

ДЪРЖАВЕН ИЗПИТ
ЗА ПРИДОБИВАНЕ НА ТРЕТА СТЕПЕН НА ПРОФЕСИОНАЛНА
КВАЛИФИКАЦИЯ – ЧАСТ ПО ТЕОРИЯ НА ПРОФЕСИЯТА

ДИПЛОМЕН ПРОЕКТ

Тема: „Разработване на десктоп приложение за за кодиране на
файлове“

Професия: 481020 „Системен програмист“

Специалност: 3810201 „Системно програмиране“

Дипломант: Виктор Дамянов Владинов 12А клас

Ръководител на дипломния проект: Божидар Василев Хинков

Подпис: _____ (дипломант)

Подпис: _____ (ръководител)

София 2024 г.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

Съдържание

1. Въведение.....	2
1.1 Задание за разработка.....	2
1.2 Анализ на заданието.....	2
2. Основна част.....	3
2.1 Избор на технология за потребителската част.....	3
2.1.1 C# (C Sharp).....	3
2.1.2 CLR Project на основата на .NET Framework.....	4
2.1.3 S.W.O.T – анализ.....	4
2.1.4 Сравнение с други технологии.....	5
2.2 Избор на технология за базата данни.....	6
2.2.1 СУБД – MySQL.....	7
2.2.2 Сравнение на релационна и нерелационна база от данни.....	7
2.2.3 Сравнение на MySQL с други технологии за бази данни.....	9
3. Софтуерен дизайн.....	11
3.1 Софтуерна архитектура.....	11
3.2 Предимства и недостатъци на MVC моделът.....	12
3.3 Файлова структура на проекта.....	13
3.4 Описание на взаимодействието с приложението.....	16
3.5 Дизайн на симетричният алгоритъм за криптиране.....	20
3.5.1 Криптиране.....	20
3.5.2 Декриптиране.....	28
3.6 Дизайн на базата данни.....	37
3.7 CRUD заявки.....	40
4. Заключение.....	47
4.1 Постигнати цели.....	47
4.2 Перспективи за развитие.....	47
5. Източници.....	49
6. Приложения.....	50



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

1. Въведение

1.1 Задание за разработка

„FileSaver“ представлява десктоп приложение, което предлага допълнителна сигурност върху съхранението на личните данни. Разработен е за операционната система “Windows”, чрез езикът C# и неговите библиотеки за разработка на десктоп софтуер, а за управление на базата данни - MySQL.

Десктоп приложението изисква вход, преди да предостави основните си функционалности, като всеки потребител може да се регистрира. Приложението предлага два типа потребителски профила – администратор и нормален потребител. Администраторът бива само един и той има допълнителни функционалности като например следене на действията на другите потребители, промяна на информация относно специфичен акаунт и изтриване потребителски профил.

1.2 Анализ на заданието

Главната идея на проекта „FileSaver“, е да предпази файловете съдържащи лична информация на компютъра на потребителя, като например снимки на лична карта, съдебни или други pdf документи съдържащи личните данни на потребителя и най-често срещаният - txt файл, в който потребителите си записват имената на акаунтите и паролите, за да не ги забравят. Тези файлове трябва да бъдат защитени от злонамерени хора, затова проектът дава решение на този проблем.

“FileSaver” използва мой личен прототип на алгоритъм за криптиране и декриптиране. Алгоритъмът е синхронен и е изграден от 4 вида трансформации. Чрез този алгоритъм съдържанието на избраният от потребителя файл се криптира с парола, въведена от потребителя. Криптираният файл може да се декодира, само и единствено чрез предоставяне на правилната парола. По този начин дори злонамерен човек да успее да проникне в компютъра на потребителя, най-важните файлове ще бъдат защитени.



2. Основна част

2.1 Избор на технология

2.1.1 C# (C Sharp)

Защо C#? C# е набрал изключителна популярност в света на разработката на десктоп приложения специфично за Windows. Този език е съвместим с CLR и е поддържан от Microsoft, като осигурява висока степен на сигурност и надеждност. Основа за C# са C++, Java и донякъде езици като Delphi, VB.NET и C. Той е проектиран да балансира мощност с възможност за бързо разработване.

