ივანე ჯავახიშვილის თბილისის სახელმწიფო უნივერსიტეტის ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი

სამაგისტრო მიმართულება: ინფორმაციული ტექნოლოგიები

ავთანდილ შარაბიძე

პროდუქტის სიცოცხლის ციკლის და კორპორატიული პროდუქტიულობის გაუმჯობესება Qt-ის მეშვეობით.

Improvement of Product Lifecycle and Corporate Productivity with Qt.

ნაშრომი შესრულებულია ინფორმაციული ტექნოლოგიების მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ნაშრომის ხელმძღვანელი:

ასოცირებული პროფესორი, დოქტორი ლელა მირცხულავა

თბილისი, 2014 წელი

სარჩევი

1.ანოტაცია

ჩვენი მიზანია მოხდეს Qt გარემოს მოხმარების პრეზენტაცია პროდუქტის შექმნის სიცოცხლის ციკლში.ამისათვის შეირჩა პროექტი რომელსაც დიდი მნიშვნელობა აქვს კორპორატიულ საქმიანობის უსაფრთხოებაში.

ეს პროექტი დაეხმარება კორპორაციებს ფაილების უსაფრთხო მიმოცვლაში. უპირველესად საჭირო იყო შერჩეულიყო ალგორითმი ფაილების შიფრაციისათვის. არჩევანი შევაჩერეთ AES Rjndael მეთოდზე, რომელსაც დღეისათვის იყენებს პენტაგონი. მეტი თავდაცვისათვის დაგაწყდა გაკეთებულიყო პროგრამა რომელიც მოახდენდა ფაილების დეშიფრაციას და მათ განთავსებას ოპერატიულ მეხსიერებაში. რაც თავიდან აგავაცილებს გაშიფრული ფაილის მყარ დისკზე მოხვედრას და მოწინააღმდეგისათვის ინფორმაციის მოპოვებას საგრძნობლად გაართულებს. ამ მიდგომამ აუცილებელი გახადა ფაილების წასაკითხად შექმნილიყო მოდული, რომელიც ახდენს ფაილების წაკითხვას მეხსიერებიდან და მათ პრეზენტაციას მომხმარებლისათვის.

საბოლოო ჯამში მივიღეთ ორი პროგრამა:

- პრველი ახდენს ფაილის შიფრაცია, დეშიფრაციას, როგორს გრაფიკული ასევე არაგრაფიკული ინტერფეისით.
- მეორე პროგრამა ახდენს ფაილის დეშიფრაციას და პრეზენტაციის მოდულისთვის მის გადაცემას, რომელიც თავის მხრივ უზრუნველყოფს ბევრი პოპულარული ფაილური ფორმატის დამუშავებას და მომხმარებლისთვის შიგთავსის ხელმისაწვდომობას

Our aim is to present Qt framework in program's development life cycle. For that purposes was selected project, which plays great role in corporate business security. This project will help corporations in secure file exchanges.

First of all, we needed to choose an algorithm for files encoding. So, we stopped on AES Rjndael encoding method, which is mainly used by Pentagon. For additional security, was created program, which had to decode files and store them in computer Random Access Memory. It will avoid us from getting our decoded files on a hard drive and will make it much harder for opposite site to get sensitive information. That approach made necessity of creation file reading module, which responsibility is reading files from RAM and showing it to user.

In the end we got two programs:

- First one is for file encryption and decryption, with both graphical and non-graphical interface.

- Second program is decoding files and then passing it to module, which in turn has ability of formatting many popular file extentions and provides user with file content.

2.შესავალი

თანამედროვე სამყაროში ინფორმაციის მიმოცვლა კომპიუტერულ ქსელში ხდება დაუშიფრავი სახით, რაც იწვევს ინფორმაციის გაჟონვას და არასასურველ პრობლემებს. XXI საუკუნეში პროექტი რომელიც მუშაობს დაუშიფრავ ინფორმაციაზე კარგავს თავის ღირებულებას და აუცილებლობას წარმოადგენს მისი ჩანაცვლება. დაუშიფრავი ფაილი ქსელში წარმოადგენს მარტივად მისაღებ ნადავლს მოწინააღმდეგისათვის.

გამოყენებული ტექნოლოგიები:

- პროგრამის შესაქმნელად გამოყენებული იქნა კროსპლატფორმული C++ გარემო Qt 4.8.4.
- შიფრაციის სტანდარტი AES Rijndael რომელიც შევისწავლეთ კურსიდან „ინფორმაციის უსაფრთხოება და დაცვა“ .
- PDF ფაილებთან სამუშაოდ ბიბლიოთეკა Poppler, რომელის შესანიშნავად მუშაობს Qt გარემოში.
- მულტიმედია ფაილებთან სამუშაოდ ბიბლიოთეკა Phonon.
- AES Rijndael ალგორითმის რეალიზებისთვის თანამედროვე და მულტიფუნქციური ბიბლიოთეკა CryploPP 5.6.2.
- Plug-and-play აპლიკაციის (ე.წ. Portable) შესაქმნელად პროგრამა WinRAR.
- პროგრამის დოკუმენტაციისათვის პოპულარული პროექტის დოკუმენტაციის შემქმნელი Doxygen.

3. გამოყენებული ტექნოლოგიები

- ***Qt Framework***

Qt არის გრაფიკულ ინტერფეისიანი პროგრამების (GUI Graphical User Interface), ასევე არაგრაფიკული პროგრამების, როგორებიცაა Command-line tools და სერვერის კონსოლები, ერთ-ერთი საუკეთესო კროს პლატფორმული **ფრეიმორკი.**

Qt იყენებს სტანდარტულ C++ -ს, მაგრამ ასევე იყენებს სპეციალურ კოდის გენერატორს (Meta Object Compiler, შემოკლებით moc) რამოდენიმე მაკროსთან ერთად როლელიც ამდიდრებს ენას. მას ასევე შეუძლია გამოიყენოს სხვადასხვა პროგრამული ენა, language bindings -ის გამოყენებით. ის ეშვება უმრავლეს Desktop პლათფორმაზე და ძირითად მობილურ სისტემებზე. მას აქვს ვრცელი ინტერნაციონალიზაციის მხარდაჭერა, SQL მონაცემთა ბაზებთან წვდომა, XML პარსერი, ნაკადების მართვა, ქსელის მხარდაჭერა და კროს-პლატფორმული აპლიკაციის პროგრამირების ინტერფეისი (Application Programimng Interface - API) ფაილებთან სამუშაოდ.

Qt ხელმისაწვდომია კომერციული, GPL v3 და LGPL v2 ლიცენზიებით. ყველა მათგანს აქვს მრავალი კომპილატორის მხარდაჭერა, თანდართული აქვს GCC C++ კომპილატორი და Visual Studio Suite.

Qt-ს დეველოპმენტს უწევს კომპანია Digia , რომლის მფლობელობაში არის მისი სასაქონლე ნიშანი და Qt project-ის მმართველობა. Qt project არის ფინური კომპანია NOKIA -ს მიერ შემუშავებული, რომელმაც Qt შეიძინა მისი დამაარსებელი ნორვეგიული კომპანია Troltech - ისაგან. 2011 წელს როდესაც ნოკიამ უარი თქვა Symbian OS – ის მასიურ დეველოპმენტზე და კურსი აიღო Microsoft – ის მობილური პლატფორმის ინტეგრაციისკენ ახალი წარმოების სმარტფონებში, მან კომპანია მიყიდა Digia – ს. ამ უკანასკნელმა კი მიზნად დაისახა ფრეიმორკი მოერგო ისეთი პოპულარული სისტემებისათვის რაოგორიცაა Android, iOS და Windows Phone 8, ასევე ის აგრძელებს Desktop და ჩაშენებული სისტემების (embedded system) მიმართულებაზე მუშაობას. მიუხედავად ამისა ნოკია მაინც ინარჩუნებს ერთერთ წამყვან როლს ფრეიმორკის განვითარებაში.

პლატფორმები:

Qt მუშაობს შემდეგ პლატფორმებზე:

- Windows – Qt for Microsoft Windows XP, Vista, 7, 8, 8.1
- OS X - Qt for Apple OS X ; Cocoa-ზე აპლიკაციების მხარდაჭერით.
- X11 – Qt for X Window System (GNU/Linux, FreeBSD, HP-UX, Solaris, AIX, და ა.შ.)
- Embadded Linux – yland - Qt for embedded platforms.
- Wayaland – Qt for Wailand
- QNX/BlackBerry 10
- Android
- iOS

- VxWorks
- Windiws CE

Qt გამოიცემა სამი ფორმით

- **GUI Framework** – კომერციული დონის GUI გამოცემა ,ქსელის და ბაზების მხარდაჭერის გარეშე ("Desktop Light")
- **Full Framework** – სრული კომერციული გამოცემა
- **Open Source** – სრული ღია კოდი

Qt devetopment tools :

Qt creator არის კროსპლატფორმული IDE, რომელიც კომპილატორთან ერთად უზრუნველყოფს აპლიკაციის შექმნის მთელ life-cycle-ს. მასში თავმოყრილია ყველა საჭირო ხელსაწყო Qt აპლიკაციის შესაქმნელად.

