ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი

ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი

კომპიუტერულ მეცნიერებათა განყოფილება

ალექსანდრე მელაძე

სამაგისტრო ნაშრომი:

ლოკალური ქსელის უსაფრთხოება

ნაშრომი შესრულებულია კომპიუტერული მეცნიერებათა მაგისტრის აკადემიური ხარისხის მოსაპოვებლად.

ხელმძღვანელი:ასოცირებული პროფესორი ზურაბ ქოჩლაძე

თბილისი

2013

ანოტაცია ---------- 2

Annotation ---------- 3

შესავალი ---------- 4

სრული ტექსტი ---------- 5-33

დასკვნა ---------- 34

გამოყენებული ლიტერატურა ---------- 35

ანოტაცია

განხილულია კომპიუტერულ სისტემების საიმედოობისა და უსაფრთხოების საკითხები. განსაკუთრებული ყურადღება გამახვილებული უსაფრთხოების პრობლემაზე. იგი აქტუალურია ნებისმიერ სისტემაში და მათ შორის ცოდნის გამოვლენისა და შეფასების კომპიუტერულ სისტემებში. გაანალიზებულია უსაფრთხოების მეთოდები და საშუალებები, ნაჩვენებია მათი დადებითი და უარყოფითი მხარეები. უსაფრთხოების პრობლემა განხილულია კომპლექსურად პროგრამული კოდის დაცვით დაწყებული სისტემის ადმინისტრატორის მიერ მონაცემების ბაზასთან შეღწევით დამთავრებული. შექმნილია ცოდნის გამოვლების და შეფასების სისტემის, ”კიბერგამოცდა”. რომლის-ის მაგალითზე დასაბუთებულია, რომ უსაფრთხოება უზრუნველყოფილ უნდა იქნეს საიმედოობისა და უსაფრთხოების დაცვის კომბინირებული მეთოდებით, როგორიცაა: იდენტიფიკაცია, პაროლების გამოყენება, ბაზის დაშიფრვა კრიფტოგრაფიის მეთოდებითა და ალგორითმებით. რადგან დაპროგრამების ენა VBA-ა არ არის მდიდარი არსებული ბიბლიოთეკებით, რომლითაც მოხდება თუნდაც ტექსტის შიფრაცია-დაშიფრაცია. ყველა ეს ალგორითმი შემუშავებულია და ალგორითმებს თან დაერთვის(ნებისმიერ მსურველს შეუძლია გამოიყენოს არსებული კოდი)

Annotation

It is considered reliability and safety of the computer systems. Particular attention is directed to the security problem. It is important to any system, including knowledge reveal and assessment electronic systems. It is given analysis the methods and means of the securite with its positive and negative sides. Specific system, "Cybertest" - is an example showing that safety should be ensured reliability and safety of combined methods, such as identification, the use of passwords, data encryption cryptography methods and algorithm. Because of programming language VBA is not rich their own libraries, the cryptography algorithms has been developed and is included as an attachment current document (this attachment programming code can use anyone)

შესავალი

ნებისმიერი პროგრამული სისტემის ერთერთი მთავარი კრიტერიუმებია საიმედოობა და უსაფრთხოება. საიმედოობის ქვეშ იგულისხმება სისტემის შეუფერხებელი ფუნქციონირება განსხვავებული სიტუაციებისას. ნებისმიერი პროგრამული სისტემა სამრეწველო ტირაჟირებამდე მრავალნაირ ტესტირებას გადის, ხორციელდება აღმოჩენილი შეცდომების ანალიზი და გასწორება. შეცდომები ძირითადად ორი სახისაა, პირველი - პროგრამის დაწერისას პროგრამისტის მიერ დაშვებული სინტაქსური, სემანტიკური შეცდომები და ტექნიკურ დავალებაში არსებული მოთხოვნების გაუთვალისწინებლობა; მეორე - პროგრამასთან მუშაობისას გაუთვალიწინებელი ე.წ. არასაშტატო სიტუაციების წარმოქმნა, მესამე - მომხმარებლის ობიექტურად განპირობებული სიტუციების გათვალიწინება. პროგრამული პროდუქტის დახვეწა პერმანენტული პროცესია, მოითხოვს წარმოქმნილი სიტუაცების ანალიზს, განზოგადოებას, დაძლევის გზებისა და საშუალებების ძიებას.მაგალითად ჩვენს მიერ შემუშავებული და დანერგილი ცოდნის გამოვლენისა და შეფასების ელექტრონული სისტემით "კიბერგამოცდა" სხვადასხვა საგნებში მრავალჯერ ჩატარდა გამოცდები. გამოცდების ჩატარების შემდეგ ხორციელდებოდა გამოცდის მიმდინარების დროს სტუდენტების მიერ არაკორექტული ქმედებებით შექმნილი სიტუაციების ანალიზი და პროგრამული პაკეტის კორექტირება. ამავე დროს წარმოებდა პედაგოგებისა და ადმინისტრაციული პერსონალის სურვილების გათვალისწინება. სისტემის რეალურ პირობებში გამოცდისა და კორექტირებების შედეგად მიღწეულ იქნა პროგრამული პაკეტის საიმედო ფუნქციონირება ავარიულ სიტუაციებშიც.

როგორც ცნობილია შესაძლებელია ნებისმიერ პროგრამულ სისტემის, თუნდაც ოპერაციული სისტემის პაროლის გაგება - სისტემის გატეხვა. აქ მთავარია ჰაკერის მიერ პროგრამაზე დადებული პაროლის გასაღების მოძებნაზე დახარჯული დრო და გასაღების მოძებნის სირთულე. პროგრამული პროდუქტების შემუშავებლები ცდილობენ რაც შეიძლება მეტი მახე დაუგონ ჰაკერს. განვიხილოდ ცოდნის გამოვლენისა და შეფასების პროგრამაში "კიბერგამოცდა" უსაფრთხოების რეალიზებული სისტემა.

"კიბერგამოცდა" სასწავლო პროცესის დაგეგმვისა და მართვის ელექტრონული სისტემის ერთერთი ფუნქციონალური ბლოკია. პროგრამები დაწერილია დაპროგრამების ენაზე VBA Access-ის გარემოში, მუშაობს ავტონომიურად და ქსელში. ავტონომიურად მუშაობისას

კლიენტის და სერვერის როლსაც Access-ი ასრულებს ხოლო ქსელში მუშაობისას რეალიზებულია კლიენტსერვერული ტექნოლოგია, სადაც Access კლიენტია, ხოლო MSSQL კი - სერვერი.

”კიბერგამოცდა”- სთან უშუალო კონტაქტი სტუდენტს აქვს. იგი დაინტერესებულია მონაცემების ბაზაში ტესტებისა და ამოცანების ნახვით (შემდგომში მათ შეკითხვებს უწოდებთ). ასეთივე დაინტერესება მერკანტული მოსაზრებებით შეიძლება სტუდენტის გარდა სხვა პირებსაც გაჩნდეთ: სისტემის ადმინისტრატორი, ოპერატორი და სხვა. ჩვენი მიზანია მონაცემების გაჟონვის ალბათობის მინიმუმამდე დაყვანა და გაჟონვის ობიექტის მარტივად იდენტიფიცირება. ”კიბერგამოცდა”- ში ტესტების, ამოცანების მონაცემების ბაზაში ჩაწერა ავტორის მიერ ხორციელდება, ამიტომ ”კიბერგამოცდა”-ში არსებული დაცვის რეალიზებული მექანიზმის არსებობისას ტესტების გავრცელების წყარო ავტორი შეიძლება იყოს.

დაინტერესებულ პირს (ჰაკერს) შეუძლია ინფორმაციის მოპარვის რამოდენიმე ხერხი გამოიყენოს ესენია:

- როდესაც სისტემა მუშაობს ავტონომიურ რეჟიმში:
    1. გადაწეროს პროგრამა (მოიპაროს), გაუშვას იგი სხვა კომპიუტერზე და მოახდინოს გამოცდის იმიტაცია.
    2. მოახდინოს Access-ის ფაილის გახსნა და ნახოს კითხვები.
    3. მოახდინოს Access-ის ფაილის გატეხვა და პროგრამული კოდის ნახვა.

- როდესაც სისტემა მუშაობს კლიენტ-სერვერულ რეჟიმში:
    1. სისტემის ადმინისტრატორმა ნახოს და გაავრცელოს ტესტები.
    2. სისტემის ადმინისტრატორმა გამოიყენოს სხვისი პაროლი და მოახდინოს გამოცდის იმიტაცია.
    3. სისტემის ადმინისტრატორმა მოახდინოს შედეგის გაყალბება.
    4. ჰაკერმა მოახდინოს ქსელის მოსმენა და შედეგში გარკვეული ცდომილებების შეტანა.

მოვახდინოთ თითოეული ამ პრობლემის განხილვა და მისი გადაჭრის გზები.

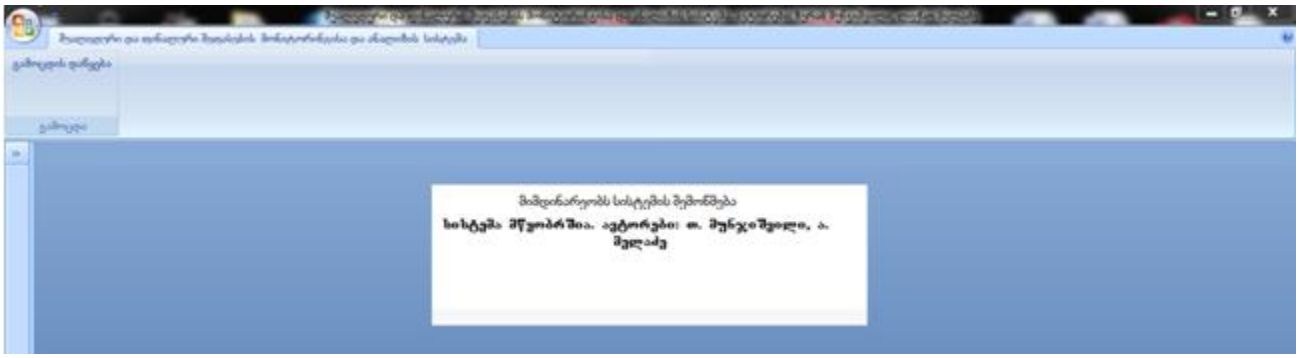
თუ სისტემა გამოიყენება ავტონომიურ რეჟიმში მუშაობისათვის, მაშინ მონაცემების ბაზა პროგრამული პაკეტის განუყოფელი ნაწილია. დაინტერესებულ პირს (ჰაკერს) შეუძლია გადაწეროს სისტემა და შეეცადოს მის გაშვებას სხვა კომპიუტერზე. მოახდინოს გამოცდის იმიტაცია და ნახოს კითხვები.

ამ პრობლემის თავიდან ასაცილებლად სისტემა გაშვებისთანავე თიშავს cd-ს და usb-ის დრაივერებს. რაც ხელს უშლის ფაილის გადაწერას. ასევე სტუდენტის მიერ სხვა ფაილის შეტანას.