Предимства:

- Лесен за изучаване и разработка.
- Интеграция с .NET Framework, предоставяйки голям брой библиотеки и инструменти.
- Висока производителност и сигурност.

Недостатъци:

- Ограничената поддръжка извън Windows операционната система.
- По-големи размери на приложението поради зависимостта от .NET Framework.



2.1.2 CLR Project на основата на .NET Framework

Защо съм избрал CLR Project? CLR(Common Language Runtime) проектът е на основата на .NET Framework дава възможността за използване на голямо количество библиотеки и компоненти.

Предимства:

- Управление на паметта, което намалява риска от утечки на памет.
- Широка библиотека с готови решения и инструменти.
- Лесна интеграция с други .NET технологии.

Недостатъци:

- По-големи размери на приложението, поради зависимостта от .NET Framework.
- Ограничения при разработка за операционни системи, различни от Windows.
- .NET Framework е безплатен за собствена употреба, но може да се наложат разходи за разпространение на приложението.

2.1.3 S.W.O.T – анализ

а) Силни страни на използваните технологии

- Висока производителност и сигурност.
- Лесна интеграция с други .NET технологии.
- Обширна общност и поддръжка.

б) Слаби страни на използваните технологии

- Ограничена поддръжка извън Windows операционната система.
- По-големи размери на приложението.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

с) Възможности

- Развитие на приложението с добавяне на нови функционалности и интеграция със API.
- Разширяване на потенциалните потребители чрез адаптация към различни операционни системи.

д) Заплахи

- Конкуренцията от други езици и технологии за разработка на десктоп приложения.
- Промени в .NET Framework или C#, които могат да повлияят на съвместимостта на приложението.

2.1.4 Сравнение с други технологии

Сравнителният анализ на различни технологии за разработка на графични потребителски интерфейси (GUI) е от съществено значение за разработчиците, които търсят подходящия инструмент за конкретните си проекти. В таблицата (Таблица 1 , стр.6) се разглеждат и сравняват четири различни технологии: C# и CLR на .NET Framework, Java Swing, Python Tkinter и C++ Window Builder.

Сравняваме точно тези четири технологии за създаване на графични потребителски интерфейси, защото те са най-широко използваните инструменти в съответните програмни езици и платформи. В сравнителния анализ се фокусираме върху различни аспекти като ефективност, удобство при използване, функционалност и поддържани платформи. Тези технологии предлагат различни подходи към разработката на GUI приложения, като C# и CLR на .NET Framework се използват в екосистемата на Microsoft, Java Swing за Java приложения, Python Tkinter за Python приложения и C++ Window Builder за приложения на C++. Сравнявайки тези технологии, разработчиците могат да вземат по-информирано решение за подходящия инструмент за техните конкретни проекти.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

Технологии	Предимства	Недостатъци	Безплатно/Такса лиценз
C# и CLR на .NET Framework	Лесен за изучаване, сигурност, обширна общност. Управление на паметта, множество библиотеки и компоненти, лесна интеграция.	Зависимост от Windows, по-високи ресурсни изисквания. Ограничения при разработка за различни операционни системи.	Безплатно
Java Swing	Платформена независимост, богат набор от компоненти, персонализация, стабилност.	По-труден за изучаване, приложенията на Swing могат да бъдат по-бавни, ограничена интеграция с Уеб.	Безплатно
Python Tkinter	Вграден в Python, лесен за използване, предлага обширен набор от компоненти, платформено независим.	Ограничени визуални възможности, не поддържа напреднали контроли като таблици и дървета, не стандартизиран външен вид, ограничени ресурси	Безплатно
C++ Window Builder	Бърз разработъчен цикъл, пълна интеграция с C++, богати библиотеки и компоненти, вградена поддръжка на множество платформи.	Не е безплатен, за него трябва да се закупи лиценз, труден за изучаване, платформена зависимост.	Pro Term License - 520 € Professional - 1869€ Enterprise – 4399 € Architect - 7149 €

Таблица 1 - Таблица сравняваща четири от най-популярните технологии за разработка на десктоп приложения



2.2 Избор на технология за базата данни

-Защо MySql?

- Приложението изисква стриктен контрол и структурност на данните, затова MySQL като релационна база данни е подходящ избор, който предлага стабилност и надеждност.