Qt creator-ში არის ინტეგრირებული :

- C++ კოდის ედიტორი, აღჭურვილი დეველოპერისათვის მოსახერხებელი ფუნქციებით როგორიცაა : Smart Code Completion, გადაადგილება კოდში ობიექტების მიხედვით, შეცდომების აღმოჩენა და მონიშვნა პროგრამის კომპილაციამდე, კონტექსტური დამხმარე ფანჯარა.
- Qt debugger იძლევა საშუალებას დააკვირდე/მართო პროგრამის ელემენტები Run-time -ს დროს და დააფიქსირო შეცდომები ან სარისკო უბნები. მას აქვს შემდეგი დებაგერების მხარდაჭერა:
    - GNU Symbolic Debugger (GDB)
    - Microsoft Console Debugger (CDB)
    - internal JavaScript debugger
    - LLVM debugger (LLDB)

- qmake აგებს (build) პროექტს სხვადასხვა პლატფორმისთვის. Qt იძლევა საშუალებას ასევე გამოვიყენოთ სხვა ხელსაწყო როგორებიცაა Cmake ან Qbs.
- Qt Designer არის ხელსაწყო რომლითაც ხდება GUI აპლიკაციის ფორმების აგება მარტივი ფორმით დეველოპერს უწევს მხოლოდ ელემებტების Drag-and-Drop პრინციპით ელემენტების ფორმაზე დაყრა და გადანაწილება.
- Qt Linguist ხელსაწყო ახდენს პროგრამის ინტერნაციონალიზაციას. პროგრამის ინტერფეისის ენის შეცვლა შეიძლება დინამიურად . ამისათვის საჭიროა Qt Linguist -

ით მოვახდინოთ საჭირო ფრაზების თარგმნა სასურველ ენაზე. პროგრამაში სასურველი ფრაზების tr() ფუნქციაში ჩასმის შემდეგ Qt Linguist მოახდეს მათ ამოცნობას და შემოგვთავაზებს მათ თარგმნას.

- Qt Assistant არის დამხმარე ხელსაწყო რომელშიც თავმოყრილია დოკუმენტაქია Qt კლასების და ფუნქციების შესახებ.

- makeqpf (ქმნის pre-rendered ფონტებს ჩაშენებული მოწყიბილობებისთვის),

- Meta-Object Compiler (moc ქმნის meta-object ინფორმაციას QObject -ის ქვეკლასებზე)

- User Interface Comppiler (uic ქმნის c++ კოდს UI ფაილებიდან)

- Resource compiler.

<u>პროდუქტის Qt -ს. გამოყენება</u>

I Qt Desktop აპლიკაციებისათვის

Qt არის შესანიშნავი C++ გარემო (framewotk), კროს-პლატფორმული Desktop ვიზუალური აპლილაციების შესაქმნელად, მოსახერხებელი API-თ. პროგამის კოდი საერთოა ყველა პოპულარული სისტემისთვის: Linux, Windows და Mac OS X. საკმარისია მხოლოდ კოდის დაკომპილირება სისტემაზე და მიიღება მისთვის თავსებადი პროგრამა. ბევრი კომპანია Qt-ს იყენებს მხოლოდ ერთ პლატფორმაზე და ის აკმაყოფილებს დეველოპერების მოთხოვნებს. Qt აპლიკაცას ის სტილი რომელიც არჩეულია მიმდინარე პლატფორმაზე, ცერები, ფონტები ხმები და სხვა სამუშაო მაგიდის პარამეტრები. მაგრამ დეველოპერს შეუძლია შექმნას საკუთარი აპლიკაციის სტილი Qt-ს ძლიერი Style Engine -ს საშუალებით. კროს-პლატფორმულ GUI აპლიკაციას აქვს, მენიუების, კონტექსტური მენიუების, drad and drop და ხელსაწყოების პანელის მხარდაჭერა. მომხმარებილის ინტერფეისი შეიძლება შეიქმნას განსხვავებული გზებით. ტრადიციული სტილის შესანარჩუნებლად გამოიყენება Qt Widget library. უფრო თანამედროვე და მართვადი UI დიზაინისათვის მოსახერხებელია Qt Quick ტექნოლოგია. Qt Quick იძლევა საშუალებას თანამედროვე დიზაინის უკან ამუშაო C++ ის წარმადობის მქონე აპლიკაცია და მიიღო გასაოცარი შედეგი.4

Qt არის გარემო, რომელის გვაწვდის ხელსაწყოებს და შესაძლებლობას, არა მარტო UI ინტერფეისის შესაქმნელად, არამედ უკანა პლანზე ვამუშაოთ კრიტიკული ბიზნეს ლოგიკა C++ ის მეშვეობით. Qt SQL მოდული ამარტივებს მონაცემთა ბაზასთან დაკავშირებული აპლიკაციის შექმნას, ხოლო Qt Network მოდული აწვდის დეველოპერს კროს-პლატფორმულ ქსელურ კლასებს, რომელთაც გააჩნიათ სხვადასხვა პროტოკოლთან სამუშაო API.

II Qt for Embedded Development

Qt კომპანიებს საშუალებას აძლევს შექმნან აპლიკაციები რომლებიც იმუშავებენ სპეციფიკურ მოწყობილობებთან. ამისათვის მას გააჩნია 700 C++ კლასი. ეს კალასები მომხმარებელს თავაზობენ GUI დიზაინზე გაცილებით მეტ შესაძლებლობებს, როგორებიცაა XML, ქსელი, ნაკადები, SQL, IPC,SVG, ინტერნაციონალიზაცია და მულტმედია. Qt იძლევა სრულ გარემოს ჩაშენებულ მოწყობილობებთან სამუშაოდ.

1 **Install Qt Enterprise Embedded**

- Download & run installer
- Flash a memory card with a pre-built image to put the Boot to Qt Software Stack to your board

2 **Get Creative with Qt!**

- Make an astonishing user interface for your device with Qt Quick
- Utilize embedded power with Qt/C++ APIs
- Work effectively within Qt Creator IDE

3 **Deploy to Device**

- Connect device with USB or wireless
- Deploy to your board with one click from Qt Creator IDE
- Debug directly on the device or within the Boot to Qt Emulator

Qt Enterprise Embedded ჩაშენებული მოწყობილობის შექმნა არის უმარტივესი Qt Enterprise Embedded -ის საშუალებით . ის სრულ მხარდაჭერას იძლევა embedded Linux და embedded Android მოწყობილობისთვის აპლიკაციის შესაქმნელად.

III Qt for Mobile Development

ეს არის Qt-ს ძირეული მოდული, რომლის ძირითადი მომხლრებალი იყო Nokia, მაგრამ ნოკიას მიერ Symbian და MeeGo ოპერაციული სისტემების უარყოფის შემდეგ ეს მოდული თითქმის უფუნქციოდ დარჩა Qt5-ის გამოსვლამდე. ბოლო ვერსიაში რევოლუციური ცვლილება შეიტანეს Qt -ს დეველოპერებმა და შეძლეს გასულიყვნენ ყველაზე პოპულარულ სატელეფონო Mobile პლატფორმებზე: Android, iOS და Windows phone.მუშაობის პრინციპი აქაც იგივეა წერ ერთხელ აკომპილირებ ყველგან.

- ***შიფრაციის სტანდარტი AES Rijndael***

გავრცელებული შიფრაციის სტანდარტი (Advanced Encryption Standard ან AES) — სიმეტრიული შიფრაციის ალგორითმი, ამორჩეული 2000 წლის ოქტომბერში NIST-ის მიერ ამერიკის მთავრობის ორგანიზაციების ახალი სტანდარტული შიფრაციისათვის.

წარმომავლობა

წარმოიშვა 1997 წლის იანვრის საერთაშორისო კანდიდატურაზე წარდგენისას რომელმაც მიიღო 15 შეთავაზება. ამ 15 ალგორითმს შორის, 5 იქნა არჩეული მძლავრი შეფასებისთვის 1999 წლის აპრილში: MARS, RC6, Rijndael, Serpent, et Twofish. ამ შეფასების ბოლოს ეს იყო საბოლოოდ კანდიდატი Rijndael, თავისი ორი შემქმნელის სახელით Joan და Vincent Rijmen (ორივე ბელგიური წარმოშობის). ეს ორი კრიპტოგრაფიის ექპერტი, უკვე იყვნენ სხვა ალგორითმის ავტორები: Square. AES არის Rijndael-ის ქვესიმრავლე: ის მუშაობს მხოლოდ 128 ბიტიან ბლოკებთან მაშინ როცა Rijndael წარადგენს ბლოკების ზომებს და 32-ის ჯერად გასაღებებს (128-ს და 256 შორის).

AES ცვლის DES-ს (სტანდარტად მიჩნეული 1970 წელს) რომლებიც დღეისთვის მოძველებულია, რადგან ის იყენებდა მხოლოდ 56 ბიტიან გასაღებებს AES იყო ადაპტირებული NIST-ის მიერ (სტანდარტების და ტექნოლოგიების საერთაშორისო ინსტიტუტი) 2001 წელს. უფრო მეტიც, მისი გამოყენება ძალიან პრაქტიკულია რადგან ის იყენებს ცოტა მეხსიერებას და არ არის დაფუძნებული Feistel-ის სქემაზე, მისი სირთულე არაა მისაღები და უფრო ადვილია დასანერგად.

მოინახულეთ მთავარი სტატია ამ თემაზე: გავრცელებული შიფრაციის სტანდარტი (Advanced Encryption Standard process)

<u>ფუნქციონირება</u>

ალგორითმი იღებს დასაწყისში 128 ბიტიან (16 ბაიტი) ბლოკს, გასაღები მოიცავს 128, 192 ან 256 ბიტს. დასაწყისში 16 ბაიტი არის გადაადგილებული წინასწარ განსაზღვრული ცხრილის მიხედვით. ეს ბაიტები განთავსებულია 4*4 ელემენტიან მატრიცაში და მისი ხაზები ექვემდებარება როტაციას მარჯვნივ. ინკრემენტი როტაციისთვის იცვლება მისი ხაზის ნომრის მიხედვით. შემდეგ მატრიცაზე გამოყენებულია წრფივი გარდაქმნა, ის მოიცავს მატრიცის თითოეული ელემენტის ორობით ნამრავლს დამხმარე მატრიციდან გამომდინარე პოლინომებით, ეს ნამრავლი დაფუძნებულია სპეციალურ წესებზე GF(2^8) მიხედვით (Galois-ის ჯგუფი ან სრული ტანი). წრფივი გარდაქმნა გიქმნის საუკეთესო გარემოს (ბიტების გავრცელება სტრუქტურაში) მრავალ ჯერზე.

საბოლოოდდ, XOR ორ მარტიცას შორის საშვალებას გაძლევს მიიღო დამატებითი მატრიცა. ეს განსხვავებული ოპერაციები გამეორებულია მრავალჯერ და განსაზღვრავენ "ჯერს". 128, 192 ან 256-იანი გასაღებისთვის AES საჭიროებს 10,12 ან 14 ჯერს.