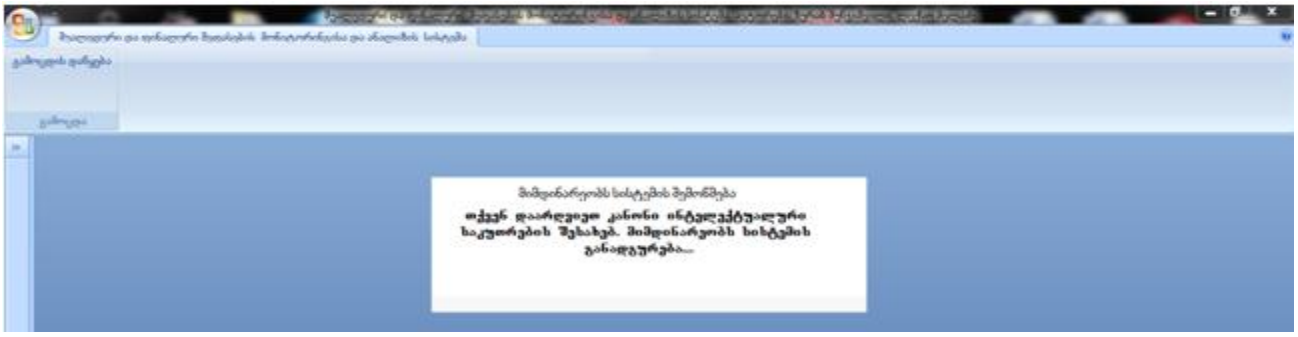
```
Public Function UsbOff()

    Dim RegKey As String

    RegKey = " HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\USBSTOR\Start"

    RegKeySave RegKey, 4

End Function
```

როგორც ფსევდო კოდიდან ჩანს დრაივერის გასათიშათ ვარედაქტირებთ რეგისტრებს რომელიც უზრუნველყოფს დრაივერის მუშაობას. თუმცა ასეთი დაცვა არ წყვეტს ჩვენს პრობლემას, რადგან ნებისმიერს შეუძლია რეგისტრების ცვლილება მოახდინოს და დრაივერი ისევ გაუშვას. ამიტომ ჩვენს მიერ შექმნილი ალგორითმით სისტემის დაინსტალირების დროს ავტომატურად ინახება Windows–ის უნიკალური ნომერი (იდენტიფიკატორი). იგი უცვლელია ოპერაციული სისტემის შეცვლამდე. ”კიბერგამოცდა”-ის გაშვებისას მოწმდება ინსტალაციისას დაფიქსირებული ნომრის შესაბამისობა მანქანაში დაყენებულ ოპერაციული სისტემის ნომერთან. შესაბამისობისას გამოდის შეტყობინება მზადყოფნის შესახებ/სურ.1/, წინააღმდეგ შემთხვევაში სისტემა იწყებს

თვითგანადგურებას: იშლება პროგრამული მოდულები და მონაცემების ბაზა/სურ.2/.



სურ.1. შეტყობინება სისტემის მზადყოფნის შესახებ



სურ.2. შეტყობინება პროგრამულ სისტემასა და მონაცემების ბაზაში არასანქციონირებული შეღწევის შესახებ

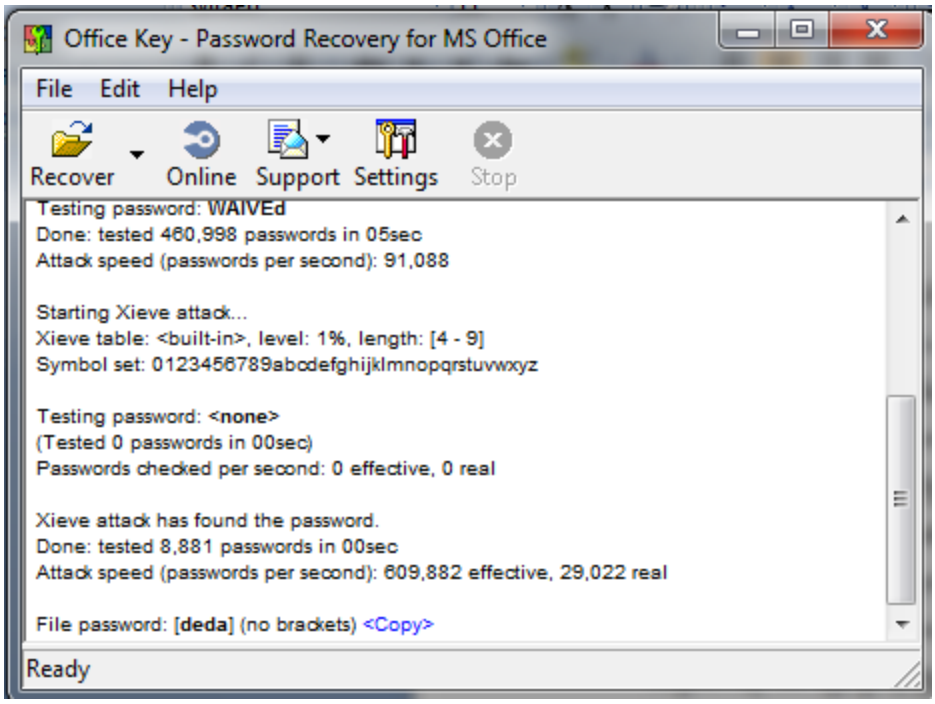
ვინდოუსის უნიკალური ნომრის სანახავად ვიყენებთ შემდეგ ფსევდოკოდს:

```
Public Function getWindowID()
Dim strComputer As String
Dim objWMIService As Object, objProcessor As Object
   strComputer = "."
   Set objWMIService = GetObject("winmgmts:{impersonationLevel=impersonate}!\\" & strComputer &
"\root\cimv2")
   Set winSID = objWMIService.ExecQuery("SELECT * FROM Win32_SID")
   For Each objWinID In winSID
      getWindowID = winSID.SID
   Next
End Function
```

არსებოვს სხვადასხვა მეთოდები კომპიუტერის უნიკალურობის შესამოწმებლად. ესენია პროცესორის უნიკალური ნომრის შემოწმება. დედაპლატის უნიკალური ნომრის შემოწმება და ა.შ. თუმცა პრაქტიკამ გვაჩვენა რომ საუკეთესო მაინც ვინდოუსის უნიკალურობის შემოწმებაა, რადგან ძველ პროცესორებს არ ენიჭებოდა ასეთი უნიკალური ნომერი რომელიც პროგრამულად ხელმისაწვდომი იქნებოდა.

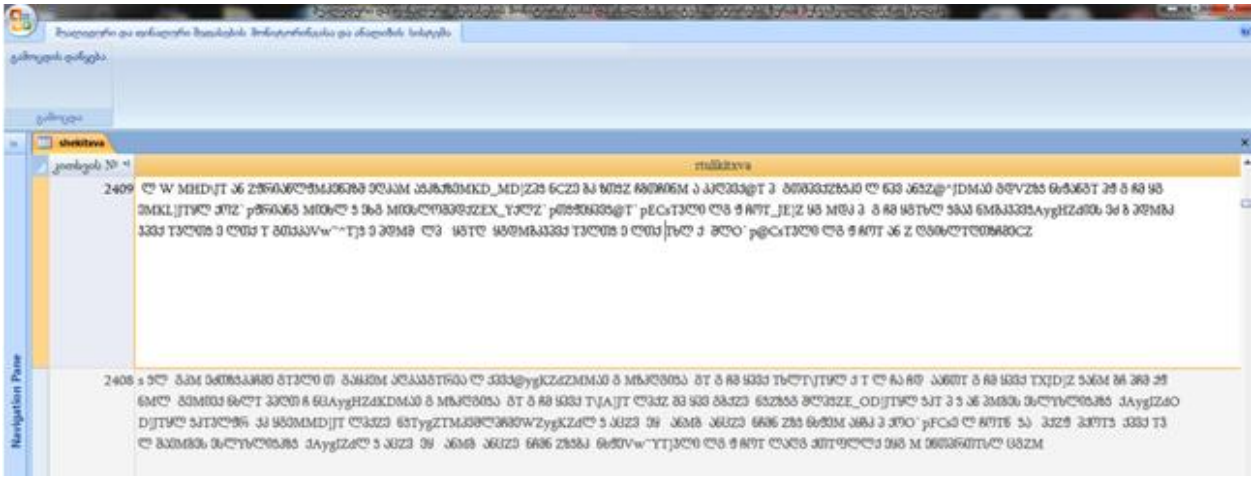
მოახდინოს Access-ის ფაილის გახსნა და ნახოს კითხვები.

Access-ში ორგანიზებულ მონაცემთა ბაზებთან წვდომა შესაძლებელია ნებისმიერი პროგრამული პროდუქტიდან თუნდაც Excel–იდან. Excel–იდან შესაძლებელია დავათვალიეროთ მონაცემთა ბაზების ცხრილები. ამის თავიდან ასაცილებლად გამოიყენება ბაზებზე პაროლის დადება. თუ ბაზები რეალიზებულია MSSQL–ზე მაშინ პაროლის გატეხვა რთულია, რადგან SQL-ში ბაზის დაცვის ძლიერი მექანიზმია რეალიზებული. მასში შესაძლებელია Log-ის შექმნა პაროლის დადება და Roll–იბის (უფლებების) გაწერა. ამით შეგვიძლია ავკრძალოთ გარკვეულ ცხრილებთან კავშირი და ინფორმაციის ინტერნეტის ქსელიდან დათვალიერება. მაგრამ თუ ბაზები რეალიზებულია Access–ში Log-ების გაწერა არ შეგვიძლია და მხოლოდ პაროლის დადების უფლება გაგვაჩნია. პაროლი კი საკმაოდ მარტივად ტყდება, მაგალითად პროგრამით Passware /სურ.3/

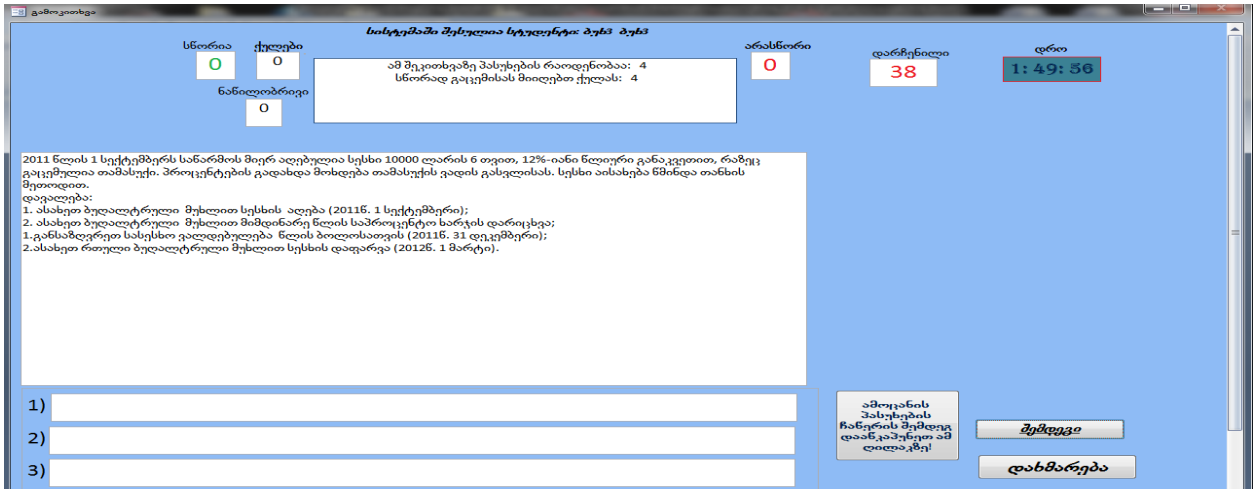


სურ.3. პაროლის გატეხვა პროგრამით Passware

MSSQL–ში პაროლის გატეხვა ძალიან რთულია, მაგრამ შესაძლებელია. ბაზის ადმინისტრატორმა შეიძლება ნახოს კითხვები და გაავრცელოს. ამ პრობლემების თავიდან ასაცილებლად საუკეთესო მეთოდია ტექსტის დაშიფვრა და ბაზაში დაშიფრული ტექსტის შენახვა. შეკითხვები , პასუხები და ყველა ის ინფორმაცია რაც აკრძალულია რომ ნახოს სტუდენტმა ყოველი ეს ტექსტი იშიფრება და ისე ინახება ბაზაში . (შეფრაციის ალგორითმებს მოგვიანებით მივუბრუნდებით).