2.2.1 СУБД – MySQL

- MySQL е най-популярната система за управление на SQL бази данни. Тя се разпространява и поддържа от Oracle Corporation.

Предимства:

- ✓ Структурност – MySQL осигурява структуриран и организиран начин за съхранение на данни чрез използване на таблици с редове и колони.
- ✓ ACID съответствие – MySQL поддържа транзакции и гарантира атомарност, консистентност, изолация и устойчивост на данните.
- ✓ Сложни заявки – MySQL позволява извършването на сложни заявки и съединения на данни, което е полезно за анализ и докладване.
- ✓ Сигурност: MySQL предлага автентикация и авторизация на потребителите, както и възможности за шифроване на данни.

Недостатъци:

- ✗ Скалируемост – MySQL може да ограничи скалируемостта, особено при големи обеми на данни.
- ✗ Фиксирани схеми – Промяната на структурата на таблиците може да бъде сложна и изисква преобразуване на данните.
- ✗ Изисква опит – Използването на MySQL изисква познания в SQL и релационни бази данни, което може да бъде предизвикателство за новите потребители.



2.2.2 Сравнение на релационна и нерелационна база от данни

- Релационните бази данни са вид бази данни, които се основават на релационен модел на данни. Те се състоят от таблици с редове и колони, където всяко поле от таблицата съдържа данни за конкретен атрибут на обект или събитие. В релационните бази данни, данните се организират във връзки между различните таблици чрез използването на ключове, които позволяват свързването на данните между различните обекти. Този модел осигурява структуриран и ефективен начин за съхранение, управление и извличане на данни.

Предимства:

- ✓ Структурираност- Подходящи за схеми, които изискват стриктна структура и връзки между данните.
- ✓ ACID съответствие- Осигуряват транзакции и данни с висока интегритетност.
- ✓ Мощни заявки- Поддържат сложни заявки и агрегация на данни.

Недостатъци:

- ✗ Сложност- Изискват добро разбиране на SQL и сложност при мащабиране.
 - ✗ Фиксирани схеми- Променянето на структурата на данните може бъде трудно.
-
- Нерелационните бази данни са системи за съхранение и управление на данни, които не следват традиционния релационен модел, използван в SQL базите данни. Те предлагат гъвкавост и мащабируемост при обработката на големи обеми данни и приложения, които изискват висока наличност и бързодействие. Нерелационните бази данни се използват за различни цели, включително уеб приложения, социални мрежи, анализ на данни в реално време и други сценарии, където е необходимо да се работи с големи и разнообразни данни.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

Предимства:

- ✓ Гъвкавост- Подходящи за данни с различна структура (например, документи, графи и времеви серии).
- ✓ Лесно мащабиране- Поддържат хоризонтално мащабиране.
- ✓ Бързи операции с писане- Подходящи за приложения с интензивно писане.

Недостатъци:

- ✗ Липса на ACID- Обикновено не предлагат същите гаранции за ACID като релационните бази данни.
 - ✗ Сложност при анализ- Могат да бъдат предизвикателство за сложни анализи и съединения на данни.
- **Защо релационна база от данни е по-добрият избор за този проект?**

Избрах релационна база данни за моя проект, защото релационните бази данни предлагат стабилна и надеждна структура за съхранение на чувствителна информация. С използването на MySQL мога лесно да организирам и управлявам данните за потребителите, включително техните идентификационни данни и криптирани файлове, с което подпомагам сигурността на приложението.

Релационните бази данни също така предоставят мощни възможности за търсене и филтриране на данни, които са важни за работата със защитените файлове и за управлението на потребителските данни. В крайна сметка, изборът на релационна база данни подчертава моето стремление към сигурност, надеждност и лесна манипулация на данни във възможно най-безопасна среда.



2.2.3. Сравнение на MySQL с други технологии за бази данни

- **MySQL и PostgreSQL**

- **Отворен код и лиценз:**

MySQL и PostgreSQL са двете отворени източници, което означава, че техните кодове са свободно достъпни и могат да бъдат променяни и разпространявани. Обаче, PostgreSQL има по-либерален лиценз, който позволява по-широко използване, докато MySQL използва комбинация от различни лицензи в зависимост от случая.