<u>თავდასხმები</u>

AES ჯერ არ ყოფილა გატეხილი რჩება მხოლოდ ამომწურავი ძებნა. Rijndael იქნა შექმნილი მეთოდების კლასიკურად ჩამოყალიბებისთვის, როგორც წრფივი კრიპტოანალიზიან ძალიან ძნელი დიფერენციალობა

<u>თავდასხმები გამარტივებულ ვერსიებზე</u>

თავდასხმა არსებობს AES-ის გამარტივებულ ვერსიებზე. Niels Ferguson-მა და მისმა ჯგუფმა AES-ის 128 ბიტიან 7 რიგიან ვერსიაზე წამოაყენეს თავდასხმა. მსგავსი თავდასხმა ტეხავს 192 ან 256-იან AES-ს. 256-იანი AES ტყდება დამატებითი საშვალებებით 9 ბიჯზე დაყვანის შემტხვევაში. მართლაც ეს უკანასკნელი თავდასხმა ეყრდნობა "related-keys" პრინციპს. ესეთ თავდასხმაში გასაღები ხდება საიდუმლო მაგრამ შემტევს შეუძლია ცვლილებების განხორციელება გასაღებზე და ტექსტების დაშიფვრა. მაშასადამე მას შეუძლია შეცვალოს გასაღები და თვალი ადევნოს AES-ის შედეგს.

<u>თავდასხმები სრულ ვერსიებზე</u>

რამდენიმე ჯგუფმა განაცხადა AES-ის გატეხის შესახებ მაგრამ გადამოცმების შემდეგ დადგინდა რომ ეს არ შეესაბამებოდა სიმართლეს. თუმცა მრავალმა მეცნიერმა სააშკარაოზე გამოიტანა აჟღებრული თავდასხმების შესაძლებლობები კერძოდ XL თავდასხმა და გაუმჯობესებული ვარიანტი XSL. ეს თავდასხმები გახდა მათი მათი ცარუმეტებლობის მიზეზი და მათი შესაძლებლობები ვერ იქნა სრულად გამოვლენილი. XSL იძახებს heurist-ის ანალიზს რომლის ცარუმატებლობა არ არის სისტემატიური. უფრო მეტიც ისენი არინ არაპრაქტიკულნი რადგან XSL ითხოვს სულ ცოტა 2^87 ოპერაციას სახელდობრ 2^100 რამდენიმე შემთხვევაში. პრინციპი მდგომარეობს equation-ების დამყარებაში რომლებიც აერთებენ შესასვლელ გამოსასვლელებს და სისტემის ამოხსნაში რომელიც ითხოვს სულ ცოტა 8.000 უცნობ და 1.600 equation 128 ბიტისთვის. ამ სისტემის გაანალიზება ამ დროისთვის არის შეუძლებელი განსასაზღვრად. AES მიჩნეულია საიმედოდ ფორმალური დამტკიცების გარეშე XSL-ის მსგავსი თავდასხმების მკაფიოობაზე.

<u>NSA-ს რეკომენდაციები</u>

NSA განაცხადა რომ ყველა ფინალისტი რომლებმაც მონაცილეობა მიიღო AES-ის კონკურსში მიჩნეულ იქნებოდა საიმედოდ და რომ ისენი იყვნენ საკმარისად კომპაქტურნი ამერიკის მთავრობის არა კლასიფიცირებული მონაცემების დასაშიფრად. 2003წ ლის ივნისში ამერიკის მთავრობამ განაცხადა: "ყველა ზომის AES-ის ალგორითმის გასაღებების არქიტექტურა და სიგრძე (128, 192 და 256) საკმარისია კლასიფიცირებული დოკუმენტების დასაცავად საიდუმლო დონეზე. "TOP SECRET" დონე ითხოვს 192 ან 256 ბიტიან გასაღებებს. AES-ის ჩანერგვა სისტემის დაცვისთვის განკუთვნილ მექანიზმებში და/ან საერთაშორისო თავდაცვასთან დაკავშირებულ დოკუმენტებში უნდა გაკეთდეს ანალიზი ან სერთიფიცირება NSA-s მიერ მათ გამოყენებამდე" ხელმისაცვდომია დეპეშის თარგმანის ორიგინალი

სხვადასხვა თავდასხმები

ეს უკანასკნელი ფრაზა იღებს მთელ თავის მნიშვნელობას მაშინ როცა ჩვენ ვიცით რომ თავდასხმები შესაძლებელია წარუმატებელ სისტემებზე. მართლაც ჩვენ შეგვიძლია ანალიზი გავუკეთოთ ელექტრულ მოხმარებას ან კიდევ შიფრაციისთვის საჭირო დროს ამ ტიპის თავდასხმა იწვევს უმთავრესად "შავი ყუთის" სისტემებს რომლებშიც მუდმივი საიდუმლო გასაღები დაშიფრულია სისტემებში და გამოყენებულია მრავალი მესიჯის დასაშიფრად. ინახულეთ განცხადება მოხმარების ანალიზი (კრიპტოგრაფია).

შეგვიძლია ვიკითხოთ რატომ არ იყო არც ერთი ოფიციალური კონკურსი ცჰატარებული AES-ზე თავდასხმების კვლევების ხელშეცყობის დახმარებისთვის. ამ სფეროში ასიმეტრიული შიფრაციის მეთოდი RSA ჯილდოვდება მრავალ დიდა ანაზღაურებელ კონკურსზე.

- ***PDF დოკუმენტების წამკითხველი Poppler***

Qt გარემოში PDF ფაილის ჩვენება შესაძლებელია ბიბლიოთეკა Popler -ის საშუალებით, რომელსაც გააჩნია დახვეწილი და მრავალფუნქციური API ფაილებთან სამუშაოდ. Popplr არის Xpdf PDF წამკითველის განშტოება, რომელიც ვრცელდება „GNU General Public License“ ლიცენზიით. ბიბლიოთეკა შექმნილია ისე რომ მარტივი შესაძლებელი იყოს მისი ინტეგრაცია Qt აპლიკაციაში.

Poppler -ის საშუალებით შესაძლებელია დოკუმენტების წაკითხვა ყველა ცნობილ პლატფორმაზე. მაგრამ Qt-ს სტანდარტულ პაკეტში ის არ არის ჩაშენებული. მისი დაყენება შესძლებელია კოდან დაბილდვის დროს სპეციალური ოპციის მითითებით.

- ***აუდიო მოდული Phonon***

Phonon არის KDE4 -ის მულტიმედიური გარემო, ისეთივე როგორიცაა ბიბლიოთეკა aRts. Phonon შეიქმნა KDE-ს მიერ როგორც დამოუკიდებელი გარემო, მას არ აქვს კავშირი ისეთ ცნობილ გარემოებთან როგორებიც არიან GStreamer ან xine. მას გააჩნია მარტივი და სტაბლური API. გარემო არ არის შექმნილი მხოლოდ Unix სისტემისათვის, ის წარმატებით გამოიყენება სქვა პლატფორმებზეც მაგალითად Microsoft windows და mac OS X.

Phonon არ არის გამიზნული, რათა შეასრულოს ყველა მულტიმედიური ფუნქია მაგრამ გვაძლევს მარტივ გზას მივაღწიოთ ყველა იმ ფნქციონალურობას რაც საჭიროა მედია აპლიკაციისთვის.

მაგალითად ფაილის დასაკრავად საჭირო C++ ს მხოლოდ 3 ხაზი:

```
MediaObject *media = new MediaObject(this); /*მედია ობიექტის შექმნა*/
media->setCurrentSource("/home/username/music/filename.ogg"); /*გზის მითითება*/
media->play();/*დაკვრა*/
```

ფონონს საშუალება აქვს იმუშაოს სხვადასხვა მულტიმედიურ backend - თან, ე.წ. მოტორებთან.

- Unix : xine, Gstreamer, VLC et MPlayer
- Windows : DirectX, VLC et MPlayer ;
- Mac OS X : Quicktime ;

Phonon ს შეუძლია მყისიერად შეცვალოს მულტიმედია გარემო, საკმარისია მხოლოდ პროცესის დაპაუზება და გაგრძელება.

ფრეიმორკს აქვს სხვადასხვა მოწყობილობებთან მუშაბის და მათი კონტროლის შესაძლებლობა. მაგალითად ყურსასმენში გაუშვა VoIP სატელეფონო საუბარი და ამავდროულად დინამიკში ჩართული იყოს მუსიკა. ამ შემთხვევაში აუდიო ნაკადები არ გადკვეთავენ ერთმანეთს.

Qt იყენებს Phonon-ს Qt 4.4 ვერსიის შემდეგ მულტიმედია ფუნქსიონალურობისათვის. GStreamer, Quicktime და DirectX მოტორების მხარდაჭერით.

- ## *Crypto++ (ცნობილია ასევე როგორც: CryptoPP, libcryptopp, libcripto++)*

Crypto++ არის კრიპტოგრაფიული ალგორითმების და სქემების უფასო C++ ბიბლიოთეკა ღია კოდით. ბიბლიოთეკა ფართოდ გამოიყენება აკადემიურ, სტუდენტურ, ღია კოდით, არაკომერციულ და კომერციულ პროექტებში. ის შეიქმნა 1995 წელს და თავსებადია უმრავლეს ოპერაციულ სისტემასთან, როგორც 32 ასევე 64 ბიტიანი არქიტექტურებისათვის. ის თავსებადია: Android (STLport), apple(Mac OS X, iOS), BSD, Linux, Colaris, Windows, Windows Phone და Windows RT. პროექტს აქვს სხვადასხვა კომპილაციის IDE-ს მხარდაჭერა: Borland Turbo C++, Borland C++ Builder, Clang, CodeWarrior Pro, GCC(ასევე Apple GCC), ICC(Intel C++ Compiler), Microsoft Visual C/C++ და Sun Studio.