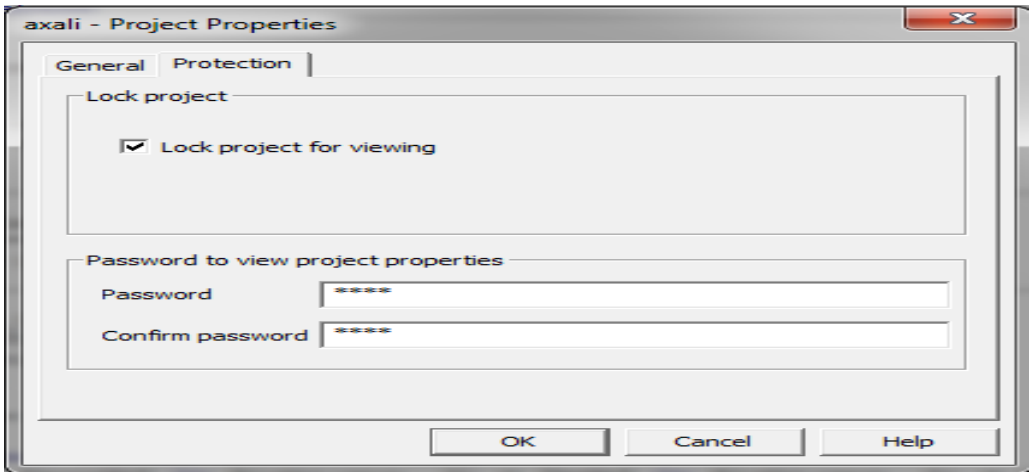
სურ.4. ბაზაში შენახული დაშიფრული კითხვები



სურ.5. გამოცდის პროცესი - განხორციელებულია შეკითხვის დეშიფრაცია

მოახდინოს Access-ის ფაილის გატეხვა და პროგრამული კოდის ნახვა.

ჰაკერს ასევე შეუძლია Access-ის გატეხვა და პროგრამული კოდის ნახვა და მასში ცვლილებების შეტანა. არსებობს ასეთი პრობლემის თავიდან აცილების რამოდენიმე მიდგომა: პირველი ეს არის პროგრამულ კოდს დავადოთ პაროლი



სურ.6. პროგრამულ კოდზე პაროლის დადება

თუმცა პაროლის გატეხვასაც არაა რთული ამიტომ უფრო მოსახერხებელი გზაა - დავიყვანოთ კოდი .ACCDE ფორმატში ანუ ორობით კოდში . .ACCDE ფორმატში დაყვანილი კოდის გახსნაც შესაძლებელია Disassembler-ით, რომელიც იღებს ორობით კოდს და დაჰყავს ასემბლერის პროგრამირების ენაზე , თუმცა ცნობილია რომ ასეთი მეთოდით პროგრამის მუშაობის ლოგიკის გაგება ძალიან რთულია .

სისტემის ადმინისტრატორმა ნახოს და გაავრცელოს ტესტები.

სისტემის ადმინისტრატორს ,რომელიც მართავს მონაცემთა ბაზას , შეუძლია ბაზაში არსებული მონაცემების ნახვა და მისი გავრცელება. ამის თავიდან ასაცილებლად ყველა ის მონაცემი რომლის გავრცელება გამოიწვევს გამოცდის პროცესის შეფერხებას (შეკითხვები , შეკითხვების პასუხები, სტუდენტის მიერ გაცემული პასუხები) ბაზაში ინახება დაშიფრული, ანუ სისტემის ადმინისტრატორსაც არ შეუძლია ნახოს კითხვები. არსებობს შიფრაცია-დეშიფრაციის მრავალი ალგორითმი. არსებობს როგორც სიმეტრიული შიფრები ასევე ასიმეტრიულ (ღია გასაღებიანი) შიფრები. რადგან ჩვენ ვმუშაობთ დიდ ტექსტებთან ამიტომ დიდი ტექსტების დასაშიფრად ვიყენებთ სიმეტრიულ შიფრებს, რადგან მის უპირატესობას ასიმეტრიულთან შედარებით წარმოადგენს სიმარტივე და მეტი სისწრაფე. სიმეტრიული ალგორითმის სიჩქარე ასიმეტრიულთან შედარებით დაახლოებით 800—1000-ჯერ მეტია, რაც მიიღწევა მარტივი არითმეტიკული გამოთვლების გამოყენებით. განვიხილოთ სიმეტრიული ალგორითმები.

სიმეტრიული კრიპტოსისტემა წარმოადგენს ისეთ სისტემას, რომელშიც ინფორმაციის შიფრაცია/დეშიფრაციისათვის გამოიყენება 1 გასაღები. მოკავშირე მხარეებს შორის ინფორმაციის გაცვლის დაწყებამდე გასაღები უნდა იყოს შეთანხმებული და შენახულ იქნას საიდუმლოდ მანამდე, სანამ ინფორმაციის საიდუმლოება უნდა იქნეს უზრუნველყოფილი. იდეალური სიმეტრიული ალგორითმის უსაფრთხოება მთლიანად განისაზღვრება გასაღების უსაფრთხოებით. სანამ უსაფრთხო და დაცულია გასაღები, მანამ დაცულია თვითონ სისტემაც. სიმეტრიული ალგორითმით შიფრაცია და დეშიფრაცია აღინიშნება შემდეგნაირად:

$$EK(M)=C$$
$$DK(C)=M$$

სადაც M — ტექსტია, C — შიფროტექსტი, K — გასაღები, E და D — შესაბამისად შიფრაციის და დეშიფრაციის ალგორითმები. E და D უნდა იყოს ურთიერთშექცევადი ფუნქციები, ანუ უნდა სრულდებოდეს პირობა : $M=DK(EK(M))$

სიმეტრიული შიფრები იყოფა 2 ნაწილად: ბლოკური შიფრი და ნაკადური შიფრი. ბლოკური შიფრები საწყის ინფორმაციას ყოფენ ბიტების (ან ბაიტების)განსაზღვრული სიგრძის ბლოკებად და შემდგომ შიფრავენ გასაღების გამოყენებით. ბლოკების სიგრძის

სტანდარტული მნიშვნელობებია 64, 128 ან 256 ბიტი. გამომავალი შიფროტექსტიც წარმოადგენს განსაზღვრული სიგრძის (ხშირად იგივე, რაც შემავალი ტექსტის სიგრძე) ბლოკს. გამომავალი შიფროტექსტის ბლოკების ერთმანეთთან დამოკიდებულების შესაქმნელად გამოიყენება ე.წ. მოქმედების რეჟიმები. გამოიყენება ე.წ. იტერაციული შიფრები, რომლის დროსაც ტექსტის 1 ბლოკი რამდენჯერმე მუშავდება.თითოეულ იტერაციას ეწოდება რაუნდი. როგორც წესი, თანამედროვე შიფრებში რაუნდების რაოდენობა 4-32-ია. მათი უმეტესობა ემყარება ე.წ. ფეისტელის ქსელებს, რომელიც ავტომატურად უზრუნველყოფს შიფრაციის და დაშიფრაციის ალგორითმების ურთიერთშექცევადობას, მაშინაც კი, როდესაც მასში შეუქცევადი მათემატიკური ფუნქციები გამოიყენება. ძირითადი შემადგენელი ნაწილებია S-ბლოკები, არითმეტიკული ოპერაციები, ბულის ოპერაციები (XOR) და სხვადასხვა პერმუტაციები. არსებობს ბევრი სიმეტრიული ალგორითმი , რომლებიც დაპატენტებულია და დღეს-დღეისობით გამოიყენება AES, TWOFISH, SKIPJACK, DES, IDEA, CAST, BLOWFISH და სხვ.

ჩვენი პროგრამის შემთხვევაში ტესტების დასაშიფრად გამოყენებულია BLOWFISH -ის ალგორითმი. ავღწეროთ ეს ალგორითმი:

BLOWFISH -ის ავტორია მსოფლიოში ერთერთი ყველაზე ცნობილი კრიპტოლოგი ბრიუს შნეიერი (**Bruce Schneier**). როგორც აღნიშნავს ავტორი, მან ალგორითმი შექმნა 1994 წელს, როდესაც DES-ის გამოყენება უკვე არასაიმედო გახდა და ყველა სხვა ალგორითმი კი დაპატენტებული იყო. ბ. შნეიერის მიზანს წარმოადგენდა რომ ნებისმიერ ადამიანს ჰქონოდა საშუალება დაეშიფრა მისთვის საჭირო ინფორმაცია საიმედოდ, ამიტომ მას არ აუღია პატენტი, პირიქით, მან გამოაქვეყნა ალგორითმის პროგრამული უზრუნველყოფაც.

ალგორითმის შექმნის დროს ავტორისთვის მთავარი კრიტერიუმები იყო

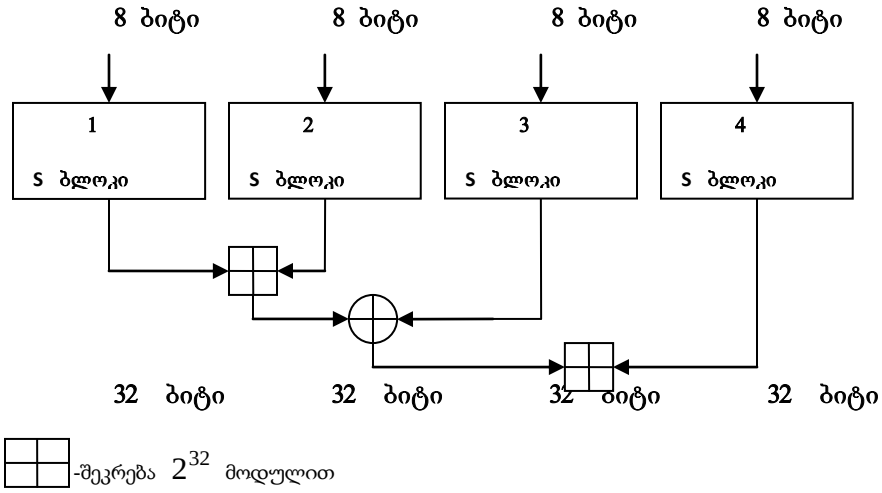
- სისწრაფე;
- კომპაქტურობა;
- სიმარტივე და
- კრიპტომედეგობის ვარირების საშუალება.