- **Функционалности:**

PostgreSQL предлага по-широк спектър на функционалности, особено по отношение на сложни заявки и манипулации с данни. Той поддържа напредничави функции като CTE (Common Table Expressions), географски типове данни и много други, които са полезни за разнообразни приложения. MySQL също разполага с обширни възможности, но в някои случаи е по-ограничен в сравнение с PostgreSQL.

- **Екосистема и инструменти:**

MySQL има по-голямо разпространение и по-широка общност от PostgreSQL, което означава, че има по-голям избор от инструменти, библиотеки и ресурси за поддръжка и разработка. Въпреки това, PostgreSQL привлича все повече внимание поради своите разширени възможности и се развива като предпочитан избор за сериозни и разнообразни проекти.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

- Производителност и мащабируемост:

И двете бази данни са високопроизводителни и мащабируеми, но производителността може да варира в зависимост от конкретните изисквания и настройки на приложението. Обикновено се смята, че PostgreSQL има по-добра производителност за сложни заявки и тежки натоварвания, докато MySQL може да е по-бърз при по-опростени заявки.

- **MySQL и MongoDB**

- **Модел на данните:**

- MySQL: MySQL е релационна база данни, която използва таблици с редове и колони за съхранение на данни. Тя следва стандартния SQL модел и поддържа транзакции, индекси и външни ключове.
 - MongoDB: MongoDB е нерелационна база данни, която използва документи в JSON/BSON формат за съхранение на данни. Тя не изисква схема на предварително дефинирани колони и позволява гъвкава структура на данните.

- **Език на заявките:**

- MySQL: Използва SQL (Structured Query Language) за заявки и манипулация на данни. SQL е мощен и широко използван език за работа с релационни бази данни.
 - MongoDB: MongoDB използва JavaScript-подобен заявячен език, наречен Query Language (MQL), за заявки и манипулация на данни.

- **Сложност на инсталация и настройка:**

- MySQL: MySQL е по-лесна за инсталиране и конфигуриране, особено за потребители с опит във връзка с традиционни релационни бази данни.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

- MongoDB: MongoDB може да бъде по-сложна за настройка, особено при мащабируеми инсталации или когато се изисква конфигурация за висока наличност и издръжливост.

3. Софтуерен дизайн

3.1 Софтуерна архитектура

Проектът е основан на софтуерната архитектура MVC (Model-View-Controller). Той е подходящ за разработване на приложения, които имат сложна логика за обработка на данни. Той разделя приложението на три основни компонента - Модел, Изглед и Контролер, които работят заедно, но с различни отговорности.

✖ Модел(Model):

- ✓ Моделът представлява данните и бизнес логиката на приложението. В този проект това включва криптиране и декриптиране на файлове със специфичния алгоритъм. Моделът не е зависим от изгледа или контролера и предоставя интерфейс за манипулиране на данни без да се интересува от начина, по който тези данни се представят или взаимодействат с потребителя.

✖ Изглед(View):

- ✓ Изгледът е отговорен за представянето на данните на потребителя и приемането на входни данни от него. В случая в този проект, изгледът би представлявал графичния интерфейс (GUI), който потребителят използва за криптиране/декриптиране на файлове. Изгледът е пасивен и не съдържа логика за обработка на данни.

✖ Контролер(Controller):



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

- ✓ Контролерът служи за свързване на изгледа и модела. Той приема входни действия от потребителя чрез изгледа и ги преобразува в действия върху модела. Контролерите на проекта обработват действия като криптиране или декриптиране на файлове и други функционалности.

3.2 Предимства и недостатъци на MVC моделът

Предимства:

- ✓ Разделяне на отговорности: MVC разделя приложението на три компонента с ясно дефинирани отговорности, което улеснява разработката и поддръжката на софтуера.
- ✓ Лесна поддръжка: Понеже всеки компонент има ясно определени функции, поддръжката и разширението на приложението става по-лесно. Промените в единия компонент обикновено не засягат другите.
- ✓ Повторно използване на код: Моделът и контролерът могат да се използват повторно в различни части на приложението или дори в различни приложения.
- ✓ Улеснено тестване: Поради разделението на логиката на данните, бизнес логиката и визуалната част, е по-лесно да се напишат модулни тестове за всеки компонент отделно.
- ✓ Многоплатформена поддръжка: MVC може да се използва в разработката на уеб, десктоп и мобилни приложения.