ალგორითმები

Crypto++ გვაწვდის სრულ კრიპტოგრაფიულ იმპლემენტაციებს და გვთავაზობს პოპულარულ და ასევე იშვიათად გამოყენებად სქემებს, რომლებიც მოყვანილია ცხრილში:

Primitive or Operation	**Algorithms or Implementations**
Pseudorandom number generators	LCG, KDF2, Blum Blum Shub, ANSI X9.17
High speed stream ciphers	Panama, SOSEMANUK, Salsa20, XSalsa20
AES and AES candidates	Rijndael (AES selection), RC6, MARS, Twofish, Serpent, CAST-256
Other block ciphers	IDEA, Triple-DES (DES-EDE2 and DES-

	EDE3), Camellia, SEED, RC5, Blowfish, TEA, XTEA, Skipjack, SHACAL-2
Block cipher modes of operation	ECB, CBC, CTS, CFB, OFB, CTR
Authenticated encryption modes	CCM, GCM, EAX
Block ciphers padding schemes	PKCS#5, PKCS#7, Zeros, One and zeros
Message authentication codes	VMAC, HMAC, CMAC, CBC-MAC, DMAC, Two-Track-MAC
Cryptographic hash function	SHA-1, SHA-2 (SHA-224, SHA-256, SHA-384, and SHA-512), SHA-3, Tiger, WHIRLPOOL, RIPEMD (RIPEMD-128, RIPEMD-160, RIPEMD-256, and RIPEMD-320)
Password based key derivation functions	PBKDF1 and PBKDF2 from PKCS #5, PBKDF from PKCS #12 appendix B
Public-key cryptography	RSA, DSA, ElGamal, Nyberg-Rueppel (NR), Rabin-Williams (RW), LUC, LUCELG, DLIES (variants of DHAES), ESIGN
Padding schemes for public-key systems	PKCS#1 v2.0, OAEP, PSS, PSSR, IEEE P1363 EMSA2 and EMSA5
Key agreement schemes	Diffie-Hellman (DH), Unified Diffie-Hellman (DH2), Menezes-Qu-Vanstone (MQV), LUCDIF, XTR-DH
Elliptic curve cryptography	ECDSA, ECNR, ECIES, ECDH, ECMQV
Secret Sharing	Shamir's secret sharing scheme, Rabin's information dispersal algorithm (IDA)

ბიბლიოთეკის განვითარება მიმდინარეობს აქტიუტად, პერიუოდულად გამოდის მისი განახლებები და რელიზები. ბოლო განახლება Crypto++ 5.6.2 გამოვიდა 2013 წლის 20 თებერვალს.

- ## *Plug-and-play აპლიკაციის (ე.წ. Portable) შესაქმნა*

Qt აპლიკაციის მომზადება მომხმარებლისათვის მოსახერხებელ ფორმაში არ საჭიროებს C++ პროგრამირების ცოდნას. ამისათვის არსებული აპლიკაცია უნდა დავბილდოთ რელიზით და აპლიკაცია მოვაწყოთ Plug HYPERLINK "http://qt-project.org/doc/qt-4.8/tools-plugandpaint.html"& HYPERLINK "http://qt-project.org/doc/qt-4.8/tools-plugandpaint.html" Paint პრინციპით, რომელსაც ქვემოთ განვიხილავთ.

სტატიკური და ზიარი ბიბლიოთეკები

აპლიკაციის წარმოებაში მისაღებად შეგვიძლია გამოვიყენოთ ორი გზა

- სტატიკური დამისამართება
- ზიარი ბიბლიოთეკები

სტატიკური დამისამართება შედეგად გვაძლევს მხოლოდ ერთ გამშვებ ფაილს. დადებითი აქვს ის რომ Deploy უნდა გააკეთო მხოლოდ რამოდენიმე ფაილისაგან, ხოლო უარყოფითი მხარე არის დიდი ზომის გამშვები ფაილი, დამატებითი ბიბლიოთეკების შემთხვევაში საჭიროა მათი დამატებაც ერთი გამშვების პრინციპი ირღვევა.

ჩვენ შემთხვევაში საჭიროა კრიპტოგრაფიის ბიბლიოთეკის დამატება და მიზანშეწონილია ზიარი ბიბლიოთეკების გამოყენება, შემდეგ კი SFX (Self-extracting) არქივის შექმნა.

Windows სისტემაზე Plug-and-play აპლიკაციის შექმნის ერთ-ერთი მარტივი გზა არის SFX არქივის შექმნა. ამისთვის არსებობს Windows- ის ჩაშენებული ხელსაწყო მაგრამ, ის არ არის დახვეწილი, მაგალითად მას არ აქვს ქვედირექტორიების აღქმის უნარი. სხვადასხვა პროგრამის ტესტირების შემდეგ პროგრამა WinRAR –მა მოგვცა საუკეთესო შედეგი SFX არქივის შესაქმნელად.

- ***Doxygen***

პროექტის სრულყოფილი სახის მისაცემად აუცილებელია მისი სრული და თვალსაჩინო დოკუმენტირება. ამისათვის გამოყენებული იქნა ყველაზეპოპულარული და მრავალფუნქციური დოკუმენტირების ხელსაწყო Doxygen.

Doxygen არის დოკუმენტაციის გენერატორი, პროგრამის კოდიდან დოკუმენტაციის შექმნის შესაძლებლობით, რომელიც ვრცელდება თავისუფალი მოხმარების ლიცენზიით. დოკუმენტაციის გენერირებისათვის საჭიროა გრამატიკის ცოდნა და რომლის გამოყენებითაც უნდა მოხდეს კოდის კომენტირება.

Doxygen-ს შეუძლია ანალიზი გაუკეთოს შემდეგი ენების კოდებს: C++, Objective C, C#, PHP, Java, Python, IDL, Fortran, VHDL, Tcl და D. დოკუმენტაციის გენერირება შეიძლბეა რამოდენიმე ფორმატში: HTML, LaTeX, RTF, PostScript, PDF ჰიპერლინკებით და XML

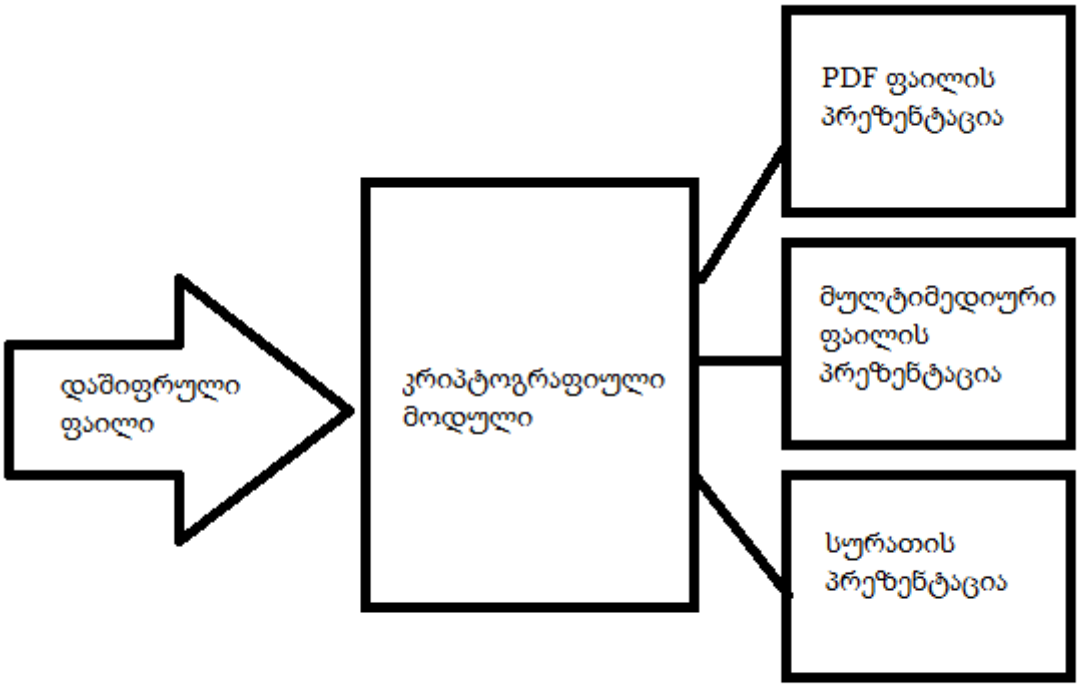
მაგალითი:

```
/**
* კლასი Time.
* @author test
*/
class Time {

    /**
    * კონსტრუკტორი.
    * დამატებითი აღწერა
    * @param timemillis დრო 1970 წლის 1 იანვრიდან
    */
    Time(int timemillis) {
        ...
    }

    /**
    * ფუნხქია იღებს მიმდინარე დროს.
    * @return result კონვერტირებული დრო
    */
    static Time now() {
        ...
       return result ;
    }
}
```

DoxyWizard არის Doxygen -ის კონფიგურირების გრაფიკული ხელსაწყო.

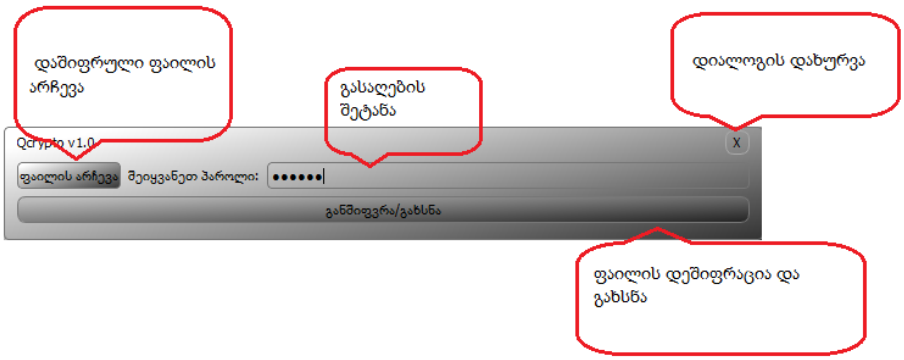
4. პროგრამის აღწერა

პროგრამის სტრუქტურა შემდეგნაირად გამოიყუტება:



დაშიფრული ფაილის არჩევას, პაროლის შეყვანას და დეშიფრაციას უზრუნველყოფს კლასი PasswordDialog.