ოთხივე ეს პარამეტრი ალგორითმში პრაქტიკულად მიღწეულია. ალგორითმის პროგრამული რეალიზაცია ოცდათორმეტ ბიტიან მიკროპროცესორებზე მონაცემებს შიფრავს ბაიტზე ოცდაექვსი ტაქტის სიჩქარით, პროგრამული რეალიზაცია იკავებს ხუთ კილობაიტ მეხსიერებას. ალგორითმში გამოყენებულია მხოლოდ ორი ოპერაცია, **XOR**-ით შეკრება და ოცდათორმეტ ბიტიანი ოპერანდის ამოღება ცხრილიდან. BLOWFISH -ის გასაღების სიგრძე იცვლება 32 ბიტიდან 448 ბიტამდე, რაც საშუალებას იძლევა შევარჩიოთ ალგორითმის სასურველი კრიპტომედეგობა.

$F(x)$ **ფუნქციის გამოთვლა.** ისევე როგორც სხვა მრავალ ალგორითმში, BLOWFISH -შიც გამოყენებულია ჰ. ფეისტელის სქემა. თექვსმეტრაუნდიანი ალგორითმი მუშაობს სამოცდაოთხ ბიტიან ბლოკთან ცვლადი სიგრძის გასაღებით, რომელიც შეიძლება იცვლებოდეს **32** ბიტიდან **448** ბიტამდე.

თითოეულ რაუნდში სრულდება გასაღებზე დამოკიდებული გადანაცვლება და გასაღებზე დამოკიდებული ჩანაცვლება. ამიტომ სანამ ალგორითმი დაიწყებს უშუალოდ მონაცემთა დაშიფრვას, ჯერ საჭიროა საწყისი საიდუმლო გასაღების საშუალებით თექვსმეტი რაუნდისთვის საჭირო $P_1 - P_{18}$ ოცდათორმეტბიტიანი ქვეგასაღებების და ჩანაცვლების ოთხი S ბლოკის მომზადება, რომელთა საშუალებითაც უნდა მოხდეს რაუნდული დაშიფრვა. ჩანაცვლების ცხრილების მიღების ალგორითმი მოყვანილია სურ. №7. შესასვლელი ოცდათორმეტი ბიტი გაიყოფა ოთხ რვაბიტიან ქვებლოკებად $[X_1, X_2, X_3, X_4]$ და თითოეული შედის ერთ S ბლოკში. ეს რვა ბიტი წარმოადგენს S_j ბლოკის ინდექსს. ბლოკის გამოსასვლელი კი არის ოცდათორმეტბიტიანი სტრიქონი, რომელიც მიიღება შემდეგნაირად. S_{1,X_1} და S_{2,X_2} იკრიბება მოდულით 2^{32}. მიღებული შედეგი იკრიბება XOR-ით S_{3,X_3} -თან და მიღებული შედეგი კვლავ იკრიბება მოდულით 2^{32} S_{4,X_4} . ესაა - $F(x)$ ის მნიშვნელობა.

$$F(x) = (((S_{1,X_1} + S_{2,X_2}) \bmod 2^{32} \oplus S_{3,X_3}) + S_{4,X_4}) \bmod 2^{32}$$



სურ. №7 ფუნქციის გამოთვლის სქემა.

ამგვარად თითოეული ოცდათორმეტბიტიანი S ბლოკი შეიცავს **256** ელემენტს.

$$S_{1,0}, S_{1,1}, \ldots, S_{1,255}$$
$$S_{2,0}, S_{2,1}, \ldots, S_{2,255}$$
$$S_{3,0}, S_{3,1}, \ldots, S_{3,255}$$
$$S_{4,0}, S_{4,1}, \ldots, S_{4,255}$$

თითო რაუნდში გამოიყენება თითო ქვეგასაღები, ანუ თექვსმეტი რაუნდისთვის საჭიროა თექვსმეტი ქვეგასაღები. კიდევ ორი ქვეგასაღები საჭიროა დამამთავრებელი გარდაქმნისთვის, ასე, რომ საერთო სიგრძე ქვეგასაღებებისა შეადგენს **576** ბიტს.

```
Public Function blf_BytesRaw(abData() As Byte, bEncrypt As Boolean) As Variant
    Dim nLen As Long, nBlocks As Long, iBlock As Long, j As Long
    Dim abOutput() As Byte, abBlock(7) As Byte
    Dim iIndex As Long
```

```
    nLen = UBound(abData) - LBound(abData) + 1
    nBlocks = nLen \ 8

    ReDim abOutput(nBlocks * 8 - 1)
    iIndex = 0
    For iBlock = 1 To nBlocks
        ' Get the next block of 8 bytes
        CopyMemory VarPtr(abBlock(0)), VarPtr(abData(iIndex)), 8&
        If bEncrypt Then
            Call blf_EncryptBytes(abBlock())
        Else
            Call blf_DecryptBytes(abBlock())
        End If
        ' Copy to output string
        CopyMemory VarPtr(abOutput(iIndex)), VarPtr(abBlock(0)), 8&
        iIndex = iIndex + 8
    Next
    blf_BytesRaw = abOutput
End Function
```

ქვეგასაღებისა და S ბლოკების გამოთვლა. ქვეგასაღებისა და ჩანაცვლების ოთხი S ბლოკის გამოსათვლელად, რომლებიც ასევე დამოკიდებულნი არიან გასაღებზე უნდა ჩავატაროთ შემდეგი პროცედურები:

$P_1 - P_{18}$ ფიქსირებულ სტრიქონებში და ოთხივე S ბლოკში უნდა ჩაიწეროს ნებისმიერი საწყისი მიმდევრობა (ავტორი გვირჩევს ჩავწეროთ თექვსმეტობით სისტემაში წარმოდგენილი π რიცხვის მანტისა).

1.ამის შემდეგ საიდუმლო გასაღების პირველი ოცდათორმეტი ბიტი შეიკრიბება XOR-ით P_1-ის ოცდათორმეტ ბიტთან, მეორე ოცდათორმეტი ბიტი P_2-ის ოცდათორმეტ ბიტთან და ასე შემდეგ. თუ არ გვყოფნის საიდუმლო გასაღების სიგრძე, ხდება ბიტების ციკლური გამეორება.

2.ავიღოთ ნოლებისგან შემდგარი ოცდათორმეტბიტიანი სტრიქონი და დავშიფროთ იმ ქვეგასაღებებით, რომლებიც მივიღეთ მეორე ეტაპზე.

3.შევცვალოთ მესამე ეტაპზე მიღებული შიფროტექსტით P_1 და P_2 ქვეგასაღებები.

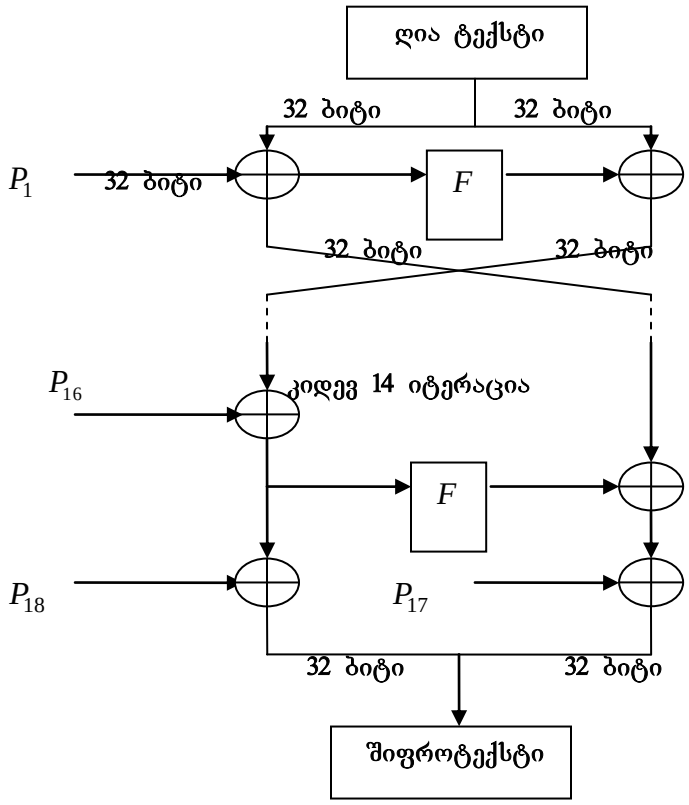
4.დავშიფროთ მესამე ეტაპზე მიღებული ტექსტი ახალი ქვეგასაღებების გამოყენებით.

5.შევცვალოთ მეხუთე ეტაპზე მიღებული ტექსტით P_3 და P_4 ქვეგასაღებები;

6.გავაგრძელოთ ეს პროცედურა სანამ არ შევცვლით ყველა ქვეგასაღებს და შემდეგ ოთხივე S ბლოკს ასეთი მიმდევრობითი დაშიფრვის შედეგებით.

სულ დაგვჭირდება 521 იტერაცია, რათა მივიღოთ ყველა საჭირო ქვეგასაღები S ბლოკების მნიშვნელობათა ჩათვლით, რომელთა საერთო სიგრძეც იქნება 4168 ბაიტი. ცხადია, რომ ყველა ეს პროცედურა უნდა ჩატარდეს წინასწარ, სანამ დაიწყება გადასაცემი, ან შესანახი ღია ტექსტის დაშიფრვა და მიღებული ქვეგასაღებები შევინახოთ მეხსიერებაში

სურ. №6.5.2. ალგორითმ **BLOWFISH**-ის სქემა.



სურ. #8 ალგორითმ **BLOWFISH**-ის სქემა.

ალგორითმის აღწერა. დაშიფრვა ხდება შემდეგი სქემით (იხ. სურ. #8.). დასაშიფრი 64 ბიტი გაიყოფა ორ, მარცხენა და მარჯვენა ქვებლოკებად. მარცხენა ნახევარი (32 ბიტი) შეიკრიბება P_1 ქვეგასაღებთან XOR ოპერაციით და შედის $F(x)$ -ს ბლოკში საიდანაც გამოსული 32 ბიტი ტექსტი შეიკრიბება მარჯვენა 32 ბიტთან. ამის შემდეგ მარჯვენა და მარცხენა ქვებლოკები გაცვლიან ადგილებს და იგივე პროცედურა გამეორდება კიდევ თხუთმეტჯერ. თექვსმეტი იტერაციის შემდეგ მიღებული 32 ბიტიანი მარჯვენა და მარცხენა ნახევრები კიდევ ერთხელ შეიკრიბება $P_{1\ 7}$ და P_{18} ქვეგასაღებებთან, რის შემდეგაც მოხდება მათი კონკატაცია და მიიღება 64 ბიტი დაშიფრული ტექსტი.

```
Public Function blf_BytesEncRawCBC(abData() As Byte, abInitV() As Byte) As Variant

    Dim nLen As Long, nBlocks As Long, iBlock As Long
```

```
Dim abBlock(7) As Byte

Dim iIndex As Long

Dim abReg(7) As Byte, abOutput() As Byte

' Initialisation vector should be a 8-byte array

' This will add zero bytes if too short or chop off any extra

ReDim Preserve abInitV(7)

' Calc number of 8-byte blocks

nLen = UBound(abData) - LBound(abData) + 1

nBlocks = nLen \ 8



' Dimension output

ReDim abOutput(nBlocks * 8 - 1)



CopyMemory VarPtr(abReg(0)), VarPtr(abInitV(0)), 8&

iIndex = 0

For iBlock = 1 To nBlocks

   CopyMemory VarPtr(abBlock(0)), VarPtr(abData(iIndex)), 8&



   ' XOR with feedback register = Pi XOR C_i-1

   Call bXorBytes(abBlock, abReg, 8)



   ' Encrypt the block Ci = Ek(Pi XOR C_i-1)
```

```
    Call blf_EncryptBytes(abBlock())
    CopyMemory VarPtr(abReg(0)), VarPtr(abBlock(0)), 8&
    CopyMemory VarPtr(abOutput(iIndex)), VarPtr(abBlock(0)), 8&
    iIndex = iIndex + 8
  Next
  blf_BytesEncRawCBC = abOutput
End Function
```

დეშიფრაცია. დეშიფრაცია ხდება ზუსტად ისევე, როგორც შიფრაცია (იმიტომ, რომ ალგორითმი იყენებს ჰ. ფეისეტლის სქემას). საჭიროა მხოლოდ გასაღებები $P_1, P_2, ..., P_{18}$ გამოვიყენოთ შებრუნებული მიმდევრობით.

