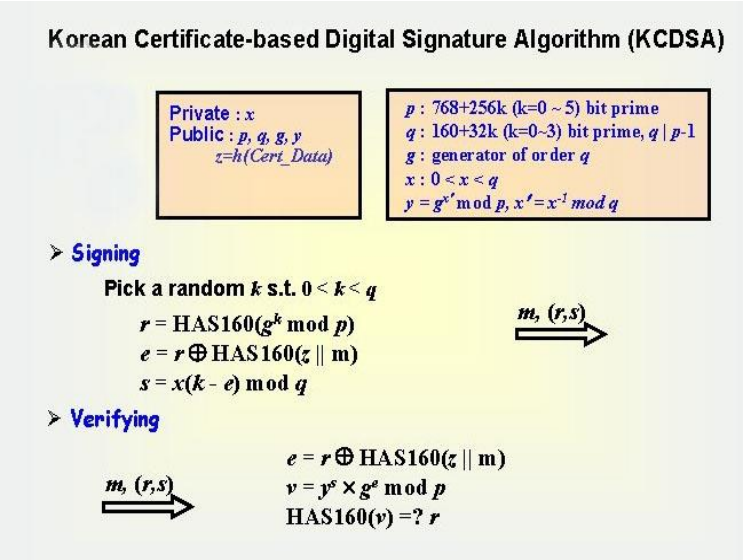
- შეუძლებელი უნდა იყოს ხელმოწერის შემდეგ დოკუმენტში ცვლილებების შეტანა;
- შეუძლებელი უნდა იყოს ხელმოწერის გაყალბება;
- ინფორმაციის ავტორს არ უნდა შეეძლოს უარყოს თავისი ხელმოწერა.

ციფრული ხელმოწერის ალგორითმების დანიშნულება არაა ინფორმაციის კონფი-დენციალურობის უზრუნველყოფა. ციფრული ხელმოწერა შეიძლება გაკეთდეს როგორც შიფროგრამაზე, ასევე ღია ტექსტზე. თუ ქვეყანაში მიღებულია კანონი ციფრული ხელმოწერის შესახებ, მაშინ ციფრულ ხელმოწერას ისეთივე ძალა აქვს, როგორიც ქაღალდზე გაკეთებულ ხელმოწერას. ციფრული ხელმოწერისათვის შეიძ-ლება გამოვიყენოთ RSA ალგორითმი, ოღონდ ამ შემთხვევაში ხელმოწერა კეთდება საიდუმლო გასაღებით და მოწმდება ღია გასაღების საშუალებით. ხელმოწერა შეიძლება გაკეთდეს როგორც ტექსტზე, ასევე მის ჰეშ-ფუნქციაზეც. ასეთ შემთხვევაში გამოითვლება ტექსტის ხელმოწერა და შემდეგ მასზე გაკეთდება ხელმოწერა. ამ მეთოდის სუსტი მხარეა ის, რომ თუ მოწინააღმდეგემ შეარჩია განსხვავებული ტექსტი, რომლის ჰეშ-ფუნქციაც ემთხვევა ხელმოწერილი ტექსტის ჰეშ-ფუნქციას, შეუძლია თქვენი ხელმოწერა გამოიყენოს საკუთარი მიზნე-ბისათვის. 1991 წლამდე, სანამ ამერიკის

სტანდარტების ნაციონალური ინსტიტუტი (NIST) მიიღებდა ახალ სტანდარტს ციფრული ხელმოწერის შესახებ, RSA გამოიყენებოდა როგორც ხელმოწერის დე-ფაქტო საერთაშორისო სტანდარტი.

ხელმოწერის ახალი სტანდარტი DSS (Digital Signatyre Standard) ეფუძნება DSA (Digital Signatyre Algorithm)-ს. ალგორითმში გამოყენებული იყო ცალმხრივ მიმართული ფუნქცია

$$f(x) = a^x(modp)$$



ნახაზი- 20 ციფრული ხელმოწერის KCDSA ალგორითმი

ალგორითმში გამოყენებულია შემდეგი პარამეტრები:

p - მარტივი რიცხვი, რომლის სიგრძე ბიტებში ტოლია L, $L = 64 \cdot k$, $512 \le L \le 1024$.

q - მარტივი რიცხვი, რომლის სიგრძე 160 ბიტი და $q \mid (p-1)$.

g - რიცხვი, რომელიც უნდა ავიყვანოთ ხარისხში, აკმაყოფილებს შემდეგ თვისებას:

$g \equiv h^{(p-1)/q} mod p$, სადაც h ნებისმიერი და (p–1)-ზე ნაკლები რიცხვია, ისეთი, რომ $h^{(p-1)/q} mod p > 1$.

x < q ნებისმიერი რიცხვია, რომელიც წარმოადგენს საიდუმლო გასაღებს.

$y = g^x (mod p)$ წარმოადგენს ღია გასაღებს. ალგორითმში გამოიყენება აგრეთვე ღია ტექსტის ჰეშ-ფუნქცია H(m), რომელიც გამოითვლება SHA-1 ალგორითმის საშუალებით.

p,q,g პარამეტრები შეიძლება საერთო იყოს მომხმარებელთა ჯგუფისათვის.

ხელმოწერის პროცედურა. იმისათვის, რომ ხელმომწერმა მოაწეროს ხელი შეტყობინებას, ის ასრულებს შემდეგ პროცედურებს:

1. დააგენერირებს შემთხვევით რიცხვს k < q .
2. გამოთვლის

$r = (g^k mod p) mod q$ და $s = (k^{-1}(H(m) + xr) mod q$.

თუ რომელიმე ამ რიცხვთაგანი იქნება ნოლის ტოლი, მაშინ მან უნდა გამოცვალოს k რიცხვი.

3. ხელმოწერის პარამეტრია სამასოცბიტიანი r || s, რომელიც მიეწერება ტექსტს ბოლოში აუცილებლად ამ მიმდევრობით.

ხელმოწერის შემოწმების პროცედურა. მიმღები მიიღებს რა ამ ხელმოწერას, მის შესამოწმებლად ატარებს შემდეგ პროცედურებას:

$$w = s^{-1} mod q$$

$$u_1 = (H(M) * w) mod q$$

1. გამოთვლის

$$u_2 = (rw) mod q$$

$$v = \left((g^{u_1} y^{u_2}) mod p\right) mod q$$

თუ რომელიმე ამ სიდიდეებისაგან მიიღო ნოლის ტოლი, ხელმოწერა არასწორია.

2. თუ r = v , მაშინ ხელმოწერა სწორია, წინააღმდეგ შემთხვევაში არა. ხელმოწერის და ხელმოწერის შემოწმების პროცედურები აბსოლუტურად განსხვავდება ერთმანეთისაგან, ამიტომ ასეთ ალგორითმებს ხშირად ასიმეტრიულ ალგორითმებს უწოდებენ.

ახლა უშუალოდ განვიხილოთ, როგორ ხდება RSA ალგორითმის გამოყენება ციფრული ხელმოწერისთვის. ვთქვათ A პიროვნებას ესაჭიროება B პიროვნებისთვის რაიმე m შეტყობინების გაგზავნა, რომელიც ელექტრონული ციფრული ხელმოწერითაა დადასტურებული.