ეს კლასი არის Qdialog-კლასის შვილობილი და მის დიალოგს აქვს შემდეგი ინტერფეისი:



კლასის კოდი:

passworddialog.h

```
#ifndef PASSWORDDIALOG_H
#define PASSWORDDIALOG_H

#include <QFileDialog>

#include "ui_passworddialog.h"
#include "crypto.h"

/**
 * @brief PasswordDialog კლასი ტვირთავს საწყის დიალოგს
 *
 * კლასში დამუშავებულია ფაილის გახსნა, პაროლის შეყვანა და შიგთავსის პერზენტაციისათვის გამზადება
 */
class PasswordDialog : public QDialog, private Ui::PasswordDialog
{
    Q_OBJECT
public:
    /**
     * @brief PasswordDialog კლასის კონსტრუქტორი
     *
     * @param parent მშობელი კლასის ობიექტი
     */
    explicit PasswordDialog(QWidget *parent = 0);
    /**
     * @brief ფუნქციას გადაეცემა ფაილის მისამართი და აბრუნებს მის შიგთავსს
     *
     * @param url ფაილის მისამართი
     * @return QByteArray  ფაილის შიგთავსი
     */
    QByteArray * getResult(QString url);
    QString m_url; /**< TODO */
private slots:
    /**
     * @brief დეშიფრაციის ღილაკზე მიმაგრებული სლოტი
     *
     */
    void on_pushButton_clicked();
    /**
```

```
     * @brief სლოტი ააქტიურებს დეშიფრაციის ღილაკს
     *
     * @param str
     */
    void setOkEnabled(QString str);
    /**
     * @brief დიალოგის დახურვის ღილაკი
     *
     */
    void on_c_clicked();
    /**
     * @brief ფაილის გახსნის ღილაკი
     *
     */
    void on_open_file_clicked();

private:
    QFile * encoded_file; /**< TODO */
    Crypto *m_crypto; /**< TODO */
    QByteArray * encripted_content; /**< TODO */
    QByteArray * result;  /**< TODO */
signals:
    /**
     * @brief პროგრამის დახურვის სიგნალი
     *
     */
    void kill();
};


#endif // PASSWORDDIALOG_H
```

passworddialog.cpp

```
#include "passworddialog.h"
#include <QDebug>
#include <QLayout>
PasswordDialog::PasswordDialog(QWidget *parent) :
    QDialog(parent)
{
    setupUi(this);

    setWindowFlags(Qt::ToolTip);
    m_crypto  = new Crypto();
    pushButton->setEnabled(false);
    lineEdit->setEchoMode(QLineEdit::Password);

connect(lineEdit,SIGNAL(textChanged(QString)),this,SLOT(setOkEnabled(QString)
));
}

void PasswordDialog::on_pushButton_clicked()
{

    encoded_file = new QFile(m_url);
    encoded_file->open(QIODevice::ReadOnly);
    encripted_content = new QByteArray(encoded_file->readAll());
    delete encoded_file;
```

```
    result = m_crypto->decryptAes(lineEdit->text().toStdString().data(),
encripted_content);
    delete encripted_content;
    close();
}



QByteArray * PasswordDialog::getResult(QString url)
{
    m_url=url;
    exec();
    return result;
}

void PasswordDialog::setOkEnabled(QString str)
{
    if(str == NULL) pushButton->setDisabled(true);
    if(str != NULL) pushButton->setEnabled(true);
}



void PasswordDialog::on_c_clicked()
{
    kill();
    close();
}

void PasswordDialog::on_open_file_clicked()
{
    m_url = QFileDialog::getOpenFileName(this,trUtf8("აირჩიეთ დასაშიფრი
ფაილი"));
    qDebug()<<m_url;
}
```

PasswordDialog კლასიდან მოქმედება გადადის MainWindow კლასში სადაც ხდება ფაიკის განშიფრული შიგთავსის გადაცემა, ფაილის ფორმატის დადგენა და პრეზენტაციის დიალოგზე გადაგზავნა

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QFileInfo>
#include <QMainWindow>

#include "passworddialog.h"
#include "pdfviewer.h"
#include "mediaplayer.h"
#include "imageviewer.h"

#include "ui_mainwindow.h"
```

```
/**
 * @brief
 *
 */
class MainWindow : public QMainWindow, private Ui::MainWindow
{
    Q_OBJECT

public:
    /**
     * @brief
     *
     * @param parent
     */
    explicit MainWindow(QWidget *parent = 0);
private:
    QFile           *downloadFileLink; /**< TODO */
    QFileInfo       *m_fileInfo; /**< TODO */
    QString          m_fileExtention; /**< TODO */
    QString          m_url; /**< TODO */

    PasswordDialog  *m_passwordDilaog; /**< TODO */
    PDFViewer       *m_pdfViewer; /**< TODO */
    MediaPlayer     *m_mediaPlayer; /**< TODO */
    ImageViewer     *m_imageViewer; /**< TODO */
    MainWindow      *m_mainWindow; /**< TODO */

private slots:
    /**
     * @brief
     *
     */
    void kill();

};
#endif // MAINWINDOW_H
```

mainwindow.cpp

```
#include "mainwindow.h"
#include <QFile>

/**
 * @brief
 *
 * @param parent
 */
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent)
{
   // setupUi(this);
  //  m_mainWindow = new MainWindow();

    m_passwordDilaog       = new PasswordDialog();
    QByteArray * file_data = m_passwordDilaog->getResult(m_url);
    m_fileInfo             = new QFileInfo(m_passwordDilaog->m_url);
    m_fileExtention        = m_fileInfo->suffix().toUpper();
    qDebug()<<m_fileExtention<<__LINE__;
```

```
    connect(m_passwordDilaog,SIGNAL(kill()),this,SLOT(kill()));


    /**
    *PDF დოკუმენტის დამუშავება იმ შემთხვევაში თუ ბმული მიუთითებს შესაბამისი
ფორმატის ფაილს
    **/
    if( m_fileExtention == "PDF")
    {
        m_pdfViewer = new PDFViewer(this,file_data);
        m_pdfViewer->exec();
        exit(0);
    }
    /**
     *სურათის დამუშავება იმ შემთხვევაში თუ ბმული მიუთითებს შესაბამისი ფორმატის
ფაილს
    **/
    if((QStringList() << "BMP" << "GIF" << "ICO" << "JPEG" << "JPG" << "MNG"
        << "PBM" << "PGM" << "PNG" << "PPM" << "SVG" << "SVGZ"<< "TGA"
        << "TIF" << "TIFF"<< "XBM" << "XPM").contains(m_fileExtention))
    {
        m_imageViewer = new ImageViewer(this,file_data);
        delete m_passwordDilaog;
        m_imageViewer->exec();
        exit(0);
    }

    /**
    *მულტიმედიურო დოკუმენტის დამუშავება იმ შემთხვევაში თუ ბმული მიუთითებს
შესაბამისი ფორმატის ფაილს
    **/
    if(m_fileExtention == "MP3" || m_fileExtention == "MP4" ||
            m_fileExtention == "AVI" || m_fileExtention == "WAV")
    {
        m_mediaPlayer =new MediaPlayer(this,file_data,m_fileExtention);
        m_mediaPlayer->exec();
        exit(0);
    }

}


/**
 * @brief
 *
 */
void MainWindow::kill()
{
    qDebug()<<"kill";
    delete m_passwordDilaog;
    delete this;
    exit(0);
}
```

MainWindow კლასიდან ხდება სამი პრეზენტაციის დიალოგიდან ერთერთის გამოძახება ფაილის ტიპის მიხედვით.

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QFileInfo>
#include <QMainWindow>

#include "passworddialog.h"
#include "pdfviewer.h"
#include "mediaplayer.h"
#include "imageviewer.h"

#include "ui_mainwindow.h"

/**
 * @brief
 *
 */
class MainWindow : public QMainWindow, private Ui::MainWindow
{
    Q_OBJECT

public:
    /**
     * @brief
     *
     * @param parent
     */
    explicit MainWindow(QWidget *parent = 0);
private:
    QFile           *downloadFileLink; /**< TODO */
    QFileInfo       *m_fileInfo; /**< TODO */
    QString          m_fileExtention; /**< TODO */
    QString          m_url; /**< TODO */

    PasswordDialog  *m_passwordDilaog; /**< TODO */
    PDFViewer       *m_pdfViewer; /**< TODO */
    MediaPlayer     *m_mediaPlayer; /**< TODO */
    ImageViewer     *m_imageViewer; /**< TODO */
    MainWindow      *m_mainWindow; /**< TODO */

private slots:
    /**
     * @brief
     *
     */
    void kill();

};
#endif // MAINWINDOW_H
```

mainwindow.cpp

```
#include "mainwindow.h"
#include <QFile>

/**
 * @brief
 *
```

```
 * @param parent
 */
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent)
{
   // setupUi(this);
  //  m_mainWindow = new MainWindow();

    m_passwordDilaog       = new PasswordDialog();
    QByteArray * file_data = m_passwordDilaog->getResult(m_url);
    m_fileInfo             = new QFileInfo(m_passwordDilaog->m_url);
    m_fileExtention        = m_fileInfo->suffix().toUpper();
    qDebug()<<m_fileExtention<<__LINE__;
    connect(m_passwordDilaog,SIGNAL(kill()),this,SLOT(kill()));


    /**
    *PDF დოკუმენტის დამუშავება იმ შემთხვევაში თუ ბმული მიუთითებს შესაბამისი
ფორმატის ფაილს
    **/
    if( m_fileExtention == "PDF")
    {
        m_pdfViewer = new PDFViewer(this,file_data);
        m_pdfViewer->exec();
        exit(0);
    }
    /**
     *სურათის დამუშავება იმ შემთხვევაში თუ ბმული მიუთითებს შესაბამისი ფორმატის
ფაილს
    **/
    if((QStringList() << "BMP" << "GIF" << "ICO" << "JPEG" << "JPG" << "MNG"
        << "PBM" << "PGM" << "PNG" << "PPM" << "SVG" << "SVGZ"<< "TGA"
        << "TIF" << "TIFF"<< "XBM" << "XPM").contains(m_fileExtention))
    {
        m_imageViewer = new ImageViewer(this,file_data);
        delete m_passwordDilaog;
        m_imageViewer->exec();
        exit(0);
    }

    /**
    *მულტიმედიურო დოკუმენტის დამუშავება იმ შემთხვევაში თუ ბმული მიუთითებს
შესაბამისი ფორმატის ფაილს
    **/
    if(m_fileExtention == "MP3" || m_fileExtention == "MP4" ||
            m_fileExtention == "AVI" || m_fileExtention == "WAV")
    {
        m_mediaPlayer =new MediaPlayer(this,file_data,m_fileExtention);
        m_mediaPlayer->exec();
        exit(0);
    }

}


/**
 * @brief
 *
 */
void MainWindow::kill()
{
    qDebug()<<"kill";
```

```
    delete m_passwordDilaog;
    delete this;
    exit(0);
}
```