BLOWFISH -ის კრიპტოანალიზი. ჩატარებულმა კრიპტოანალიზის შედეგებმა აჩვენა, რომ თუ ცნობილია S ბლოკები, მაშინ r რაუნდიანი ალგორითმისთვის დიფერენციალური კრიპტოანალიზის საშუალებით შეიძლება აღვადგინოთ გასაღებების მასივი, ოღონდ ამისთვის საჭიროა 2^{8r+1} შერჩეული ღია ტექსტი. ზოგიერთი სუსტი გასაღებებისთვის, რომლებიც ახდენენ ცუდი S ბლოკების გენერირებას (სუსტი გასაღების ამორჩევის ალბათობა ტოლია $1/2^{14}$), იგივე შეტევა აღადგენს გასაღებების მასივს მხოლოდ 2^{4r+1} შერჩეული ღია ტექსტის საშუალებით. თუ S ბლოკები უცნობია, იგივე შეტევას შეუძლია აღმოაჩინოს, რომ გამოყენებული იყო სუსტი გასაღები, მაგრამ არ შეუძლია თვით გასაღების აღმოჩენა. აშკარაა, რომ თექვსმეტრაუნდიანი **BLOWFISH** -ის წინააღმდეგ ეს შეტევა უძლურია. რაც შეეხება სუსტ გასაღებებს, ამ ალგორითმისთვის სუსტ გასაღებებს წარმოადგენენ ისეთი გასაღებები, რომლებიც მოცემული S ბლოკისთვის არჩევენ მინიმუმ ორ ერთნაირ ელემენტს. სანამ არ მოხდება გასაღების გაფართოება, ამ მომენტის აღმოჩენა შეუძლებელია, ამიტომ, თუ თქვენ არ გსურთ, რომ გამოიყენოთ სუსტი გასაღები, მაშინ თქვენ ჯერ უნდა მოახდინოთ გასაღების გაფართოება და შემდეგ შეამოწმოთ, ხომ არ არის S ბლოკში ერთნაირი ელემენტები.

სისტემის ადმინისტრატორმა გამოიყენოს სხვისი პაროლი და მოახდინოს გამოცდის იმიტაცია.

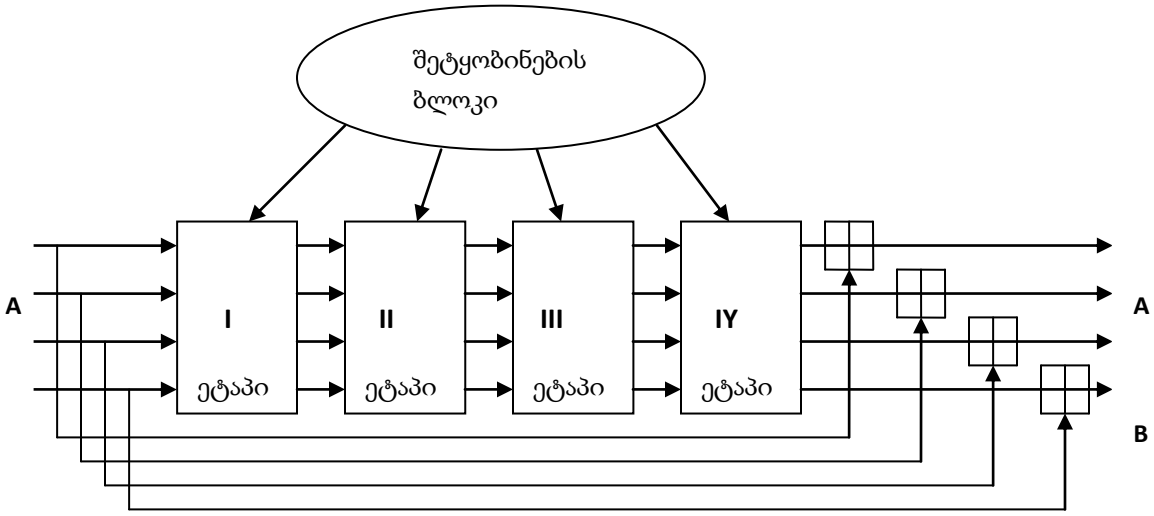
სისტემის ადმინისტრატორს შეუძლია ნახოს სტუდენტის პაროლი, მოახდინოს მისი მონაცემებით სისტემაში ავტორიზაცია და ჩააბაროს გამოცდა (სტუდენტის მაგივრად) ან მოახდინოს კითხვების ნახვა და შემდგომში მათი გავრცელება. ანალოგიური პრობლემა შეიძლება ნებისმიერი სისტემის წინაშე იდგეს. ასეთი პრობლემების თავიდან ასაცილებლად ხშირად იყენებენ პაროლის დაშიფვრას რომელიმე ჰეშ-ფუნქციის გამოყენებით. არსებობს რამოდენიმე მსგავსი ჰეშ-ფუნქცია ესენია: MD5, SHA-1, WHIRLPOOL, SHA-3(Keccak). ასეთი ალგორითმების მთავარი იდეა მდგომარეობს იმაში რომ ისინი არიან ცალმხრივმიმართული ფუნქციები ანუ ამ ფუნქციების გამოყენებით ხდება ტექსტის ისე დაშიფვრა რომ დაშიფრული ტექსტისგან საწყისი ტექსტის აღდგენა შეუძლებელია. მათგან ყველაზე გავრცელებულია MD5-ი ალგორითმი , თუმცა ყველაზე თანამედროვე და დაცულ ფუნქციად დღეს-დღეისობით ითვლება SHA-3(Keccak) – ხელმძღვანელი იოან დამენი (Joan Daemen) , რომელმაც 2012 წელს გაიმარჯვა კონკურსში და დაპატენტდა. განვიხილოთ ყველაზე პოპულარული ალგორითმი MD5.

ალგორითმი MD-5 შეტყობინების წინასწარი დამუშავების შემდეგ გადაამუშავებს მას 512 ბიტიან ბლოკებად, რომლებიც დაყოფილია თექვსმეტ 32 ბიტიან ქვებლოკებად. ალგორითმის გამოსასვლელია ოთხი 32 ბიტიანი ქვებლოკი, რომლებიც ერთიანდებიან ერთ 128 ბიტიან ბლოკში. წინასწარი დამუშავების დროს ტექსტის სიგრძე იზრდება 512*k-64 ბიტამდე, ამ მიზნით მას ემატება 64 ერთიანი და იმდენი ნოლი რამდენიც საჭიროა შესაბამისი სიგრძის მისაღებად. ამის შემდეგ მას ემატება 64 ბიტიანი რიცხვი, რომელიც აღნიშნავს შეტყობინების სიგრძეს (რეალურს და არა დამატების შედეგად მიღებულს). ეს ხდება იმისთვის, რომ ჯერ ერთი ტექსტის ზომა ჯერადი იყოს 512-ის, რაც აუცილებელია ალგორითმის შემდგომი მუშაობისთვის და მეორეც, ჩანაწერი ტექსტის სიგრძის შესახებ იძლევა გარანტიას, რომ სხვადასხვა შეტყობინებები არ დაემსგავსებიან ერთმანეთს დამატების შემდეგ. ალგორითმში განისაზღვრება ოთხი ცვლადი, რომლებსაც უწოდებენ გადაბმის ცვლადებს და რომელთა მნიშვნელობები თექვსმეტობით სისტემაში ასეთია:

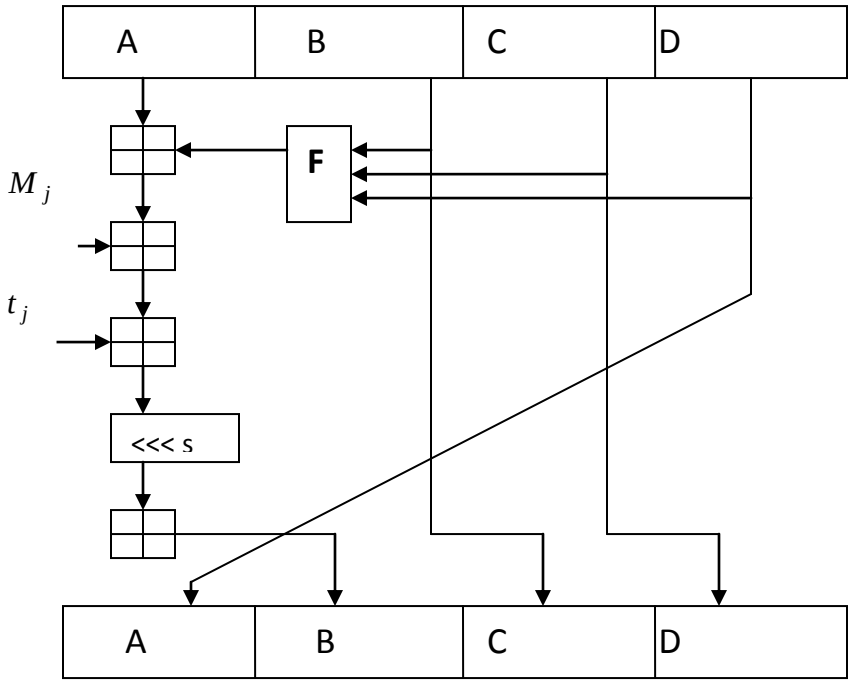
$$A = (01234567)_{hex}; B = (89absdef)_{hex}; C = (fedcba98)_{hex}; D = (76543210)_{hex}$$

ალგორითმის მთავარი ციკლის სქემა მოყვანილია (სურ. 9). ეს ციკლი გრძელდება სანამ არ ამოიწურება შეტყობინების ყველა 512 ბიტიანი ბლოკი. ციკლში შესვლის წინ ხდება

ცვლადების კოპირება შესაბამისად a,b,c და d ცვლადებში და 512 ბიტიანი ტექსტი დაიყოფა 16 ოცდათორმეტიან ქვებლოკებად. მთავარი ციკლი შედგება ოთხი, ერთმანეთის მსგავსი გარდაქმნისგან, რომელსაც ეწოდება ეტაპი. თითოეულ ეტაპზე თექვსმეტჯერ (ერთი ქვებლოკისთვის ერთხელ) გამოიყენება სხვადასხვა ოპერაციები და თითოეული ასეთი ოპერაცია წარმოადგენს არაწრფივ ფუნქციას. ოპერაციები სრულდება მოცემული ოთხი ცვლადიდან სამზე (იხ. სურ. 10).