Недостатъци:

- ✗ Сложност: За по-големи и сложни приложения MVC може да се окаже твърде сложен и да влече голям брой файлове и структура.
- ✗ Възможен прекомерен брой връзки: При неправилно използване, моделът и изгледът може да се свържат прекомерно тясно чрез контролера, което може да наруши разделянето на отговорности и да направи приложението по-сложно за поддръжка и разширение.
- ✗ Възможен излишен брой съобщения: При някои реализации на MVC модела, изгледът може да изпраща много заявки към контролера, което може да предизвика излишен обмен на данни между компонентите.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

- ✱ Не е подходящ за всеки тип приложение: MVC е подходящ за множество приложения, но може да не е най-добрият избор за по-прости или по-сложни проекти, които изискват различни модели на архитектурата.

3.3 Файлова структура на проекта

Папка FileSAVER – съдържа структурата на проект – шест графични прозореца и един допълнителен клас, който всички графични форми наследяват. Всеки графичен прозорец има два файла съдържащи C# кодът за графичната част и за функционалността.

- AdminTools.cs – Дефинира класа AdminTools, който е част от функционалността за администриране на приложението.
- AdminTools.Designer.cs – Файл, генериран от Visual Studio, който съдържа дизайнерския код за AdminTools.
- CustomForm.cs: Дефинира клас CustomForm, който бива наследяван от всички графични форми и съдържа общи методи и информация нужна за всички графични форми .
- Login.cs: Дефинира класа Login, който е отговорен за логическата част на формуляра за вход.
- Login.Designer.cs: Файл, генериран от Visual Studio, който съдържа дизайнерски код за Login.
- MainPage.cs: Дефинира класа MainPage, който е отговорен за логическата част на главната страница на приложението.
- MainPage.Designer.cs: Файл, генериран от Visual Studio, който съдържа дизайнерски код за MainPage.
- MyAccount.cs: Дефинира класа MyAccount, който е отговорен за логическата част на страницата "Моят профил".
- MyAccount.Designer.cs: Файл, генериран от Visual Studio, който съдържа дизайнерски код за MyAccount.
- Program.cs: Точката на вход на приложението.
- RecoverPassword.cs: Дефинира класа RecoverPassword, който е отговорен за логическата част на страницата за възстановяване на парола.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

- RecoverPassword.Designer.cs: Файл, генериран от Visual Studio, който съдържа дизайнерски код за RecoverPassword.
- Register.cs: Дефинира класа Register, който е отговорен за логическата част на формуляра за регистрация.
- Register.Designer.cs: Файл, генериран от Visual Studio, който съдържа дизайнерски код за Register.

Файлове с ресурси:

- AdminTools.resx: Съдържа ресурси (текстове, изображения), използвани от AdminTools.
- Login.resx: Съдържа ресурси (текстове, изображения), използвани от Login.
- MainPage.resx: Съдържа ресурси (текстове, изображения), използвани от MainPage.
- MyAccount.resx: Съдържа ресурси (текстове, изображения), използвани от MyAccount.
- RecoverPassword.resx: Съдържа ресурси (текстове, изображения), използвани от RecoverPassword.
- Register.resx: Съдържа ресурси (текстове, изображения), използвани от Register.

Други файлове:

- FileSAVER.csproj: Файл с проекта на Visual Studio, който съдържа информация за компилация и конфигурация.
- FileSAVER.csproj.user: Файл с потребителски настройки за проекта.

Допълнителна информация относно формата на файловете:

- Файловете с дизайнерско разширение .Designer.cs се генерират автоматично от Visual Studio и не е нужно да се редактират ръчно.
- Файловете с ресурсно разширение .resx се използват за съхранение на текстове, изображения и други ресурси, използвани от приложението.
- C# файловете съдържат код, който дефинира логиката и функционалността на приложението.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

Папки в папката FileSAVER:

- bin – Съдържа компилираните DLL файлове на проекта.
- obj – Съдържа временни файлове, генерирани при компилация.
- properties – Съдържа конфигурационни файлове на проекта

Папката „resources“

- Съдържа иконките, използвани в графичния интерфейс на приложението.