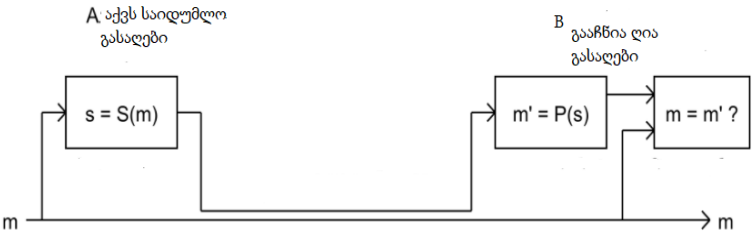
A პიროვნებას გააჩნია თავისი დახურული გასაღები. m ღია ტექსტისთვის იქმნება s ციფრული ხელმოწერა $\{d, n\}$ დახურული გასაღების მეშვეობით:

$$s = S_A(m) = m^d \ mod \ n$$

მის შემდეგ ხდება $\{m, s\}$ წყვილის გადაცემა, რომელიც შედგება შეტყობინებისა და ხელმოწერისაგან. B პიროვნება ღებულობს $\{m, s\}$ წყვილს, მას გააჩნია A პიროვნების ღია $\{e, n\}$ გასაღები. ხელმოწერის მეშვეობით ხდება შეტყობინების m' სახის გამოთვლა,

$$m' = P_A(s) = s^e \ mod \ n.$$

ბოლო ეტაპზე ხდება ხელმოწერის ნამდვილობის (და შეტყობინების უცვლელობის) შემოწმება, m და m'-ის შედარებით.



ნახაზი- 21 ხელმოწერის ნამდვილობის შემოწმების სქემა

რადგანაც ციფრული ხელმოწერა შეტყობინების ავტორის აუთენტიფიკაციას უზრუნველყოფს და ხელმოწერილი შეტყობინების სისწორეს ადასტურებს, ის ცხადია, იგივე დანიშნულების მატარებელია, რაც ჩვეულებრივი ხელმოწერა. შევნიშნოთ, რომ ციფრული ხელმოწერის შემოწმება შეუძლია ნებისმიერს, ვისაც გააჩნია ამ ხელმოწერის ავტორის ღია გასაღები. ხელმოწერის ნამდვილობის დადას-

ტურების მერე ამ ადამიანს შეუძლია გადასცეს შეტყობინება მესამე პირს, რომელსაც აგრეთვე შეუძლია ღია გასაღების მეშვეობით შეამოწმოს შეტყობინებისა და ხელმოწერის ნამდვილობა. ასეთი ხერხით, მაგალითად, შეიძლება ელექ-ტრონული ჩეკის გადაცემა და მერე მისი განაღდება ბანკში, რაც ძალიან მოსახერხებელია (ელექტრონული ხელმოწერის ნამდვილობის შემოწმება ბანკის თანამშრომელსაც შეუძლია). შევნიშნოთ ის ფაქტიც, ელექტრონული ხელმოწერის მქონე შეტყობინება არ არის დაშიფრული. ის გადაიგზავნება თავდაპირველი სახით და არ არის კონფიდენციალური. თუ ჩვენ შეტყობინების შიფრაციაც გვესაჭიროება, მაშინ ავტორმა ჯერ თავის შეტყობინებას უნდა დაამატოს ციფრული ხელმოწერა, ხოლო შემდეგ შეტყობინება ხელმოწერით უნდა დაშიფროს ღია გასაღებით, რომელიც ამ შეტყობინების მიმღებს ეკუთვნის. შეტყობინების მიმღები პირი ახდენს მის დეშიფრაციას დახურული გასაღების მეშვეობით.

კრიპტოგრაფია ელიფსურ წირებზე

ელიფსური კრიპტოგრაფია - კრიპტოგრაფიის ნაწილია, ის შეისწავლის ასიმეტრიულ კრიპტოსისტემებს, რომლებიც ეფუძნება სასრულ ველებზე განსაზღვრულ ელიფსურ წირებს. ელიფსური კრიპტოგრაფიის ძირითადი უპირატესობა იმაში მდგომარეობს, რომ დღესდღეობით უცნობია დისკრეტული გალოგარითმების სუბექსპონენციალური ალგორითმები. ელიფსური წირების ალგებრული თვისებების კრიპტოსისტემებისთვის გამოყენება ორმა მეცნიერმა ნილ კობლიცმა და ვიქტორ მილლერმა ერთმანეთის დამოუკიდებლად გადაწყვიტეს 1985 წელს. სწორედ ამ მომენტიდან დაიწყო კრიპტორაფიის ახალი მიმართულების სწრაფი განვითარება, მას უწოდეს „კრიპტორაფია ელიფსურ წირებზე“. ძირითად კრიპტოგრაფიულ ოპერაციად აქ სკალარული ნამრავლი გამოიყენება. განიხილება ელიფსური წირის წერტილის მოცემულ მთელ რიცხვზე სკალარული ნამრავლი, რომელიც განსაზღვრულია შეკრებისა და წირის წერტილების გაორმაგების საშუალებით. ეს ოპერაციები კი განისაზღვრება (შეკრების, გამრავლებისა და ინვერტირების ოპერაციებით) სასრულ ველებზე, სადაც განიხილება ეს ელიფსური წირები. კრიპტოგრაფიაში ელიფსური წირების გამოყენების განსაკუთრებულ ინტერესს

წარმოადგენს ამ მიმართულების უსადენო ტექნოლოგიებში გამოყენების უპირატესობები, კერძოდ, მაღალი სწრაფქმედება და შედარებით მოკლე სიგრძის გასაღებების გამოყენების შესაძლებლობა. ასიმეტრიული კრიპტოგრაფია ემყარება ზოგიერთი მათემატიკური პრობლემის გადაწყვეტის სირთულეს. პირველი ღია გასაღებიანი სისტემების, მაგალითად, RSA ალგორითმის კრიპტომედეგობა ემყარებოდა შედგენილი რიცხვის მარტივ მამრავლებად წარმოდგენის სირთულეს. ხოლო ელიფსურ წირებზე ალგორითმების გამოყენების დროს მთელი სირთულე გადატანილია იმ ფაქტზე, რომ არ არსებობს დისკრეტული გალოგარითმების ამოცანის გადაწყვეტის სუბექსპონენციალური ალგორითმები ელიფსური წირის წერტილთა ჯგუფებზე. ამოცანის სირთულეს განაპირობებს ელიფსური წირის ჯგუფის რიგი. მაგალითად, RSA ალგორითმთან შედარებით, იგივე რიგის კრიპტომედეგობის მისაღწევად 2005 წლის RSA-ს კონფერენციაზე *National Security Agency, NSA* სააგენტომ დაანონსა ალგორითმი სახელწოდებით «Suite B», რომელშიც გამოიყენეს ელიფსური წირე-ბის კრიპტოგრაფია „დაბალი“ რიგის ჯგუფებზე, 384 ბიტიანი გასაღებებით. ამით ინფორმაციის შენახვისა და გადაცემის ხარჯები საგრძნობლად იყო შემცირებული.