სურ. #9 md5 ალგორითმის მთავარი ციკლის სქემა



სურ. #10. MD5 – ის ერთი ეტაპის სქემა. აქ M_j არის 32 ბიტიანი ქვებლოკი ($j = \overline{1,16}$), t_j - ცვლადია, F – არაწრფივი ფუნქციაა, $<<< s$ ნიშნავს წაძვრას მარცხნივ s თანრიგით და შეკრებას 32 მოდულით

მიღებულ შედეგს შემდეგ ემატება ჯერ მეოთხე ცვლადის მნიშვნელობა, შემდეგ ტექსტის ქვებლოკი (M_j) და შემდეგ მუდმივი რიცხვი (t_j). ყოველ ეტაპზე გვაქვს განსხვავებული არაწრფივი ფუნქცია:

პირველ ეტაპზე - $F(X,Y,Z) = (X \wedge Y) \vee ((\neg X \wedge Z)$

მეორეზე – $G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge (\neg Z))$

მესამეზე – $H(X,Y,Z) = X \oplus Y \oplus Z$

მეოთხეზე – $I(X,Y,Z) = Y \oplus (X \vee (\neg Z))$

თითოეულ ეტაპზე თექვსმეტჯერ სრულდება შემდეგი ოპერაციები:

$$a = b + ((a + F(b,c,d) + M_j + t_j) <<< s)$$
$$a = b + ((a + G(b,c,d) + M_j + t_j) <<< s)$$

$$a = b + ((A + H(b,c,d) + M_j + t_j) <<< s)$$
$$a = B + ((a + I(b,c,d) + M_j + t_j) <<< s)$$

ამასთან ოპერაციების ყოველი შესრულების დროს t_j ღებულობს განსხვავებულ მნიშვნელობებს. ოთხივე ეტაპის დამთავრების შემდეგ მიღებული შედეგი ოცდათორმეტის მოდულით კვლავ იკრიბება გადაბმის ცვლადებთან და ალგორითმი იწყებს ახალი ბლოკის დამუშავებს. საბოლოო გამოსასვლელს წარმოადგენს გადაბმის ცვლადების კონკატაცია.მისი პროგრამული კოდი ასეთია:

```
Public Function MD5(sMessage As String) As String

    Dim X, K, AA, BB, CC, DD

    Dim a, b, C, d

    Const S11 = 7, S12 = 12, S13 = 17, S14 = 22, S21 = 5, S22 = 9, S23 = 14, S24 = 20, S31 = 4

    Const S32 = 11, S33 = 16, S34 = 23, S41 = 6, S42 = 10, S43 = 15, S44 = 21


    init

    X = ConvertToWordArray(sMessage)

    a = &H67452301

    b = &HEFCDAB89

    C = &H98BADCFE

    d = &H10325476

    For K = 0 To UBound(X) Step 16

        AA = a

        BB = b

        CC = C

        DD = d


        FF a, b, C, d, X(K + 0), S11, &HD76AA478

        FF d, a, b, C, X(K + 1), S12, &HE8C7B756
```

FF C, d, a, b, X(K + 2), S13, &H242070DB

FF b, C, d, a, X(K + 3), S14, &HC1BDCEEE

FF a, b, C, d, X(K + 4), S11, &HF57C0FAF

FF d, a, b, C, X(K + 5), S12, &H4787C62A

FF C, d, a, b, X(K + 6), S13, &HA8304613

FF b, C, d, a, X(K + 7), S14, &HFD469501

FF a, b, C, d, X(K + 8), S11, &H698098D8

FF d, a, b, C, X(K + 9), S12, &H8B44F7AF

FF C, d, a, b, X(K + 10), S13, &HFFFF5BB1

FF b, C, d, a, X(K + 11), S14, &H895CD7BE

FF a, b, C, d, X(K + 12), S11, &H6B901122

FF d, a, b, C, X(K + 13), S12, &HFD987193

FF C, d, a, b, X(K + 14), S13, &HA679438E

FF b, C, d, a, X(K + 15), S14, &H49B40821

GG a, b, C, d, X(K + 1), S21, &HF61E2562

GG d, a, b, C, X(K + 6), S22, &HC040B340

GG C, d, a, b, X(K + 11), S23, &H265E5A51

GG b, C, d, a, X(K + 0), S24, &HE9B6C7AA

GG a, b, C, d, X(K + 5), S21, &HD62F105D

GG d, a, b, C, X(K + 10), S22, &H2441453

GG C, d, a, b, X(K + 15), S23, &HD8A1E681

GG b, C, d, a, X(K + 4), S24, &HE7D3FBC8

GG a, b, C, d, X(K + 9), S21, &H21E1CDE6

GG d, a, b, C, X(K + 14), S22, &HC33707D6

GG C, d, a, b, X(K + 3), S23, &HF4D50D87

GG b, C, d, a, X(K + 8), S24, &H455A14ED

GG a, b, C, d, X(K + 13), S21, &HA9E3E905

GG d, a, b, C, X(K + 2), S22, &HFCEFA3F8

GG C, d, a, b, X(K + 7), S23, &H676F02D9

GG b, C, d, a, X(K + 12), S24, &H8D2A4C8A

HH a, b, C, d, X(K + 5), S31, &HFFFA3942

HH d, a, b, C, X(K + 8), S32, &H8771F681

HH C, d, a, b, X(K + 11), S33, &H6D9D6122

HH b, C, d, a, X(K + 14), S34, &HFDE5380C

HH a, b, C, d, X(K + 1), S31, &HA4BEEA44

HH d, a, b, C, X(K + 4), S32, &H4BDECFA9

HH C, d, a, b, X(K + 7), S33, &HF6BB4B60

HH b, C, d, a, X(K + 10), S34, &HBEBFBC70

HH a, b, C, d, X(K + 13), S31, &H289B7EC6

HH d, a, b, C, X(K + 0), S32, &HEAA127FA

HH C, d, a, b, X(K + 3), S33, &HD4EF3085

HH b, C, d, a, X(K + 6), S34, &H4881D05

HH a, b, C, d, X(K + 9), S31, &HD9D4D039

HH d, a, b, C, X(K + 12), S32, &HE6DB99E5

HH C, d, a, b, X(K + 15), S33, &H1FA27CF8

HH b, C, d, a, X(K + 2), S34, &HC4AC5665

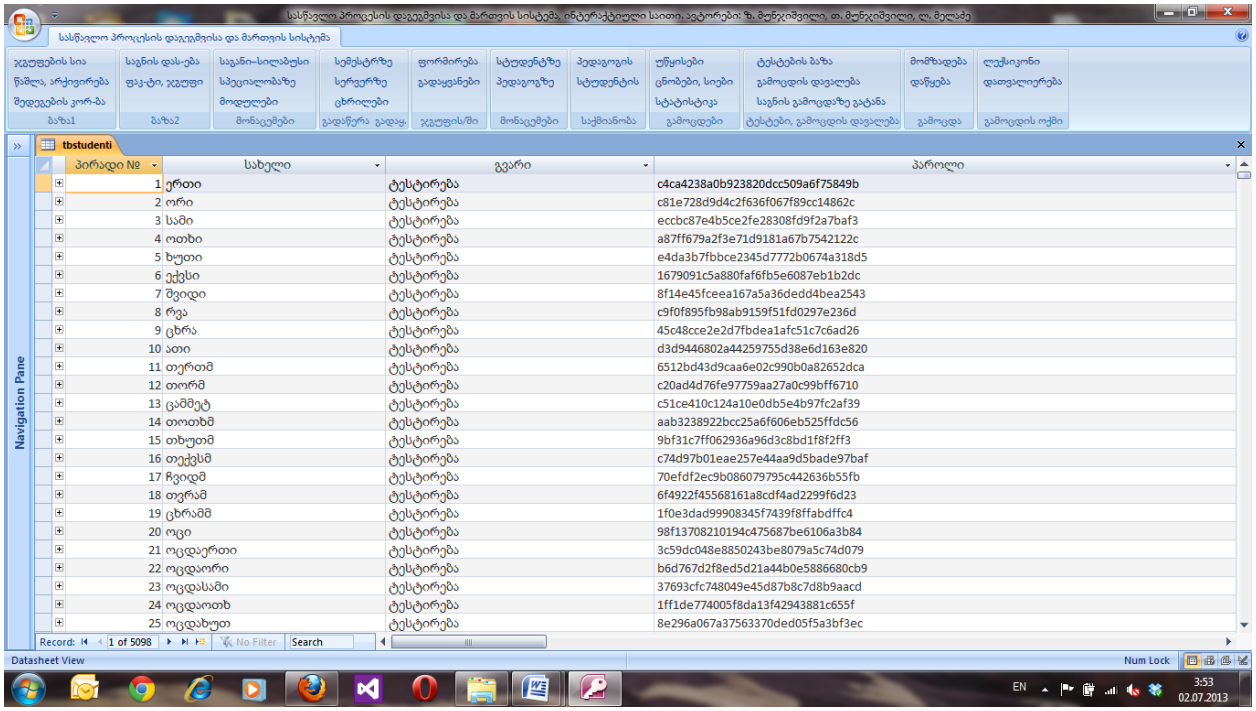
II a, b, C, d, X(K + 0), S41, &HF4292244

II d, a, b, C, X(K + 7), S42, &H432AFF97

II C, d, a, b, X(K + 14), S43, &HAB9423A7

```
        II b, C, d, a, X(K + 5), S44, &HFC93A039
        II a, b, C, d, X(K + 12), S41, &H655B59C3
        II d, a, b, C, X(K + 3), S42, &H8F0CCC92
        II C, d, a, b, X(K + 10), S43, &HFFEFF47D
        II b, C, d, a, X(K + 1), S44, &H85845DD1
        II a, b, C, d, X(K + 8), S41, &H6FA87E4F
        II d, a, b, C, X(K + 15), S42, &HFE2CE6E0
        II C, d, a, b, X(K + 6), S43, &HA3014314
        II b, C, d, a, X(K + 13), S44, &H4E0811A1
        II a, b, C, d, X(K + 4), S41, &HF7537E82
        II d, a, b, C, X(K + 11), S42, &HBD3AF235
        II C, d, a, b, X(K + 2), S43, &H2AD7D2BB
        II b, C, d, a, X(K + 9), S44, &HEB86D391

        a = AddUnsigned(a, AA)
        b = AddUnsigned(b, BB)
        C = AddUnsigned(C, CC)
        d = AddUnsigned(d, DD)
    Next
    MD5 = LCase(WordToHex(a) & WordToHex(b) & WordToHex(C) & WordToHex(d))
End Function
```

ჩვენ სისტემაშიც პაროლების დასაშიფრად გამოიყენება MD5-ის ალგორითმი. ყოველი სტუდენტის პაროლი დაშიფრულია და ბაზაში ინახება დაშიფრული ტექსტი (იხ.სურ.11). სტუდენტის ყოველი ავტორიზავიისას ხდება მის მიერ შეყვანილი კოდის დაშიფვრა და დაშიფრული ტექსტის შედარება არსებულ პაროლთან. დამთხევის შემთხვევაში სტუდენტი გადის ავტორიზაციას , წინააღმდეგ შემთხვევაში გამოდის შესაბამისი ტექსტი.