Папката „sql“

- Съдържа dump файловете на MySQL, които мога да се използват за възстановяване на базата данни. Dump файловете са текстови файлове, които съдържат SQL команди за създаване на таблици и вмъкване на данни.

3.4. Описание на взаимодействието с приложението

Нерегистрираният потребител, няма достъп до основните функционалности на приложението (Виж Фиг.1). Причината за това, е че когато един потребител пожелае да защити своя файл съдържащ лична информация, този файл трябва да бъде асоцииран с някой потребител, защото на един компютър може да бъде използван от различни хора.

Регистрираните потребители ще могат да използват всичките функционалности на приложението с изключение на инструментите на администратора, освен ако те не са администратор вече. (Виж Фиг.2) Администраторския профил е само един и се регистрира, като идеята е ако един компютър се използва от повече хора, администратора да следи техните действия и да им съдейства ако имат нужда от промяна на данни и други.

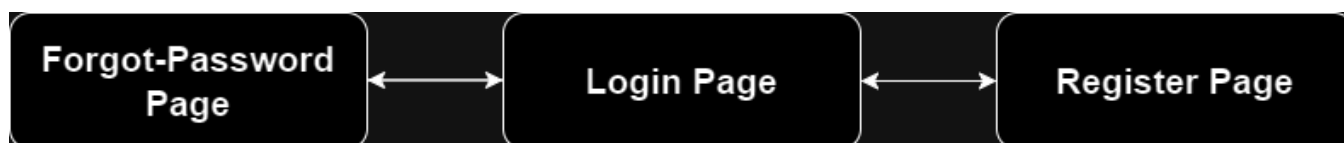
Обикновените потребители ще могат да се възползват от останалите функционалности – да криптират/декриптират свой файл, като файлът не може да бъде повече от 2GB, поради



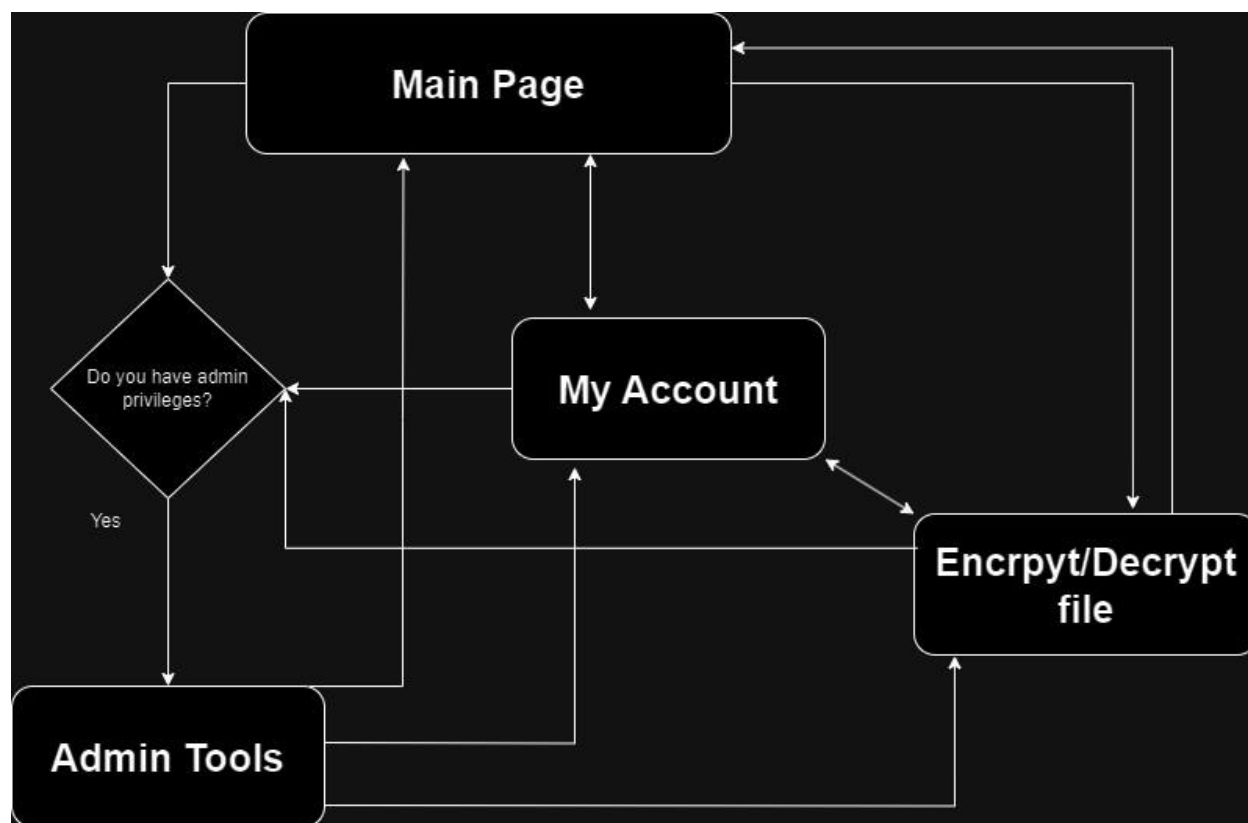
„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