კრიპტოგრაფია ელიფსურ წირებზე ცნობილია, როგორც ECC აბრევიატურა, ხოლო ფართოდ გავრცელებული ალგორითმები - ECDH და ECDSA სახელებითაა ცნობილი და ღია გასაღებიან კრიპტოგრაფიულ ალგორითმებს წარმოადგენენ. კრიპტოსისტემები ელიფსურ წირებზე გამოიყენება TLS, PGP и SSH ტექნოლოგიებში, აგრეთვე, Bitcoin და სხვა კრიპტოვალუტებში.

მოვიყვანოთ ელიფსური წირის განსაზღვრება. ვეიერშტრასის მიერ ელიფსური წირი წარმოადგენს წერტილთა სიმრავლეს, რომელიც აკმაყოფილებს შემდეგ პირობას

$$y^2 = x^3 + ax + b$$

სადაც $4a^3 + 27b^2 \neq 0$.

E ელიფსური წირი K ველზე ეწოდება წერტილთა $\{(x, y) \cup O\}$, სადაც (x, y) წერტილები აკმაყოფილებენ განტოლებას:

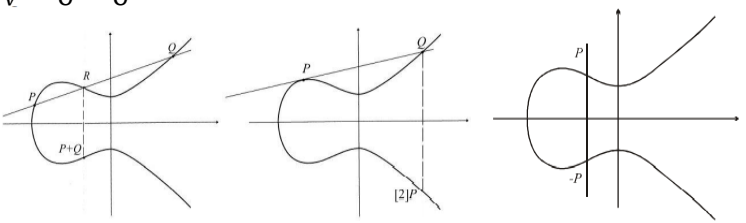
$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

O წერტილი კი არის „წერტილი უსასრულობაში“, რომელიც წარმოადენს ელიფსურ წირზე განსაზღვრული წერტილების შეკრების ოპერაციისთვის ერთეულოვან ელემენტს.

$a_1, a_2, a_3, a_4, a_6 \in K$ და განტოლების დეტერმინანტი $\Delta \neq 0$. პირობა $\Delta \neq 0$ წარმოადგენს ელიფსური წირის გლუვობის პირობას, რაც ნიშნავს, რომ ელიფსურ წირზე არ მოიძებნება ისეთი წერტილები, სადაც

ორივე კერძო წარმოებული ერთდროულად ნულის ტოლია.

ვთქვათ, მოცემულია E ელიფსური წირი K ველზე. არსებობს „ქორდისა და მხების“ წესი, რომელიც წირის ორი წერტილიდან მესამე წერტილის მიღების საშუალებას გვაძლევს. თუ ამ წესს განვსაზღვრავთ, როგორც შეკრების ოპერაციას E ელიფსური წირის წერტილებისთვის K ველზე, მივიღებთ ჯგუფს, რომელშიც ნულოვან ელემენტს O წერტილი წარმოადგენს. ამ ჯგუფს იყენებენ ელიფსურ კრიპტოსისტემებში. სურათზე მოცემულია ორი წერტილის შეკრების ოპერაციის გეომეტრიული ინტერპრეტაცია ელიფსურ წირებზე:



ა) $P+Q$ ბ) $P+P$ გ) $P+(-P)=O$

ნახაზი- 22 ოპერაციები ელიფსურ წირებზე.

სასრული ჯგუფი $F(q)$ ველზე. *კრიპტოგრაფიაში ელიფსური წირი განიხილება არა ნამდვილ რიცხვთა სიმრავლეზე, არამედ $F(q)$ სასრულ ველზე. ჩვენ უნდა ანვიხილოთ ის წერტილები, რომლებიც ერთდროულად ეკუთვნიან როგორც ელიფსურ წირს, ასევე სასრულ $F(q)$ ველსაც. აღვნიშნოთ ამ წერტილების სიმრავლე $E(F_q)$ -თი.*

შეიძლება ვაჩვენოთ, რომ ეს სიმრავლე მოთავსებულია

$$[E(F_q)+1-2\sqrt{q}, E(F_q)+1+q\sqrt{q}]$$

ინტერვალში, ანუ $|E(F_q)| = q+1-t$, სადაც $|t| \le 2\sqrt{q}$. t-ს ეწოდება E წირის კვალი $F(q)$ ველზე. (დამტკიცება აქ არ გვაქვს მოყვანილი). არსებობს პოლინომური ალგორითმი **SEA** (**Scoof-Elkies-Atkin**), რომელიც საშუალებს გვაძლევს, გამოვთვალოთ ნებისმიერი ელიფსური წირის წერტილები, რომლებიც ამავდროულად ეკუთვნიან $F(q)$ ველსაც.

განვიხილოთ ელიფსური წირების არითმეტიკა. თავისთავად ცხადია, რომ კომპიუტერული გამოთვლების დროს არანაირი გეომეტრიული ნახაზები არ დაგვჭირდება, გამოითვლების წარმოება შესაძლებელია ალგებრულად. ქვემოთ მოყვანილია ამ ოპერაციების ალგებრული აღწერა. ვთქვათ, განვიხილოთ კვლავ განტოლება

$$y^2 = x^3 + ax + b$$

ამ განტოლებას აუცილებლად უნდა გააჩნდეს სამი ფესვი, წინააღმდეგ შემთხვევაში ჩვენ ვერ შევძლებთ შეკრების ოპერაციების შესრულებას. ამისთვის საჭიროა, რომ განტოლების დისკრიმინანტი $\Delta = -16(4a^3 + 27b^2) \neq 0$. გარდა ამისა, რადგანაც $F(q)$ ველში საქმე გვაქვს მთელ რიცხვებთან, $a, b \in Z$.

1. ორი განსხვავებული რიცხვის შეკრება. მოცემულია $P(x_1, y_1)$ და $Q(x_2, y_2)$. უნდა მოვძებნოთ მესამე (x_3, y_3) წერტილის კოორდინატები.

როგორც ვნახეთ, გეომეტრიულად ამ წერტილების შესაკრებად უნდა გავატაროთ წრფე ამ ორ წერტილზე. დავუშვათ, წრფის განტოლებაა $y = ax + \beta$ (თუ წრფე არ იქნება OX ღერძის პერპენდიკულარული (როგორც ეს გვაქვს მესამე შემთხვევაში), მაშინ შეგვიძლია დავწეროთ, რომ

$$a = (y_2 - y_1)/(x_2 - x_1) \text{ და } \beta = y_1 - ax_1.$$

წრფის წერტილი $(x.ax + \beta)$ მაშინ და მხოლოდ მაშინ შეიძლება ეკუთვნოდეს E ელიფსურ წირს, თუ შესრულდება პირობა $(ax + \beta)^2 = x^3 + ax + b$, საიდანაც მივიღებთ, რომ $x^3 - (ax + \beta)^2 + ax + b = 0$ განტოლებას გააჩნია სამი განსხვავებული ამონახსნი, ე.ი. წრფე სამჯერ