სურ. #10 md5-ით დაშიფრული პაროლები ცხრილში.

სისტემის ადმინისტრატორმა მოახდინოს შედეგის გაყალბება.

არსებობს პრობლემა რომ სისტემის ადმინისტრატორმა მოახდინოს შედეგების გაყალბება. ვთქვათ სტუდენტმა ჩააბარა გამოცდა და მიიღო შესაბამისი ქულა. სისტემის ადმინისტრატორს შეუძლია გააღოს ცხრილი და მოახდინოს მასში ინფორმაციის ჩასწორება . მაგალითად გადააკეთოს სტუდენტის მიერ მიღებული ქულა, ან შეცვალოს სტუდენტის მიერ დაფიქსირებული პასუხი , რაც გამოიწვევს შედეგის გაყალბებას. უნდა მოვახდინოთ ასეტი პრობლემის თავიდან აცილება. ასეთი პრობლემის თავიდან ასაცილებლად ვიყენებთ ციფრულ ხელმოწერას .

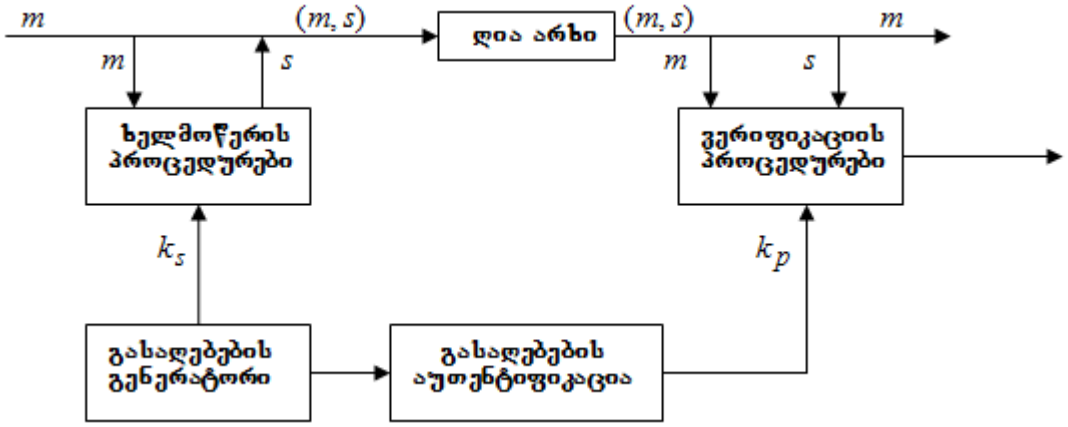
ციფრული სახით წარმოდგენილი ინფორმაციის შემთხვევაში ციფრული ხელმოწერა ასრულებს იგივე ფუნქციას, რასაც ქაღალდზე წარმოდგენილი დოკუმენტის შემთხვევაში ჩვეულებრივი ხელმოწერა და ბეჭედი. მრავალ ქვეყანაში მიღებულია სპეციალური კანონი და

დოკუმენტს ასეთი ხელმოწერით გააჩნია იურიდიული ძალა. ეს ნიშნავს, რომ თუ თქვენ ზუსტად იცავთ იმ განაწესსპროტოკოლს, რომლის შესრულებაც კანონით განსაზღვრულია მოცემულ ქვეყანაში, ყველა სადავო საკითხი, დაკავშირებული ციფრულ ხელმოწერასთან, შეიძლება განხილული იქნას იურიდიულ ასპექტში (სასამართლოში).

ციფრული ხელმოწერა (Digital Signature – DS) არის პროცედურა, რომელიც საშუალებას იძლევა:

1. მოვახდინოთ გადაცემული შეტყობნების 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.
2. ხელმოწერის ავტორმა ვერ უნდა შეძლოს უარყოს თავისი ხელმოწერა. ამ თვისებას უწოდებენ **ხელმოწერის ვერუარყოფის თვისებას.**

ციფრული ხელმოწრა სრულდება შემდეგი სქემის საშუალებით (იხ. სურ. #11).



სურ. #11 ციფრული ხელმოწერის სქემა

ფორმალურად, ღია გასაღებიანი ციფრული ხელმოწერის სქემა შეიძლება აღვწეროთ

შემდეგი პარამეტრებით (M, K_s, K_p, A, V) სადაც:

1. M არის ხელმოსაწერი შეტყობინებების სიმრავლე;
2. K_s არის საიდუმლო გასაღებების სიმრავლე;
3. K_p არის ღია გასაღებების სიმრავლე;
4. A არის იმ ალგორითმების სიმრავლე, რომლითაც ხდება შეტყობინების ხელმოწერა;
5. V არის იმ ალგორითმების სიმრავლე, რომლითაც ხდება ხელმოწერის ვერიფიკაცია;
6. ამასთან $\forall a \in A$ და $\forall v \in V$ სრულდება ტოლობა $a(m, k_s) = v(m, k_p)$.

ციფრული ხელმოწერის ძირითადი ტიპები. არსებობს ციფრული ხელმოწერის ორი განსხვავებული ფორმა:

1. ციფრული ხელმოწერა, როგორც შეტყობინების დამატება (with appendix)
2. ციფრული ხელმოწერა შეტყობინების აღდგენის (message recivery).

პირველ შემთხვევაში ციფრული ხელმოწერა გადაეცემა ადრესატს შეტყობინებასთან ერთად, ანუ შეტყობინებას ემატება გარკვეული რაოდენობის ბიტები (როგორც წესი ბოლოში), რაც წარმოადგენს მოცემული შეტყობინების ციფრულ ხელმოწერას. მეორე შემთხვევაში კი ადრესტს გადაეცემა ხელმოწერა და გასაღები, რომლითაც მოხდა შეტყობინების ხელმოწერა და ადრესატმა უნდა აღადგინოს შეტყობინება. თუ გადასაცემია დაშიფრული შეტყობინება, შესაძლებელია ჯერ მოეწეროს ხელი შეტყობინებას და შემდეგ დაიშიფროს, ან პირიქით, ჯერ დაიშიფროს და შემდეგ მოეწეროს ხელი.

ციფრული ხელმოწერის ფალსიფიკაცია. თუ მოწინააღმდეგეს შეუძლია შექმნას მოცემული შეტყობინების შესაბამისი კორექტული ხელმოწერა, ითვლება, რომ მას შეუძლია გააყალბოს ხელმოწერა. არსებობს გაყალბების სხვადასხვა შემთხვევები:

- სისტემის სრული გატეხვა: მოწინააღმდეგე ახერხებს ღია გასაღებიდან მიიღოს საიდუმლო გასაღები და გააყალბოს ნებისმიერი ხელმოწერა;
- უნივერსალური გაყალბება: მოწინააღმდეგეს შეუძლია ღია გასაღების საშუალებით შექმნას ისეთი ალგორითმი, რომლის საშუალებითაც გააყალბებს ნებისმიერ ხელმოწერას;
- შერჩევითი გაყალბება: ღია გასაღების გამოყენებით მოწინააღმდეგს შეუძლია შექმნას ისეთი შეტყობინება, რომლის ხელმოწერაც მას შეუძლია გააყალბოს;

- ეგზისტენციალური გაყალბება: მოწინააღმდეგეს ღია გასაღების საშუალებით შეუძლია შექმნას წყვილი: შეტყობინება – ყალბი ხელმოწერა, მაგრამ არ შეუძლია აკონტროლოს მიღებული შეტყობინების შინაარსი.

აქ ჩამოთვლილი კრიტერიუმები დალაგებულია სიძლიერის მიხედით, ანუ თუ შესაძლებელია სისტემის სრული გატეხვა, ეს ნიშნავს, რომ ციფრული ხელმოწერის სისტემა ძალიან სუსტია და პირიქით, თუ შეუძლებელია ეგზისტენციალური გაყალბებაც კი – სისტემა ძალიან ძლიერია უსაფრთხოების თვალსაზრისით.

თავისთავად ცხადია, ამ დროს გათვალისწინებული უნდა იქნას თუ რა შესაძლებლობები გააჩნია მოწინააღმდეგს. შეზღუდულია თუ არა ის დროში და გამოთვლით საშუალებებში. შეუძლია თუ არა მიიღოს ხელმოწერის ნიმუშები ან გამოიყენოს შეკითხვები ორაკულისადმი და ასე შემდეგ.

ყველაზე მარტივი და გავრცელებული ინსტრუმენტი ციფრული ხელმოწერის არის RSA კრიპტოალგორითმი. თუმცა, არსებობს ათობით კრიპტოალგორითმები ციფრული ხელმოწერის. ჩვენ შემთხვევაში ვიყენებთ RSA-ს.

RSA-ს გამოყენება ციფრული ხელმოწერისთვის. სანამ ამერიკის სტანდარტების ინსტიტუტი (NIST) 1994 წელს მიიღებდა ახალ ხელმოწერის სტანდარტს, RSA წარმოადგენდა ფაქტობრივ სტანდარტს როგორც ამერიკაში, ასევე მთელს მსოფლიოში. ეს არაა გასაკვირი, რადგანაც RSA არის ერთერთი ყველაზე სანდო და ძლიერი ღია გასაღებიანი ალგორითმი (მიუხედავად იმისა, რომ მას გააჩნია ბევრი სუსტი წერტილი). RSA სისტემა შეიძლება გამოვიყენოთ შეტყობინების ხელმოსაწერად როგორც შეტყობინების დამატების სქემაში, ასევე შეტყობინების გამოთვლის სქემაში. RSA-ს გამოსაყენებლად ციფრული ხელმოწერისთვის, საჭიროა ჩავატაროთ შემდეგი პროცედურები:

მოსამზადებელი ეტაპი. ჩვენ შემთხვევაში კლიენტის როლში გვევლინება Access და სერვერის როლში MS SQL. სტუდენტი როდესაც აბარებს გამოცდას მისი დაფიქსირებული პასუხები იგზავნება კლიენტიდან სერვერზე. ჩვენი მიზანია ციფრული ხელმოწერა სწორედ ამ ტექსტს დავადოთ. დავუშვათ Access-ი გზავნის m შეტყობინებას . მან უნდა მოაწეროს ხელი ამ m შეტყობინებას. ამ მიზნით Access-ში შეირჩევა $n = p \cdot q$ მოდულის ფუძე და გამოითვლება ღია და საიდუმლო გასაღებების (e,d) წყვილი. Access-ი საჯაროდ (ქსელში) გადაუგზავნის (n,d) წყვილს ანუ ამ შემთხვევაში ღია გასაღებს დეშიფრაციას. დაშიფრვის გასაღები (n, e) , ისევე როგორც p-ს და q-ს მნიშვნელობები ინახება საიდუმლოდ.

ხელმოწერის ეტაპი. Access-ი დაშიფრავს შეტყობინებას $c \equiv m^e \pmod n$ და გადაუგზავნის c-ს SQL-ს.