ფაილების პრეზენტაციას უზრუნველყოფენ კლასები:

ImageViewer

PdfViewer


Dialog
First
Previous
4
Next
Last
AES Rijndael
ალგორითმის შექმნის ისტორია
შიფრაციის ფორმა
მომხმარებლები

MediaPlayer

imageviewer.h

```
#ifndef IMAGEVIEWER_H
#define IMAGEVIEWER_H

#include <QLabel>
#include <QScrollArea>
#include <QLayout>
#include "ui_imageviewer.h"


/**
 * @brief
 * ImageViewer კლასი ახდენს სურათების პრეზენტაციას
 *
 *
 */
class ImageViewer : public QDialog, private Ui::ImageViewer
{
    Q_OBJECT
private:
   double scaleFactor; /**< TODO */
public:
    /**
     * @brief ImageViewer კლასის კონსტრუქტორი
     *
     * @param parent მშობელი კლასის ობიექტზე მიმთითებელი
     */
    explicit ImageViewer(QWidget *parent = 0);
    /**
```

```
     * @brief ImageViewer კლასის კონსტრუქტორი
     * ImageViewer   კლასის კონსტრუქტორი რომელიც იღებს დასამუშავებელი ფაილის ბინარულ შიგთავსს
     * @param parent მშობელი კლასის ობიექტზე მიმთითებელი
     * @param by     დასამუშავებელი ფაილის ბინარული შიგთავსი
     */
    ImageViewer(QWidget *parent = 0, QByteArray *by = 0);
private slots:
     /**
      * @brief  zoomIN ღილაკის დასამუშავებელი სლოტი.
      * სლოტი ახდენს სურათის გადიდებას.
      */
     void on_BT_zoomIN_clicked();
     /**
      * @brief zoomOut ღილაკის დასამუშავებელი სლოტი
      * სლოტი ახდენს სურათის დაპატარავებას.
      */
     void on_BT_zoomOut_clicked();
private:
     /**
      * @brief ფუნქცია ახდენს სურათის მასშტაბის ცვლილებას
      * მასშტაბის ცვლილება ხდება factor ცვლადის ცვლილებით, მომატების შემთხვევაში იზრდება სურათის მასშტაბი მოკლებისას პირიქით
      * @param factor სურათის მასშტაბის კოეფიციენტი
      */
     void scaleImage(double factor);
     /**
      * @brief   scrollBar - ის დეგულირება
      *
      * @param scrollBar
      * @param factor
      */
     void adjustScrollBar(QScrollBar *scrollBar, double factor);
};

#endif // IMAGEVIEWER_H
```

imageviewer.cpp

```
#include "imageviewer.h"
#include <QDebug>
#include <QScrollBar>

ImageViewer::ImageViewer(QWidget *parent) :
    QDialog(parent)
{
    setupUi(this);
}
ImageViewer::ImageViewer(QWidget *parent, QByteArray *by) :
    QDialog(parent)
{
    setupUi(this);
    setWindowFlags(parent->windowFlags());

    imageLabel->setBackgroundRole(QPalette::Base);
    imageLabel->setSizePolicy(QSizePolicy::Ignored, QSizePolicy::Ignored);
```

```
    imageLabel->setScaledContents(true);
    scrollArea->setBackgroundRole(QPalette::Dark);
    QImage image;
    image.loadFromData(*by);
    imageLabel->setPixmap(QPixmap::fromImage(image));
    imageLabel->adjustSize();
}
void ImageViewer::on_BT_zoomIN_clicked()
{
    scaleImage(1.25);
}
void ImageViewer::on_BT_zoomOut_clicked()
{
    scaleImage(0.8);
}
void ImageViewer::scaleImage(double factor)
{
    Q_ASSERT(imageLabel->pixmap());
    scaleFactor *= factor;
    if (scaleFactor>3.0) scaleFactor = 3.0;
    if (scaleFactor<0.333) scaleFactor = 0.33;
    imageLabel->resize(scaleFactor * imageLabel->pixmap()->size());
  //  adjustScrollBar(scrollArea->horizontalScrollBar(), factor);
  //  adjustScrollBar(scrollArea->verticalScrollBar(), factor);
}
void ImageViewer::adjustScrollBar(QScrollBar *scrollBar, double factor)
{
    scrollBar->setValue(int(factor * scrollBar->value()
                            + ((factor - 1) * scrollBar->pageStep()/2)));
}
```

pdfviewer.h

```
#ifndef PDFVIEWER_H
#define PDFVIEWER_H

#include "ui_pdfviewer.h"
#include "poppler-qt4.h"
/**
 * @brief The PDFViewer class
 *  კლასი უზრუნველყოფს გადაცემული PDF დოკუმენტის ჩვენებას
 */
class PDFViewer : public QDialog, private Ui::PDFViewer
{
    Q_OBJECT

public:
    explicit PDFViewer(QWidget *parent = 0);
    explicit PDFViewer(QWidget *parent = 0, QByteArray *pdfData = 0);
private:
    /**
     * @brief m_document პოპლერის PDF დოკუმენტი
```

```
     */
    Poppler::Document *m_document;
    /**
     * @brief m_pageIndex მიმდინარე გვერდი იინდექსი
     */
    qint32      m_pageIndex;
    /**
     * @brief m_pageCount დოკუმენტის გვერდების დამუშვება
     */
    qint32      m_pageCount;
    /**
     * @brief m_zoom      გვერდის ზომა
     */
    qreal       m_zoom;

private:
    /**
     * @brief showDocumentPage აჩვენებს PDF დოკუმენტის გვერდს
     * @param indexOf          საჩვენებელი გვერდის ინდექსი
     */
    void showDocumentPage(qint32 indexOf);
    /**
     * @brief setDocument       ქმნის PDF დოკუმენტს
     * @param FileData          დოკუმენტის შიგთავსი
     */
    void setDocument(QByteArray * FileData);

private slots:
    /**
     * @brief on_PB_First_clicked       აჩვენებს დოკუმენტის პირველ გვერდს
     *
     */
    void on_PB_First_clicked();
    /**
     * @brief on_PB_Previous_clicked    აჩვენებს დოკუმენტის წინა გვერდს
     *
     */
    void on_PB_Previous_clicked();
    /**
     * @brief on_PB_Next_clicked        აჩვენებს დოკუმენტის შემდეგ გვერდს
     *
     */
    void on_PB_Next_clicked();
    /**
     * @brief on_BT_Last_clicked        აჩვენებს დოკუმენტის ბოლო გვერსდ
     */
    void on_BT_Last_clicked();
    /**
     * @brief on_spinBox_valueChanged   საჩვენებელი დოკუმენტის გვერდის spinBox-
ით არჩევა
     *
     * @param arg1
     */
    void on_spinBox_valueChanged(int arg1);
    /**
     * @brief on_PB_zoomIn_clicked      გვერდის ზომის მომატება
     *
     */
    void on_PB_zoomIn_clicked();
    /**
     * @brief on_PB_zoomUot_clicked     გვერდის ზომის დაკლება
```

```
     *
     */
    void on_PB_zoomUot_clicked();
    /**
     * @brief on_ScrollBar_NextPage_valueChanged ScrollBar-ის მეშვეობით
დოკუმენტის დათვალიერება
     *
     * @param value გვერდის ნომერი
     */
    void on_ScrollBar_NextPage_valueChanged(int value);
    /**
     * @brief on_ScrollBar_NextPage_sliderPressed           სლაიდერზე დაჭერის
ეფექტის დამუშავება
     *
     */
    void on_ScrollBar_NextPage_sliderPressed();
    /**
     * @brief on_ScrollBar_NextPage_sliderReleased          სლაიდერზე აშვების
ეფექტის დამუშავება
     *
     */
    void on_ScrollBar_NextPage_sliderReleased();


};

#endif // PDFVIEWER_H
```