კვეთს ელიფსურ წირს, ამასთან, ორი გადაკვეთა იქნება $P(x_1, y_1)$ და $Q(x_2, y_2)$ წერტილებში, ანუ x_1, x_2 წარმოადგენენ განტოლების ამონახსნებს. მაშინ, ვიეტის თეორემის თანახმად, $x_3 = a2 - x_1 - x_2$. მარტივი გამოთვლებით მივიღებთ, რომ

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \quad \text{და}$$

$$y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1 .$$

2. ერთი და იგივე წერტილის შეკრება. თუ $P = Q$, მაშინ როგორც ნახაზიდან ჩანს, გვექნება წირის მხები p წერტილში, თუ a არის არაცხადი ფუნქციის წარმოებული P წერტილში. მაშინ მივიღებთ, რომ

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \quad \text{და}$$

$$\left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1 \quad .$$

3. მოპირდაპირე წერტილების შეკრება. რაიმე ორი $P(x, y)$ და $P(x, -y)$ მოპირდაპირე წერტილების შეკრებისას მათი ჯამი იქნება უსასრულობის წერტილი (წერტილი, რომელსაც კოორდინატები არ გააჩნია).

განვიხილოთ ახლა **კრიპტოსისტემის აგების საკითხი ელიფსურ წირზე.** ელიფსური წირის წერტილების საშუალებით აიგება აბელის ადიციური ჯგუფი და დისკრეტული ლოგარითმის პრობლემა, რომელიც წარმოადგენდა ცალმხრივმიმართულ ფუნქციას აბელის მულტიპლიკატორულ ჯგუფში, ერთი ერთზე გადმოდის ამ ადიციურ ჯგუფში. აქ ეს საკითხი უფრო რთულ ამოცანაა, რადგანაც თუ მულტიპლიკაციურ ჯგუფში არსებობდა სუბექსპონენციალური ალგორითმები ლოგარითმის გამოსათვლელად, აქ ასეთი ალგორითმები არ არსებობს. ეს განაპირობებებულია იმ ფაქტით, რომ ამ სიმრავლეში არ მოიძებნება მარტივი რიცხვების მსგავსი ელემენტები.

ნებისმიერი სისტემა, რომელიც დაფუძნებულია დისკრეტული ლოგარითმის პრობლემის გადაწყვეტაზე, მარტივად შეიძლება გადავიტანოთ ელიფსურ წირზე (შევნიშნოთ, რომ წირის წერტილები აღნიშნულია დიდი ასოებით, $\boldsymbol{F_p}$ ველის ელემენტები კი - პატარა ასოებით). ძირითადი განსხვავება მდგომარეობს $\boldsymbol{y = g^x (mod\ p)}$ ოპერაციის $Y = x \cdot G(mod\ p)$ ოპერაციით შეცვლაში. წრფეზე მოქმედებების დროს მოდულს არ მიუთითებენ, თუმცა ყველა გამოთვლა ხდება მოდულით.

ნამრავლის - $n \cdot P$ მნიშვნელობების გამოთვლა. კრიპტოგრაფიაში ხშირ შემთხვევაში საქმე გვაქვს წერტილის თავისთავთან შეკრების ოპერაციასთან. ეს ოპერაცია აღინიშნება $n \cdot P$ სახით. წერტილის თავისთავთან რამდენჯერმე შეკრება მხოლოდ წესის გამოყენებით საკმაოდ გრძელი პროცედურაა (ისევე, როგორც F_p^* ჯგუფში რიცხვის თავისთავზე n-ჯერ გამრავლება ანუ ახარისხების ოპერაცია). ისევე როგორც F_p^* ჯგუფში გვქონდა განმეორებით კვადრატში აყვანის ალგორითმი, აქაც გვაქვს ანალოგიური პროცედურა, რომელიც საშუალებას გვაძლევს მარტივად გამოვთვალოთ $n \cdot P$-ს მნიშვნელობა. ოღონდ, რადგანაც ამ შემთხვევაში საქმე გვაქვს ადიციურ ჯგუფთან, ალგორითმს ***გაორმაგებისა და შეკრების ალგორითმს უწოდებენ.*** ალგორითმის ძირითადი იდეა მდგომარეობს იმაში, რომ n მამრავლი წარმოადგინონ ორობით ფორმაში:

$$n = n_0 + n_1 \cdot 2 = n_2 \cdot 2^2 + ... + n_r \cdot 2^r$$

სადაც $n_0, n_1, ..., n_{r-1} \in \{0,1\}, n_r = 1$. ამის შემდეგ გამოითვლება $Q_0 = P,\ Q_r = 2 \cdot Q_{r-1}$ ... და გამოითვლება $n \cdot P = n_0 \cdot Q_0 + n_1 \cdot Q_1 + ... + n_r \cdot Q_r$. მაშასადამე, ალგორითმს აქვს შემდეგი სახე:

შესასვლელი: წერტილი $P \in E(F_p)$ და მთელი რიცხვი n

გამოსასვლელი: წერტილი $R = n \cdot P$

ბიჯი 1: $Q \leftarrow P; R \leftarrow O$

ბიჯი 2: სანამ $n > 0$ **შეასრულე:**

ბიჯი 3: თუ $n \equiv 1 (\mathrm{mod}\ 2)$ **მაშინ** $R \leftarrow R + Q$

$Q \leftarrow 2Q$ და $n \leftarrow \lfloor n/2 \rfloor$

ბიჯი 4: თუ $n > 0$ **გადადი ბიჯი 2-ზე.**

გამოიტანე წერტილი $R = n \cdot P$

განვიხილოთ მარტივი მაგალითი. მოცემული გვაქვს $E(F_{11}): y^2 = x^3 + x + 6$ ელიფსური წირი. ამ წირის ერთ-ერთი ელემენტია $P(2,7)$. გამოვთვალოთ $7 \cdot P$ სიდიდის მნიშვნელობა. პირველ რიგში წარმოვადგინოთ რიცხვი 7 ორობით ფორმაში. გვექნება

$$7 = 1 + 1 \cdot 2 + 1 \cdot 2^2$$

მაშინ გვექნება $Q_0 = P$, $Q_1 = 2 \cdot Q_0 = 2 \cdot P$, და $Q_2 = 2 \cdot Q_1 = 2 \cdot 2 \cdot P = 4 \cdot P$.

გამოვთვალოთ, $2P = (5,2);\ 4P = (10,2)$. შევკრიბოთ ახლა ეს წერტილები და მივიღებთ, რომ $7 \cdot P = (7,2)$. შევნიშნოთ, რომ ამ ალგორითმში შესაძლებელია გამოკლების ოპერაციის გამოყენებაც, რაც კიდევ უფრო ამცირებს ოპერაციათა რაოდენობას.

დისკრეტული გალოგარითმების პრობლემა შეიძლება განისაზღვროს ნებისმიერ ჯგუფში, და ამ მიზნით ნებისმიერი აბელის ჯგუფი შეიძლება გამოვიყენოთ. ეს ჯგუფი უნდა აკმაყოფილებდეს შემდეგ პირობებს:

- ჯგუფისთვის არ არსებობდეს რიცხვის ფაქტორიზაციის და დისკრეტული

ლოგარითმის გამოთვლის პოლინომური და სუბექსპონენციალური ალგორითმები;

- ადვილად იყოს შესაძლებელი ჯგუფური ოპერაციის რეალიზაცია;
- ჯგუფის აგება უნდა იყოს ადვილი და ელემენტების რაოდენობა იყოს დიდი.