ხელმოწერის შემოწმების (ვერიფიკაციის) ეტაპი. SQL-ი მისთვის ცნობილი დეშიფრაციის გასაღების (n, d) მიხედვით გამოთვლის $m \equiv c^d \pmod n$.

როგორ უნდა მოხდეს ასეთ შემთხვევაში ხელმოწერის შემოწმება? თუ ტექსტი დაწერილია რომელიმე ბუნებრივ ენაზე, მაშინ SQL მხოლოდ შეამოწმებს, რომ მიღებული ტექსტიც დაწერილია იმავე ენაზე, რაც რა თქმა უნდა არაა საკმარისი. თუ ჩვენ ვიყენებთ RSA-ს პირდაპირ ვარიანტს, მაშინ საუკეთესო გამოსავალი იქნება მივცეთ ღია ტექსტს ისეთი სტანდარტული ფორმა, რომელიც დაარწმუნებს SQL-ი რომ ტექსტში ცვლილებები არ მომხდარა. მაგალითად, დავუშვათ მოდულის სიგრძე ბიტებში არის k , m შეტყობინების კი t და t<k-32 . დავუმატოთ შეტყობინებას მარჯვნიდან იმდენი ნოლი, რომ მისი სიგრძე გახდეს 8-ს ჯერადი. ამის შემდეგ დავუმატოთ მარცხნიდან m(k-t)/8 ბაიტი. მივიღებთ სტრიქონს:

$$P = 00 \| 01 \| FF \| \ldots \| FF \| m$$

Access-ი ხელს აწერს ასეთი სახით წარმოდგენილ შეტყობინებას.SQL-ი მოახდენს მის დეშიფრაციას და შეამოწმებს მიღებული სტრიქონის ფორმას. მაგრამ თუ შეტყობინების სიგრძე მეტი იქნება მოდულის ფუძეზე, მაშინ მოგვიწევს დავყოთ შეტყობინება ბლოკებად და თითოეულ ბლოკს ცალცალკე მოვაწეროთ ხელი. მაგრამ ამ შემთხვევაში ჯერ ეს ერთი დაგვჭირდება ძალიან დიდი დრო და მეორეც გაჩნდება საშიშროება, რომ ჰაკერს შეუძლია შეცვალოს ზოგიერთი ბლოკები. ამიტომ უკეთესია, თუ გამოვიყენებთ ე.წ. MR (Message Recovery) შეტყობინების აღდგენის სქემას. ასეთი სქემის გამოყენების დროს ხდება შეტყობინების ჰეშირება და Access-ი ხელს აწერს მიღებულ სიდიდეს, SQL კი ღია ტექსტს აღადგენს მიღებული შეტყობინებიდან. მაგრამ ამ შემთხვევაში ჰეშ-ფუნქციის გამოთვლის ალგორითმი მედეგი უნდა იყოს "დაბადების დღეებით" შეტევის მიმართ და Access-იდან გადაგზავნის დროს უნდა მივიღოთ დამატებითი ზომები, რათა დარწმუნებული ვიყოთ, რომ ის ხელს აწერს ნამდვილად გადასაცემი შეტყობინების ჰეშ-ფუნქციას.

ვერიფიკაციის პროცედურა Sql-ზე

```
SELECT points,
       CONVERT(NVARCHAR(100),
       DecryptByAsymKey(AsymKey_ID('AsymmetricKey'),
              CipherText, N'VeryVeryStrongPassword')) AS Decrypted,
       CASE
          WHEN VerifySignedByAsymKey(AsymKey_Id('AsymmetricKey'),
                                     points, Signature) = 1
            THEN N'The data has not been changed.'
          ELSE N'The data has been modified!'
       END AS IntegrityCheck
FROM AsymmetricTemp;
```

RSA-ს საიმედოობა, რომელიც ეფუძნება დიდი რიცხვის ფაქტორიზაციის ამოცანას და რომელიც შემოწმებულია მრავალი წლის გამოცდილებით, გვაძლევს იმის საფუძველს, ჩავთვალოთ, რომ ციფრული ხელმოწერა, შესრულებული ამ ალგორითმით, იქნება საიმედო. მაგრამ სინამდვილეში RSA-ს პირდაპირი ვარიანტი ძალიან სუსტია იმ შეტევების მიმართ, რომლებიც დღეს არსებობს ღია გასაღებიან კრიპტოსისტემებზე, ამიტომ, ისევე, როგორც დაშიფრვის დროს არ გამოიყენება პირდაპირი ვარიანტი, ასევე ხელმოწერის შემთხვევაში ამ ვარიანტის პირდაპირი გამოყენება არ შეიძლება.

PKCS#1. v 2.1 სტანდარტი RSA-ს ციფრული ხელმოწერისთვის.

ხელმოწერის ეტაპი.

დავუშვათ, მოდულის ფუძეა n რიცხვი, რომლის ჩანაწერი ორობით სისტემაში შეადგენს k ბაიტს. ხელმოწერის შესასრულებლად Access-ი ატარებს შემდეგ პროცედურებს:

1. გამოთვლის შეტყობინების კრებსით კოდს ჰეშ-ფუნქციის გამოთვლის რომელიმე (მაგალითად MD-5) ალგორითმის საშუალებით;
2. დაშიფრავს მიღებულ კრებსით კოდს და ჰეშ-ფუნქციის გამოთვლის ალგორითმის იდენტიფიკატორს
3. მიღებულ სტრიქონს მარცხნიდან დაუმატებს ნულოვან ბაიტს და შემდეგ იმდენ FF ბაიტს (სულ მცირე რვას), რომ მიიღოს k-2 ბაიტის სიგრძის სტრიქონი, შემდეგ დაუმატებს 01 ბაიტს და მიიღებს k-1 სიგრძის სტრიქონს;
4. ეს სტრიქონი გარდაიქმნება მთელ რიცხვად;
5. Access-ი მოაწერს ხელს ამ რიცხვს ხელმოწერის პირდაპირი ალგორითმით;
6. მიღებული რიცხვი გარდაიქმნება k ბაიტის სიგრძის სტრიქონად და გადაეცემა ბექას.

ვერიფიკაციის ეტაპი.

1. მიღებულ სტრიქონს SQL-ი გადაიყვანს მთელ რიცხვში; თუ ეს რიცხვი მეტია მოდულის ფუძეზე, მაშინ გადადის მეორე პროცედურაზე, თუ არა მესამეზე;
2. SQL-ი გამოიყენებს პირდაპირი ვერიფიკაციის პროცედურას და მიიღებს ახალ რიცხვს;
3. გადაიყვანს ამ რიცხვს სტრიქონში;
4. SQL-ი ამოწმებს, რომ სტრიქონს ჰქონდეს შემდეგი სტრუქტურა

$$00 \| 01 \| FF...FF \| 00 \| D$$

5. SQL-ი მოახდენს D-ს დეშიფრაციას და მიიღებს შეტყობინების კრებსით კოდს და ჰეშ-ფუნქციის გამოთვლის ალგორითმის იდენტიფიკატორს; შეამოწმებს, რომ შესაძლებელი იყოს ჰეშ-ფუნქციის ამ ალგორითმის გამოყენება;
6. SQL-ი ითვლის მიღებული შეყობინების დაიჯესტს და ადარებს მიღებულ დაიჯესტთან.

დასკვნა

სამაგისტრო ნაშრომის თემატიკას წარმოადგენს კომპიუტერული სისტემების უსაფრთხოება. ახსნილია თუ როგორი კომბინირებული მეთოდები უნდა გამოვიყენოთ რომ დავიცვათ თავი შეტევებისგან. შექმნილია პროგრამული პროდუქტი „კიბერგამოცდა“ რომელიც დაპატენტებულია და წარმატებით, ყოველდღიურად, გამოიყენება სხვადასხვა უნივერსიტეტებში. მის საფუძველზე ნაჩვენებია თუ როგორ უნდა შევზღუდოთ მომხმარებელი რომ არ მოახდინოს პროგრამის გადაწერა და მისი ავტონომიურ რეჟიმში გამოყენება . ასევე როგორ დავიცვათ თავი რომ ვერ შეძლოს პროგრამული კოდის გატეხვა. ახსნილია და ნაჩვენებია ,პროგრამული კოდებით, კრიპტოგრაფიის თანამედროვე მეთოდები, რომლის საფუძველზე დაცულია და დაშიფრულია ბაზა. მოყვანილია ბაზაზე შეტევის მეთოდები და შეტევებისგან თავის დაცვის საშუალებები. ასევე მოყვანილია სიტუაციები თუ როგორ შეუძლია სისტემის ადმინისტრატორს გააყალბოს შედეგები და გასცეს კითხვები. მოყვანილია ასეთი პრობლემების თავდაცვის გზები. გამოყენებულია თანამედროვე კრიპტოგრაფიული ალგორითმები როგორიცაა ტექსტების დაშიფვრა **BLOWFISH**-ის ალგორითმით.პაროლების დაშიფვრა **MD-5** ალგორითმით. ციფრული ხელმოწერა **RSA**-ს ალგორითმით. ყველა ალგორითმი რეალიზებულია პროგრამირების ენა VBA-ზე და ალგორითმებს თან დაერთვის.ნაშრომის ძირითადი აზრი მდგომარეობს იმაში რომ საუკეთესო დაცვისთვის უნდა გამოვიყენოთ კომბინირებული მეთოდები როგორიცაა: იდენტიფიკაცია, პაროლები, ბაზის დაშიფრვა კრიფტოგრაფიის მეთოდებით და ალგორითმებით.

გამოყენებული ლიტერატურა:

1. ზ.ქოჩლაძე, თანამედროვე კრიპტოგრაფიის საფუძვლები (მასალა გადაგზავნილია უნივერსიტეტის გამომცემლობაში და წიგნი დაიბეჭდება 2013-წელს)
2. The Fourth International Conference "New Trends In Education: Research And Development" (2011) A.Meladze,“Knowledge Reveal And Assessment Electronic Systems Reliability And Safety“ http://gu.edu.ge/docs/d.2011/conference/programm%20international%20english%202011.pdf
3. The Fourth International Conference "New Trends In Education: Research And Development" (2011) T.Munjishvili, A.Meladze “Knowledge Reveal And Assessment Electronic System "CYBERTEST"” http://gu.edu.ge/docs/d.2011/conference/programm%20international%20english%202011.pdf
4. Санкт-Петербург. Труды северно-западного политехнического института. Издательство СЗТУ 2011
   A.Meladze, T.Munjishvili “Система электронного планирования и управления учебным процессом”
5. J. Buchmann Introduction to Cruptography Springer, 2004 http://ge.tt/8QhHvpG/v/52
6. M. Welshendbach Criptography in C and C++ , (2005) http://www.amazon.com/Cryptography-C-Michael-elschenbach/dp/1590595025#reader_1590595025
7. Dan Boneh (Stanford University) , Online Cryptography Courses, (2013) https://www.coursera.org/course/crypto
8. ზ.მუნჯიშვილი, ა.მელაძე, თ. მუნჯიშვილი. ცოდნის გამოვლენის ელექტრონული სისტემა, საქპატენტი, საავტორო მოწმობა №5222
9. ზ.მუნჯიშვილი, ა.მელაძე, თ. მუნჯიშვილი. ელექტრონული ჟურნალი. საქპატენტი, საავტორო მოწმობა №5223