pdfviewer.cpp

```
#include "pdfviewer.h"
#include <QDebug>

PDFViewer::PDFViewer(QWidget *parent) :
    QDialog(parent)
{
    setupUi(this);
}
PDFViewer::PDFViewer(QWidget *parent, QByteArray *pdfData) :
    QDialog(parent)
{
    setupUi(this);


connect(ScrollBar_NextPage,SIGNAL(valueChanged(int)),spinBox,SLOT(setValue(in
t)));

connect(spinBox,SIGNAL(valueChanged(int)),ScrollBar_NextPage,SLOT(setValue(in
t)));


    setWindowFlags(parent->windowFlags());
    setDocument(pdfData);
    label_PDF->setBackgroundRole(QPalette::Base);
    label_PDF->setScaledContents(true);
}
void PDFViewer::showDocumentPage(qint32 indexOf)
```

```
{

    label_PDF->clear();
    update();
    label_PDF->setPixmap(
    QPixmap::fromImage(m_document->page(indexOf-1)->renderToImage(
                       physicalDpiX()*m_zoom,
                       physicalDpiY()*m_zoom)));
    label_PDF->update();
}

void PDFViewer::setDocument(QByteArray * FileData)
{
    m_document = Poppler::Document::loadFromData(*FileData);
    m_document->setRenderBackend(Poppler::Document::SplashBackend);
    m_document->setRenderHint(Poppler::Document::TextAntialiasing);
    m_pageCount = m_document->numPages();

    spinBox->setMaximum(m_pageCount);

    ScrollBar_NextPage->setRange(1,m_pageCount);
    spinBox->setRange(1,m_pageCount);


    m_zoom = 0.8;
    m_pageIndex = 1;
    showDocumentPage(m_pageIndex);

}

void PDFViewer::on_PB_First_clicked()
{
    spinBox->setValue(0);
}

void PDFViewer::on_PB_Previous_clicked()
{
    spinBox->setValue(spinBox->value()-1);
}
void PDFViewer::on_PB_Next_clicked()
{
    spinBox->setValue(spinBox->value()+1);
}
void PDFViewer::on_BT_Last_clicked()
{
    spinBox->setValue(spinBox->maximum());

}
void PDFViewer::on_spinBox_valueChanged(int arg1)
{
    m_pageIndex=arg1;
    showDocumentPage(m_pageIndex);
    label_PDF->update();
}
void PDFViewer::on_PB_zoomIn_clicked()
{
    m_zoom>2 ? m_zoom: m_zoom+=0.2;
    showDocumentPage(m_pageIndex);
}
void PDFViewer::on_PB_zoomUot_clicked()
{
    m_zoom<=0.2?m_zoom : m_zoom-=0.2;
    showDocumentPage(m_pageIndex);
```

```
}
void PDFViewer::on_ScrollBar_NextPage_valueChanged(int value)
{
    m_pageIndex = value;
    showDocumentPage(m_pageIndex);
}

void PDFViewer::on_ScrollBar_NextPage_sliderPressed()
{
    m_zoom=0.4;
    showDocumentPage(m_pageIndex);

}

void PDFViewer::on_ScrollBar_NextPage_sliderReleased()
{
    m_zoom=0.8;
    showDocumentPage(m_pageIndex);
}
```

mediaplayer.h

```
#ifndef MEDIAPLAYER_H
#define MEDIAPLAYER_H

#include <QTime>
#include <QBuffer>

#include "ui_mediaplayer.h"
#include "phonon/phonon"


/**
 * @brief MediaPlayer კლასი ახდენს მედია ფაილების პრეზენტაციას
 *
 */
class MediaPlayer : public QDialog, private Ui::MediaPlayer
{
    Q_OBJECT


public:
    /**
     * @brief
     *
     * @param parent
     */
    explicit MediaPlayer(QWidget *parent = 0);
     /**
      * @brief
      *
      * @param parent
      * @param by
      * @param Extention
      */
     MediaPlayer(QWidget *parent = 0, QByteArray * by = 0, QString Extention
= NULL);
private slots:
    /**
     * @brief
     *
     */
```

```
    void on_PB_Play_clicked();
    /**
     * @brief
     *
     */
    void updateTime();
    /**
     * @brief
     *
     * @param value
     */
    void on_volumSlider_valueChanged(int value);
    /**
     * @brief
     *
     */
    void seekFile();
    /**
     * @brief
     *
     * @param newstate
     * @param oldstate
     */
    void stateChanged(Phonon::State newstate, Phonon::State oldstate);
    /**
     * @brief
     *
     */
    void makeStyles();
    /**
     * @brief
     *
     * @param time
     * @return QString
     */
    QString calculateTime(int time);

private:
    QBuffer *m_buffer; /**< TODO */

};

#endif // MEDIAPLAYER_H
```

mediaplayer.cpp

```
#include "mediaplayer.h"
#include <QDebug>


MediaPlayer::MediaPlayer(QWidget *parent) :
    QDialog(parent)
{
    setupUi(this);
}

MediaPlayer::MediaPlayer(QWidget *parent, QByteArray *by,QString Extention) :
    QDialog(parent)
{
    setupUi(this);
    setWindowFlags(parent->windowFlags());
```

```
    if(Extention == "MP3" || Extention == "WAV"){
        videoPlayer->hide();
setMaximumHeight(100);
    }

    videoPlayer->mediaObject()->setTickInterval(1);


    connect(videoPlayer->mediaObject(), SIGNAL(tick(qint64)), this,
SLOT(updateTime()));
    connect(videoPlayer->mediaObject(), SIGNAL(stateChanged(Phonon::State,
Phonon::State)), this, SLOT(stateChanged(Phonon::State, Phonon::State)));
    connect(videoPlayer->mediaObject(), SIGNAL(totalTimeChanged(qint64)),
this, SLOT(updateTime()));
    connect(videoPlayer->mediaObject(), SIGNAL(metaDataChanged()), this,
SLOT(updateTime()));
    connect(videoPlayer->mediaObject(), SIGNAL(finished()),videoPlayer-
>mediaObject(), SLOT(stop()));
    connect(timeSlider,SIGNAL(sliderPressed()),videoPlayer-
>mediaObject(),SLOT(pause()));
    connect(timeSlider,SIGNAL(sliderReleased()),videoPlayer-
>mediaObject(),SLOT(play()));
    connect(timeSlider, SIGNAL(sliderReleased()),this, SLOT(seekFile()));
    qDebug()<<"read media from ByteArray"<<by->length();

     QByteArray * mediaData =  by;
     m_buffer =new QBuffer(mediaData);
     videoPlayer->mediaObject()->setCurrentSource(m_buffer);
     volumSlider->setValue(50);
     videoPlayer->mediaObject()->play();
     lengthLabel->setText(calculateTime(videoPlayer->mediaObject()-
>totalTime()));
     timeSlider->setRange(0,videoPlayer->mediaObject()->totalTime());
     videoPlayer->mediaObject()->pause();
}


void MediaPlayer::on_PB_Play_clicked()
{
    lengthLabel->setText(calculateTime(videoPlayer->mediaObject()-
>totalTime()));
    timeSlider->setRange(0,videoPlayer->mediaObject()->totalTime());


    if (videoPlayer->mediaObject()->state() == Phonon::PlayingState)
        videoPlayer->pause();
    else if (videoPlayer->mediaObject()->state() == Phonon::PausedState ||
videoPlayer->mediaObject()->state() == Phonon::LoadingState || videoPlayer-
>mediaObject()->state() == Phonon::StoppedState)
        videoPlayer->play();

    qDebug()<<videoPlayer->mediaObject()->currentTime();

}

void MediaPlayer::updateTime()
{
    long len = videoPlayer->mediaObject()->totalTime();
    long pos = videoPlayer->mediaObject()->currentTime();
    timeSlider->setValue(pos);
    QString timeString;
    if (pos || len)
    {
```

```
            int sec = pos/1000;
            int min = sec/60;
            int hour = min/60;
            int msec = pos;

            QTime playTime(hour%60, min%60, sec%60, msec%1000);
            sec = len / 1000;
            min = sec / 60;
            hour = min / 60;
            msec = len;

            QTime stopTime(hour%60, min%60, sec%60, msec%1000);
            QString timeFormat = "hh:mm:ss";

            timeString = playTime.toString(timeFormat);
        }
        timeLabel->setText(timeString);
}

void MediaPlayer::on_volumSlider_valueChanged(int value)
{
    videoPlayer->setVolume((float)value/100.0);

}

void MediaPlayer::seekFile()
{
    videoPlayer->seek(timeSlider->value());
}

void MediaPlayer::stateChanged(Phonon::State newstate, Phonon::State
oldstate)
{
    if (oldstate == Phonon::LoadingState)
    {
        PB_Play->setIcon(style()->standardIcon(QStyle::SP_MediaPause));
        PB_Play->setToolTip("Pause");
    }

    if( newstate == Phonon::StoppedState || newstate == Phonon::PausedState)
    {
        PB_Play->setIcon(style()->standardIcon(QStyle::SP_MediaPlay));
        PB_Play->setToolTip("Play");
    }
    else if (newstate == Phonon::PlayingState)
    {
        PB_Play->setIcon(style()->standardIcon(QStyle::SP_MediaPause));
        PB_Play->setToolTip("Pause");
    }
    else if (newstate == Phonon::ErrorState)
    {
        //QMessageBox::warning(this, "Phonon Mediaplayer", videoPlayer-
>mediaObject()->errorString(), QMessageBox::Close);
        if (videoPlayer->mediaObject()->errorType() == Phonon::FatalError)
        {
            qDebug()<<"ErrorState";
        } else
        {
            videoPlayer->mediaObject()->pause();
        }
    }
}
```

```
QString MediaPlayer::calculateTime(int time)
{
    time = time /1000;
    int hours = time / 3600;
    int minutes = (time - (hours*3600)) / 60;
    int seconds = (time - (hours*3600) - (minutes*60)) % 60;
    QString hzero = "";
    QString mzero = "";
    QString szero = "";
    if (hours < 10) hzero = "0";
    if (minutes < 10) mzero = "0";
    if (seconds < 10) szero = "0";
    QString times = hzero+QString::number(hours)+":"+
mzero+QString::number(minutes)+":"+szero+QString::number(seconds);

    return times;

}

void MediaPlayer::makeStyles()
{
}
```