ელიფსურ წირებზე აგებული ჯგუფები აკმაყოფილებენ როგორც პირველ, ასევე მესამე პირობებს. არსებობს პოლინომური ალგორითმები, რომელთა საშუალებითაც შესაძლებელია როგორც ჯგუფის რიგის დადგენა, ასევე მისი ელემენტების გამოთვლა.

ამ ჯგუფებში არ არსებობენ მარტივი რიცხვის ან დაუყვანადი პოლინომის ანალოგიური ელემენტები, ამიტომ შესაბამისად, არ არსებობენ დისკრეტული ლოგარითმის გამოთვლის სუბ-ექსპონენციალური ალგორიმებიც. თითოეული სასრული ველისთვის არსებობს საკმარისად ბევრი ელიფსური წირი და მათ მიერ შექმნილი ჯგუფები.

რაც შეეხება მეორე პირობას, მარტივი შესასრულებელი იქნება თუ არა ჯგუფური ოპერაცია, დამოკიდებულია სასრულ ველზე. თუ ველის მახასიათებელი იქნება დიდი მარტივი რიცხვი, ჯგუფური ოპერაციის შესრულება (განსაკუთრებით მისი აპარატურული რეალიზაციის დროს) იქნება ისეთივე რთული, როგორც ეს არის Z_p^* ჯგუფში. მაგრამ თუ სასრული

ველის მახასიათებელი იქნება ორის ტოლი, მაშინ ჯგუფური ოპერაციის შესრულება შედარებით მარტივია.

აუთენტიფიკაციის ამოცანა RSA ალგორითმის გამოყენებით

ცნობილია, რომ RSA ალგირითმი გამოიყენება არა მარტო შეტყობინებების შიფრაციისთვის, არამედ ციფრული ხელმოწერისთვის. ასეთი ციფრული ელექტრონული ხელმოწერა გვაძლევს საშუალებას, რომ დავადასტუროთ ელექტრონული დოკუმენტის ავტორობა. ციფრული ხელმოწერა შეიძლება გამოვიყენოთ როგორც რეალური პიროვნების იდენტიფიკაციისთვის, ასევე კრიპტოსავალუტო სისტემის აქაუნთის დადასტურებისას. ელექტრონული ხელმოწერა კრიპტოგრაფიული მეთოდების მეშვეობით დაკავშირებულია როგორც ავტორთან, ასევე თვით დოკუმენტთან და ასეთი ხელმოწერის გაყალბება შეუძლებელია ჩვეულებრივი ასლების (კოპირების) შექმნის მეთოდით.

როგორც ჩვენთვის უკვე ცნობილია, RSA ალგორითმში, რომელიც ძირითადად გამოიყენება ინფორმაციის შიფრაციის მიზნით, ღია გასაღებით ხდება ინფორმაციის დაშიფრვა, ხოლო საიდუმლო გასაღებით კი დეშიფრაცია. ციფრული ხელმოწერის ალგორითმებში კი - პირიქით მოქმედებებს ვასრულებთ: საიდუმლო გასაღებით ხდება ტექსტის ხელმოწერა, ხოლო ღია გასაღებით კი ხელმოწერის შემოწმება. როგორც ვიცით, ღია გასაღები ნიშნავს, რომ ეს გასაღები ცნობილია სისტემის ყველა მომხმარებლისთვის და ყველას

შეუძლია ისარგებლოს ამ გასაღებით. ასეთი სისტემების დიდ უპირატესობას სიმეტრიულ ერთ გასაღებიან სისტემებთან შედარებით წარმოადგენს ის, რომ აღარ გვჭირდება დახურული არხი (საიდუმლო გასაღების გადასაცემად), რაც ზოგადად სირთულეებთან არის დაკავშირებული.

ახლა ერთმანეთისაგან განვასხვავოთ იდენტიფიკაციისა და აუთენტიფიკაციის ცნებები. იდენტიფიკაციის ქვეშ იგულისხმება რაიმე სისტემის მომხმარებლისთვის ისეთი უნიკალური ატრიბუტის მინიჭება, რომლის საშუალებითაც შესაძლებელი იქნება სხვა მომხმარებლებისგან მისი გამორჩევა. სწორედ ასეთ ატრიბუტს იდენტიფიკატორი ეწოდება. ახლა განვიხილოთ აუთენტიფიკაციის ცნება. ის წარმოადგენს მომხმარებლის მიერ წარმოდგენილი იდენტიფიკატორის შემოწმებას და ამ იდენტიფიკატორის ნამდვილობის დადას-ტურებას. აქედან ვასკვნით, რომ იდენტიფიკატორი საჭიროა იმისთვის, რომ გავიგოთ მომხმარებლის ვინაობა, ხოლო აუთენტიფიკაცია კი - დავრწმუნ-დეთ, რომ მომხმარებელი ნამდვილად ისაა, ვისი სახელითაც შემოდის სისტემაში.

იდენტიფიკაციისა და აუთენტიფიკაციის პროცესი ცხადია, ერთმანეთთან მჭიდრო კავშირშია. სხვათა შორის, ეს სქემა უნდა ითვალისწინებდეს როგორც მომხმარებლის, ასევე ოპერატორის შესაძლო შეცდომებს. ამიტომ თუ

შესვლათა მცდელობების რაოდენობა არ აღემატება წინასწარ დათქმულ რაოდენობას, მომხმარებელს შეუძლია ხელახლა სცადოს სისტემაში შესვლა. თუ შესვლათა რაოდენობა გადააჭარბებს წინასწარ ცნობილ დასაშვებ რაოდენობას, სისტემა (უმეტესად დროებით) ბლოკავს მომხმარებელს და ამავე დროს სისტემის ადმინისტრატორს აფრთხილებს არასანქცირებული შესვლის მცდელობის შესახებ.

ახლა მოკლედ შევეხოთ აუთენტიფიკაციის მეთოდებსაც. აუტენტიფიკაციისათვის ძირითადად შემდეგი მეთოდები გამოიყენება: მეთოდები, რომლებიც ეფუძნება რაიმე საიდუმლო ინფორმაციის ცოდნაზე. ასეთი მეთოდების კლასიკურ მაგალითს წარმოადგენს პაროლების სისტემა, როდესაც აუთენტიფიკაციისთვის სიმბოლოთა რაიმე მიმდევრობა გამოიყენება, რომელიც, მომხმარებელმა, ცხადია, სისტემაში უნდა შეიყვანოს. შესაძლებელია, აუთენტიფიკაციის მეთოდი დაფუძნებული რაიმე ინფორმაციაზე, რომელიც ასოცირდება მომხმარებელთან, მაგალითად, მომხმარებლის ბიოგრაფიის რაიმე ფაქტი, ან მისი ადგილსამყოფელი, კოორდინატები.

შესაძლებელია ერთდროულად რამდენიმე მეთოდის გამოყენებაც, როცა სისტემაში შენახული ან დამუშავებული ინფორმაცია განსაკუთრებით მნიშვნელოვანია. პაროლური სისტემა ყველაზე

ფართოდ გავრცელებულ მეთოდს წარმოადგენს, რაც განპირობებულია იმით, რომ ის არ მოითხოვს დამატებით დანახარჯებს და მომხმარებლისთვის მოსახერხებელია, მაგალითად, თვალის ბადურის სკანირების მეთოდისგან განსხვავებით. აქვე უნდა აღინიშნოს შემდეგი ფაქტი: რაც უფრო საიმედოა პაროლი, მით უფრო ძნელი გამოსაყენებელია ის მომხმარებლისთვის. მართლაც ყველაზე საიმედო იქნება პაროლი, რომელიც რაიმე სიმბოლოების შემთხვევით მიმდევრობას წარმოადგენს, მაგრამ ასეთი პაროლის დამახსოვრება ადამიანისთვის პრაქტიკულად შეუძლებელია. თუ მომხმარებელი სადმე ჩაიწერს ასეთ პაროლს, ეს პაროლის გატეხვის მიზეზი შეიძლება გახდეს. ამიტომაც მომხმარებელი ცდილოს შექმნას ადვილად დამახსოვრებადი პაროლი. ეს კი უადვილებს მოწინააღმდეგეს პაროლის გატეხვას. მოწი-ნააღმდეგე პაროლური სისტემის გასატეხად აანალიზებს ადამიანის სისუსტეებს: ეს კი გულისხმობს თვალყურის დევნებას, მოსმენას, შანტაჟს, მუქარას და ა.შ. მოწინააღმდეგემ შეიძლება საგანგებოდ გამოიყენოს შერჩევის მეთოდები, როგორიცაა სრული გადარჩევის მეთოდი; შერჩევა ლექსიკონის დახმარებით (არსებობს ყველაზე ხშირად გამოყენებული პაროლების ლექსიკონები, რომელთა გამოყენებას მნიშვნელოვნად ამცირებს სრულ გადარჩევას); შესაძლებელია მომხმარებლის შესახებ

ინფორმაციის გამოყენებით შერჩევა, მაგალითად, ბოროტმოქმედმა შეიძლება გამოიყენოს დაბადების დღის თარიღი, შვილების, მეუღლის სახელები და ა.შ... გატეხვის დროს შესაძლებელია იდენტიფიკაციის პაროლური სისტემის რეალიზაციის დროს დაშვებული შეცდომების გამოყენებაც (ან თუნდაც იდენტიფიკაციის ცალკეული კომპონენტების რეალიზაციის დროს დაშვებული შეცდომების გამოყენება).

ახლა უშუალოდ განვიხილოთ, როგორ ხდება RSA ალგორითმის გამოყენება ციფრული ხელმოწერისთვის. ვთქვათ A პიროვნებას ესაჭიროება B პიროვნებისთვის რაიმე m შეტყობინების გაგზავნა, რომელიც ელექტრონული ციფრული ხელმოწერითაა დადასტურებული.

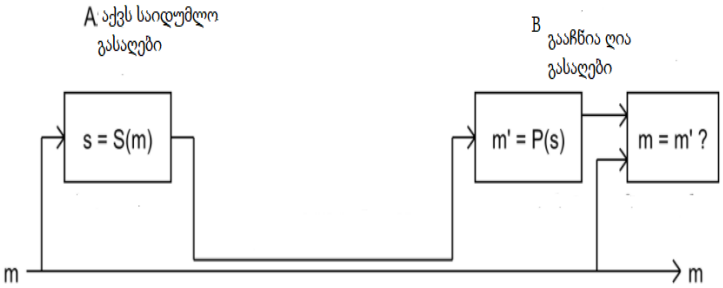
A პიროვნებას გააჩნია თავისი დახურული გასაღები. m ღია ტექსტისთვის იქმნება s ციფრული ხელმოწერა $\{d, n\}$ დახურული გასაღების მეშვეობით:

$$s = S_A(m) = m^d \ mod \ n$$

მის შემდეგ ხდება $\{m, s\}$ წყვილის გადაცემა, რომელიც შედგება შეტყობინებისა და ხელმოწერისაგან. B პიროვნება ღებულობს $\{m, s\}$ წყვილს, მას გააჩნია A პიროვნების ღია $\{e, n\}$ გასაღები. ხელმოწერის მეშვეობით ხდება შეტყობინების m' სახის გამოთვლა,

$$m' = P_A(s) = s^e \ mod \ n.$$

ბოლო ეტაპზე ხდება ხელმოწერის ნამდვილობის (და შეტყობინების უცვლელობის) შემოწმება, m და m'-ის შედარებით.

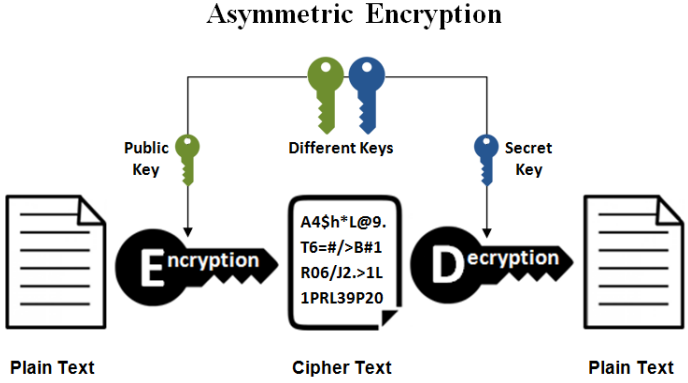


ნახაზი- 23 შეტყობინების უცვლელობის შემოწმება

რადგანაც ციფრული ხელმოწერა შეტყობინების ავტორის აუთენტიფიკაციას უზრუნველყოფს და ხელმოწერილი შეტყობინების სისწორეს ადასტურებს, ის ცხადია, იგივე დანიშნულების მატარებელია, რაც ჩვეულებრივი ხელმოწერა. შევნიშნოთ, რომ ციფრული ხელმოწერის შემოწმება შეუძლია ნებისმიერს, ვისაც გააჩნია ამ ხელმოწერის ავტორის ღია გასაღები. ხელმოწერის ნამდვილობის დადასტურების მერე ამ ადამიანს შეუძლია გადასცეს შეტყობინება მესამე პირს, რომელსაც აგრეთვე შეუძლია ღია გასაღების მეშვეობით შეამოწმოს შეტყობინებისა და ხელმოწერის ნამდვილობა. ასეთი ხერხით, მაგალითად, შეიძლება ელექტრონული ჩეკის გადაცემა და მერე მისი განაღდება ბანკში, რაც

ძალიან მოსახერხებელია (ელექტრონული ხელმოწერის ნამდვილობის შემოწმება ბანკის თანამშრომელსაც შეუძლია). შევნიშნოთ ის ფაქტიც, ელექტრონული ხელმოწერის მქონე შეტყობინება არ არის დაშიფრული. ის გადაიგზავნება თავდაპირველი სახით და არ არის კონფიდენციალური. თუ ჩვენ შეტყობინების შიფრაციაც გვესაჭიროება, მაშინ ავტორმა ჯერ თავის შეტყობინებას უნდა დაამატოს ციფრული ხელმოწერა, ხოლო შემდეგ შეტყობინება ხელმოწერით უნდა დაშიფროს ღია გასაღებით, რომელიც ამ შეტყობინების მიმღებს ეკუთვნის. შეტყობინების მიმღები პირი ახდენს მის დეშიფრაციას დახურული გასაღების მეშვეობით.