ფაილების შიფრაცია/დეშიფრაციისათვის პროექტში არის კლასი **Crypto**

crypto.h

```
#ifndef CRYPTO_H
#define CRYPTO_H

#include <cryptopp/dll.h>
#include <cryptopp/rijndael.h>

#include <QByteArray>
#include <cryptopp/md5.h>
#include <cryptopp/cryptlib.h>
#include <cryptopp/files.h>
#include <cryptopp/sha.h>
#include <cryptopp/seckey.h>
#include <cryptopp/modes.h>
#include <cryptopp/hex.h>


using namespace std;

using namespace CryptoPP;
/**
 * @brief  ტიპის განსაქღვრა რათა არ მოხდეს უთანხმოება Qt-ს ტიპებსა და სტანდერტულ
C++ ტიპებს შჰორის
 *
 */
typedef unsigned char byte;            // put in global namespace to avoid
ambiguity with other byte typedefs

/**
 * @brief Crypto კლასში აღწერილია პროგრამის კრიპტოგრაფიული მხარისათვის საჭირო
ფუნქციები
 *
 */
```

```
class Crypto
{
public:
    /**
     * @brief Crypto კლასისი კონსტრუქტორი
     *
     */
    Crypto();
    /**
     * @brief ფუნქცია შიფრავს მიღებულ ფაილს
     *
     * @param AES_KEY       გასაღები
     * @param inFileName    შემავალი ფაილი
     * @param outFileName   გამომავალი ფაილი
     */
    void encryptAes(const char * AES_KEY, const char * inFileName, const char
* outFileName);
    /**
     * @brief ფუნქცია ახდენს ფაილის დეშიფრაციას
     *
     * @param AES_KEY       გასაღები
     * @param inFileName    შემავალი ფაილი
     * @param outFileName   გამომავალი ფაილი
     * @return bool         ოპერაციის წარმატებულად შესრულების მაჩვენებელი
     */
    bool decryptAes(const char * AES_KEY, const char * inFileName, const char
* outFileName);
    /**
     * @brief ფუნქცია ახდენს ბინარული მასივის დეშიფრაციას
     *
     * @param AES_KEY       გასაღები
     * @param cryptedArray  საშიფრული მასივი
     * @return QByteArray   განშიფრული მასივი
     */
    QByteArray * decryptAes(const char * AES_KEY, QByteArray *cryptedArray);
    /**
     * @brief ფუნქცია ახდენს SHA2 ჰეშის გამოთვლას
     *
     * @param Password გასაღები
     * @return QString ჰეში
     */
    QString SHA256Digest(QString Password);


private:
    byte key[Rijndael::DEFAULT_KEYLENGTH], iv[Rijndael::BLOCKSIZE]; /**< TODO
*/
};

#endif // CRYPTO_H
```

crypto.cpp

```
#include "crypto.h"
#include <QDebug>


/**
```

```
 * @brief
 *
 */
Crypto::Crypto()
{
}
/**
 * @brief
 *
 * @param AES_KEY
 * @param cryptedArray
 * @return QByteArray
 */
QByteArray * Crypto::decryptAes(const char * AES_KEY, QByteArray
*cryptedArray)

{
    qDebug()<<"decryptAes form QByteArray ";
    if(cryptedArray->length()==0) {
        qDebug()<<"ERROR: encrypted  flie size is: 0";
        return NULL;

    }
    qDebug()<<"encrypted  flie size is: "<<cryptedArray-
>size()<<"\n"<<"Decrypt the encrypted text from a array";

    StringSource(AES_KEY, true,new HashFilter(*(new SHA256), new
ArraySink(key,Rijndael::DEFAULT_KEYLENGTH)) );
    memset( iv, 0x00, Rijndael::BLOCKSIZE );
    try
    {
        CBC_Mode<Rijndael>::Decryption Decryptor( key, sizeof(key), iv );


        byte *buf = (byte *)malloc(cryptedArray->size());

        StringSource((const unsigned char *)cryptedArray->data(),
                     cryptedArray->size(),true,
                     new StreamTransformationFilter ( Decryptor,  new
ArraySink(buf,cryptedArray->size())));


        QByteArray * result = new QByteArray((char *)buf, cryptedArray-
>size() );
        delete buf;
        cryptedArray->resize(0);
        return  result;

    }
    catch ( Exception& e)
    {
        qDebug()<<e.what();
    }
    catch (...)
    {
        qDebug()<<"Unknown Error";
    }
    return 0;
}

/**
 * @brief
 *
```

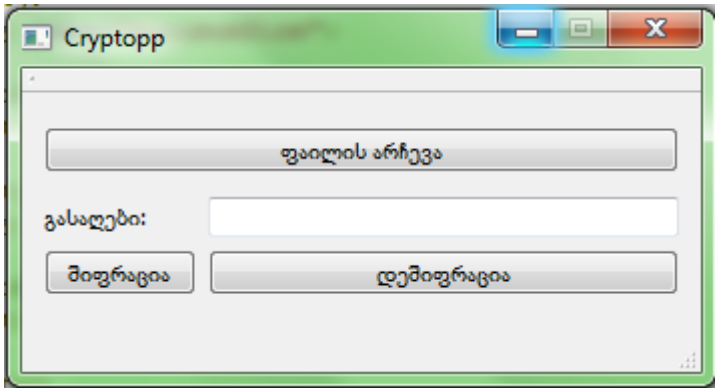
```
 * @param AES_KEY
 * @param inFileName
 * @param outFileName
 * @return bool
 */
bool Crypto::decryptAes(const char * AES_KEY, const char * inFileName, const
char * outFileName)
{ //deshifacia failidan
    StringSource( AES_KEY, true,new HashFilter(*(new SHA256), new
ArraySink(key,Rijndael::DEFAULT_KEYLENGTH)) );
    memset( iv, 0x00, Rijndael::BLOCKSIZE );
    try
    {
        CBC_Mode<Rijndael>::Decryption Decryptor( key, sizeof(key), iv );
        FileSource( inFileName, true, new StreamTransformationFilter(
Decryptor, new FileSink(outFileName) ) );
        return true;

    }
    catch ( Exception& e)
    {
        qDebug()<<e.what();
    }
    catch (...)
    {
        qDebug()<<"Unknown Error";
    }
}
/**
 * @brief
 *
 * @param AES_KEY
 * @param inFileName
 * @param outFileName
 */
void Crypto::encryptAes(const char* AES_KEY, const char * inFileName, const
char *outFileName)
{
    StringSource(AES_KEY, true, new HashFilter(*(new SHA256), new
ArraySink(key,Rijndael::DEFAULT_KEYLENGTH)) );
    memset( iv, 0x00, Rijndael::BLOCKSIZE );


    CryptoPP::CBC_Mode<Rijndael>::Encryption Encryptor( key, sizeof(key), iv
);
    FileSource(inFileName, true, new StreamTransformationFilter( Encryptor,
new FileSink(outFileName) ) );
}


/**
 * @brief
 *
 * @param Password
 * @return QString
 */
QString Crypto::SHA256Digest(QString Password)
{
    byte digest[ CryptoPP::SHA256::DIGESTSIZE ];
    std::string message = Password.toStdString();
    CryptoPP::SHA256().CalculateDigest( digest, (byte*) message.c_str(),
message.length());
```

```
    QString sha256, buf;
    for(int i =0; i<sizeof(digest); i++)
    {
        buf = QString::number(digest[i],16);
        if (buf.size()==1)
            buf.insert(0,"0");
        sha256.append(buf);
    }
    return sha256.toUpper();
}
```

მეორე დამხმარე პროგრამა ახდენს მხოლოდ ფაილის შიფრაციას, როგორც GUI ინტერფეისით ასევე Command line - დან.

დამხმარე პროგრამა:

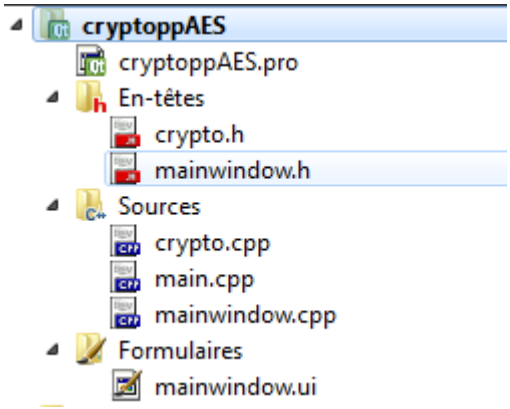
GUI ინტერფეისი



Console

SYNOPSIS: cryptoppAES [OPTION] [InputFILE] [OutputFILE]

-e --encrypt: file encryption
-d --decrypt: file decryption
-h --help: help



5. დასკვნა

დღევანდელი მდგომარეობით საქართველოს ბაზარზე ძირითადი მოთხოვნა არის .Net და Java აპლიკაციებე, რაც წარმოუდგენელია ევროპელი დამკვეთებისათვის , რომლებიც უპირატესობას ანიჭებენ არა პროგრამული უზრუნველყოფის სწტაფ შექმნას, არამედ მის წარმადობას და მოქნილობას. აქედან გამომდინარე ევროპულ ბაზარზე დიდი მოთხოვნა არის ისეთ ენებზე როგორიცაა C, C++, Python თუმცა უნდა აღინიშნოს Java რომელის ასევე მოთხოვნადია. პროექტის მიზანი იყო გაგვეცნო Qt პლატფორმამ როგორც C++ აპლიკაციების შექმნის ერთერთი საუკეთესო საშუალება. პროექტის მიმდინარეობისას აღმოვაჩინე ბევრი საინტერესო ნიუანსი ამ გარემოს შესახებ და შევისწავლე ახალი C++ კლასები. ასევე საფუღვლიანად შევისწავლე ისეთი დეტალები რომელთანაც წვდომა შეუძლებელია .Net დახურულ გარემოში. პროექტის დასრულებისას მივედი დასკვნამდე რომ Qt- ზე შეიძლება დაიწეროს სრულყოფილი და მრავალფუნქციური პროგრამები, რომლებიც იმუშავებენ ნებისმიერ პოპულარულ პლატფორმაზე და C++ -ის წყალობით ექნებათ გაცილებით მაღალი წარმადობა ვიდრე მის კონკურენტებს.

6. ბიბლიოგრაფია

http://qt.digia.com/

http://qt-project.org/

C++ GUI Programming with Qt4, 2nd edition. **Jasmin Blanchette Mark Summerfield**

http://www.cryptopp.com/wiki/Main_Page

http://freedesktop.org/wiki/Software/poppler/

http://techbase.kde.org/Development/Tutorials

http://www.stack.nl/~dimitri/doxygen/manual/index.html