განვიხილოთ აუტენთიფიკაციის ამოცანა RSA ალგორითმის გამოყენებით. ამ მაგალითში ხდება ავტორების დაფიქსირება სახელდებული რიცხვების გაცვლით.



ნახაზი- 24 ასიმეტრიული კრიპტოსისტემის მუშაობის პრინციპი

X მხარე ირჩევს K_1 რიცხვს, ხოლო Y მხარე ირჩევს K_2 რიცხვს. მათ შორის ამ რიცხვების გაცვლა უნდა შესრულდეს ისე, რომ მოწინააღმდეგემ არაფერი არ უნდა შეიტყოს.

X მხარის საწყისი მონაცემებია:

$$N_1 = 15, (p_1 = 3, q_1 = 5);\ \varphi(N_1) = 8;$$
$$e_1 = 3, d_1 = 3, K_1 = 3,$$

Y მხარის საწყისი მონაცემებია:

$$N_2 = 21, (p_2 = 3, q_2 = 7);\ \varphi(N_2) = 12,$$
$$e_2 = 5, d_2 = 5, K_2 = 7,$$

X მხარე დაშიფრავს K_1 -ს და გაუგზავნის Y მხარეს, m_1 -ს :

$$K_1^{d_1} mod\ N_1 = 3^3 mod\ 15 \equiv 12,$$
$$m_1' = 12.$$
$$(m_1')^{e_2} mod\ N_2 = 12^5 mod\ 21 \equiv 248832\ mod\ 21$$
$$\equiv 3\ mod\ 21;$$
$$m_1 = 3.$$

Y მხარე გაშიფრავს მიღებულ m_1 ინფორმაციას:

$$m_1{}^{d_2} mod N_2 = 3^5 mod 21 \equiv 243\ mod\ 21$$
$$\equiv 12\ mod\ 21;$$
$$m_1' = 12.$$
$$(m_1')^{e_1} mod\ N_1 = 12^3 mod\ 15 \equiv 1728\ mod\ 15$$
$$\equiv 3\ mod\ 15;$$
$$K_1 = 3.$$

ანალოგიურად მოხდება Y მხარე K_2 ინფორმაციის დაშიფრვა და X მხარისათვის გადმოგზავნა:

$$K_2^{d_2} mod\ N_2 = 7^5 mod\ 21 \equiv 16807\ mod\ 21$$
$$\equiv 7\ mod\ 21,$$
$$m_2' = 7.$$

$$(m_2')^{e_1} mod\ N_1 = 7^3 mod\ 15 \equiv 343\ mod\ 15$$
$$\equiv 13\ mod\ 15;$$
$$m_2 = 13.$$

X მხარე გაშიფრავს მიღებულ m_2 ინფორმაციას:

$$m_2{}^{d_1} mod\ N_1 = 13^3 mod 15\ \equiv 2197\ mod\ 15$$
$$\equiv 7\ mod\ 15;$$
$$m_2' = 7.$$
$$(m_2')^{e_2} mod\ N_2 = 7^5 mod\ 21 \equiv 16807\ mod\ 21$$
$$\equiv 7\ mod\ 21;$$
$$K_2 = 7.$$

პროგრამული რეალიზაცია:

```
#include<iostream>
#include<stdlib.h>
#include<math.h>
#include<time.h>

using namespace std;

        long long int  usg(long long int x, long long int y)
//უდიდესი საერთო გამყოფის პოვნა
          {while(x!=y)
          if(x>y) x-=y;
          else y-=x;
                              return x;
                              }
           long long e_urtiertmartivi (long long int f1)
           {
           long long int e;
```

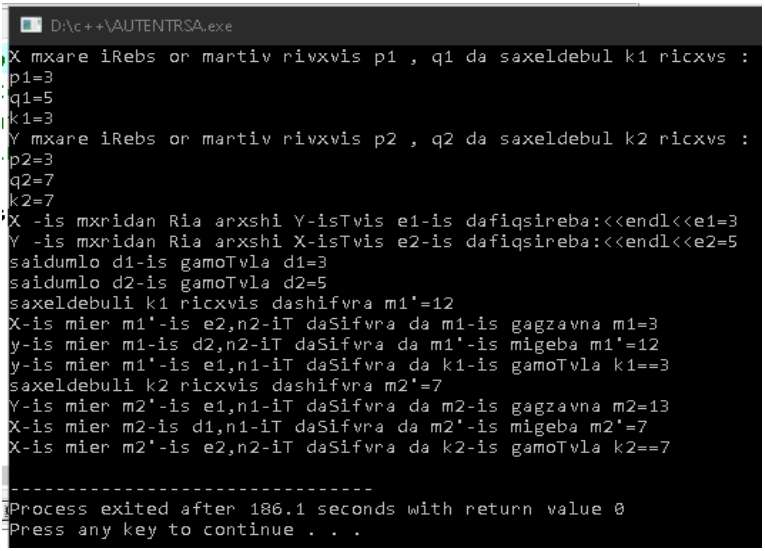
```
                        bool f=true;
                       for(int i=2;i<f1 && f;i++ ) //ღია
//გასაღების e სიდიდის არჩევა
                                if(usg(i,f1)==1){
                                  f=false; e=i;}

                       return e;
                        }
                        int d_mod(long long int a, long
long int b)
{
  long long int q, r, x1, x2, y1, y2,x,y,d;
  if (b == 0) {
    d=a; x=1; y = 0;
    return x;
  }
  x2=1; x1=0; y2=0; y1=1;
  while (b > 0) {
    q=a/b; r=a-q*b;
    x=x2-q*x1; y=y2-q*y1;
    a=b; b=r;
    x2=x1; x1=x; y2=y1; y1=y;
  }
  d = a; x = x2; y = y2;
 if (d == 1) return x;
}
```

```
long long int gamotvla(long long int  a1,
long long int x1, long long int p1)

{

long long int t; //დაშიფვრისა და
//გაშიფვრის ფუნქცია
if(x1==1)
    return a1;
t=gamotvla(a1,x1/2,p1);
if(x1%2==0)
    return (t*t)%p1;
else
    return (((t*t)%p1)*a1)%p1;
}

int main()
{

    long long int
p1,q1,n1,d1,e1,k1,m1,m11,f1,p2,q2,n2,d2,e2,k2,m2,m2
1,f2;
    cout<<"X mxare iRebs or martiv rivxvis p1 , q1 da
saxeldebul k1 ricxvs :"<<endl;
    cout<<"p1="; cin>>p1;cout<<"q1="; cin>>q1;
cout<<"k1="; cin>>k1;
    cout<<"Y mxare iRebs or martiv rivxvis p2 , q2 da
saxeldebul k2 ricxvs :"<<endl;
```

```
      cout<<"p2="; cin>>p2;cout<<"q2="; cin>>q2;
cout<<"k2="; cin>>k2;
       n1=p1*q1; n2=p2*q2;
       f1=(p1-1)*(q1-1); f2=(p2-1)*(q2-1);
      e1=e_urtiertmartivi(f1);
      cout<<"X -is mxridan Ria arxshi Y-isTvis e1-is
dafiqsireba:<<endl<<e1="<<e1<<endl;
       e2=e_urtiertmartivi(f2);
      cout<<"Y -is mxridan Ria arxshi X-isTvis e2-is
dafiqsireba:<<endl<<e2="<<e2<<endl;
      d1=d_mod(e1,f1);
      cout<<"saidumlo d1-is gamoTvla d1="<<d1<<endl;
      d2=d_mod(e2,f2);
      cout<<"saidumlo d2-is gamoTvla d2="<<d2<<endl;
       m11=gamotvla(k1,d1,n1);
      cout<<"saxeldebuli k1 ricxvis dashifvra
m1'="<<m11<<endl;
      m1=gamotvla(m11,e2,n2);
      cout<<"X-is mier m1'-is e2,n2-iT daSifvra da m1-is
gagzavna m1="<<m1<<endl;
      m11=gamotvla(m1,d2,n2);
      cout<<"y-is mier m1-is d2,n2-iT daSifvra da m1'-is
migeba m1'="<<m11<<endl;
      long long int k11=gamotvla(m11,e1,n1);
      cout<<"y-is mier m1'-is e1,n1-iT daSifvra da k1-is
gamoTvla k1=="<<k11<<endl;

            m21=gamotvla(k2,d2,n2);
```

```
    cout<<"saxeldebuli k2 ricxvis dashifvra
m2'="<<m21<<endl;
    m2=gamotvla(m21,e1,n1);
    cout<<"Y-is mier m2'-is e1,n1-iT daSifvra da m2-is
gagzavna m2="<<m2<<endl;
    m21=gamotvla(m2,d1,n1);
    cout<<"X-is mier m2-is d1,n1-iT daSifvra da m2'-is
migeba m2'="<<m21<<endl;
    long long int k21=gamotvla(m21,e2,n2);
    cout<<"X-is mier m2'-is e2,n2-iT daSifvra da k2-is
gamoTvla k2=="<<k21<<endl;
}
```

შედეგი:



დავალება:

განახორციელეთ მონაცემების შიფრაცია - დეშიფრაცია (აუტენტიფიკაციის ამოცანა და მისი გადაწყვეტა RSA ალგორითმის გამოყენებით), როდესაც

$$p_1 = 7; q_1 = 17;\ K_1 = 3, M_1 = 19;$$
$$p_2 = 3; q_2 = 11;\ K_2 = 7, M_2 = 5.$$

გამოყენებული ლიტერატურა

1. J.Buchmann, Introduction to Cruptography Springer, 2004
2. Shneier B. Applied Cryptography. John Wiley and Sons. Inc. New York. 1996.
3. Б.Шнайер, Прикладная криптография Москва Изд. “ТРИУМФ” –2003г
4. ქოჩლაძე ზ., თანამედროვე კრიპტოგრაფიის საფუძვლები. თბილისის უნივერსიტეტის გამომცემლობა, 2013
5. მეგრელიშვილი რ. ინფორმაციის დაცვის სისტემები. თბილისის სახელმწიფო უნივერსიტეტის გამომცემლობა. 2007.
6. https://habr.com/ru/post/534620/?fbclid=IwAR25NXADyLdZIbYJGTKWHu_jKvK_nMJVleHvjKytLiHRFFa4bHoq1RDwpDs
7. გულნარა კოტრიკაძე, ინფორმაციის დაცვა კომპიუტერულ სისტემებში. (დისერტაცია), თბილისი, საქართველოს ტექნიკური უნივერსიტეტი, 2008.
8. ლალი ბესელია, გენეტიკური ალგორითმების გამოყენება კრიპტოანალიზში. (დისერტაცია), თბილისი, სოხუმის სახელმწიფო უნივერსიტეტი, 2016.
9. სოფია შენგელია, გასაღების ღია არით გაცვლის მატრიცული ალგორითმისა და ფრაქტალური სტრუქტურების გამოკვლევა და სინთეზი. (დისერტაცია), თბილისი, სოხუმის სახელმწიფო უნივერსიტეტი, 2015.
10. მალხაზ ჭელიძე, კრიპტოსისტემისა და ციფრული ხელმოწერების ალგორითმების სინთეზისათვის. (დისერტაცია), თბილისი, საქართვე-

ლოს ტექნიკური უნივერსიტეტი, 2008. https://gtu.ge/Disertacia/chelidze_malxazi.pdf
11. Vernam G. S. Cipher printing telegraph systems for secret wire and radio telegraphic communication, J. Am. Inst. Electr. Eng. 45(1926), 109-115.
12. Diffie W. and Hellman M. E. New directions in cryptography, IEEE Trans. Inform. Theory, vd, It – 22, pp. 644-654 Nov. 1976.
13. Shannon C.E. A mathematical theory of communication. Bell System Tech J. 27, 1948, n.3, pp. 379-423. 1948. п.4, pp. 623-656. Русский пер. Шеннон К. Работы по теории информации и кибернетике. М., Ил, 1963.
14. Tuher Elgamal. A public Key cryptosystem and signature scheme based on discrete logarithms in G. R. Klukley and David Chaum Fditors. Advences in cryptology: Proseeding of crypto 84, volume 196 of lecture Notes in Computer Sience, pp.10-18, 19-22, Aug. 1984, Sgringer-Verlag 1985.
15. Анин Б. Зашита компьютерной информации. Москва. 2000.
16. Молдовян А. А., Молдовян Н. А., Гуц Н.Д., Изотов Б. В. Криптография, Скоростные шифры. Санкт-Петербург. 2002.
17. Wiener W. The Human Use of Human Beings, Cybernetics and Society. Houghton Mifflin Co., Boston, 1948. ქართული თარგმანი: ვინერი ვ. კიბერნეტიკა და საზოგადოება. თბილისი, 1958.
18. S. C. Pohlig and M. E. Hellman, AN improved algorithm for computing logarithms in GF(p) and its

cryptographic significance, IEEE Trans. Inform. Theory, vol. IT-24, pp.106-111, Jan. 1978.
19. Merkle R.C., Hellman M.E. Hidding information and signatures in trapdoor Knapsak, IEEE Trans. Inform. Theory, IT-24 (1978), pp 530-535
20. Rivest R.L. Shamir A. and Adelman L. On digital signatures and public key cryptosystems. Commun. ACM, vol, 21, no. 2, pp. 120-126. Feb1978.
21. Rivest R.L. Shamir A. and Adelman L., A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. https://people.csail.mit.edu/rivest/Rsapaper.pdf