библиотеката, чрез която се реализира избиране на файл или да променят информация за акаунта си.



Фиг. 1 Графични прозорци достъпни за нерегистриран потребител.



Фиг. 2 Графични прозорци достъпни за регистриран потребител



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

1. Вход (Виж Приложение 1)

Функционалност:

- Позволява на потребителя да влезе в приложението с име на потребител и парола.

Взаимодействие:

- Потребителят въвежда своето име на потребител и парола.
- При натискане на бутон "Вход" се извършва проверка на валидността на данните.
- При успешно влизане, потребителят се пренасочва към главната страница.
- При невалидни данни, се показва съобщение за грешка.

Връзки:

- Регистрация
- Забравена парола

2.Регистрация (Виж Приложение 2)

Функционалност:

- Позволява на потребителя да се регистрира в приложението.

Взаимодействие:

- Потребителят въвежда своето име на потребител, парола, имейл, възраст и роля (администратор/обикновен потребител).
- При натискане на бутон "Регистрация" се извършва проверка на валидността на данните.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

- При успешна регистрация, потребителят се пренасочва към главната страница.
- При невалидни данни се показва съобщение за грешка.

3. Възстановяване на парола (Виж Приложение 3)

Функционалност:

- Помага на потребителя да възстанови забравена парола.

Взаимодействие:

- Потребителят въвежда своето потребителско име.
- При натискане на бутон "Възстановяване" се изпраща имейл с инструкции за промяна на паролата.
- При невалиден имейл адрес, се показва съобщение за грешка.

4. Главна страница (Виж Приложение 4)

Функционалност:

- Предоставя основната функционалност на приложението след успешно влизане.

Взаимодействие:

- Потребителят може да криптира и декриптира файлове с размер до 4 GB.
- Потребителят задава парола за своя файл, след което избира файлът и при натискане на бутона за криптиране/декриптиране се извършва съответната операция.

5. Моят Профил (Виж Приложение 5)

Функционалност:

- Позволява на потребителя да променя своите данни.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

Взаимодействие:

- Потребителят може да променя своето име, име на потребител, имейл адрес, възраст и парола.
- При натискане на бутон "Запази" се извършват промените.

6.Инструменти за администратор (Виж Приложение 6)

Функционалност:

- Предоставя функции, достъпни само за администратори..

Взаимодействие:

- Администраторът може да променя името, имейл адреса и възрастта на другите потребители.
- Администраторът може да изтрива потребители.
- Администраторът може да преглежда действията, които потребителите извършват.
- Всички графични форми са проектирани да бъдат интуитивни и лесни за използване. Приложението използва валидация на данните, за да се гарантира, че са въведени правилно. Използвани са съобщения за грешка, за да се информира потребителят за възникнали проблеми. Графичният интерфейс на приложението е проектиран да е функционален, лесен за използване и сигурен.

3.5Дизайн на симетричният алгоритъм за криптиране

В този раздел ще бъде представен алгоритъмът за криптиране и декриптиране, разработен за нуждите на приложението. Алгоритъмът е прототип и в него са включени четири вида



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

трансформации, които се прилагат върху данните в определена последователност. Съдържанието на файла и на паролата е представен чрез шестнайсетичната бройна система.

3.5.1 Криптиране

firstStepOfEncryption (List<string> key, List<string> file) – този метод представлява първата стъпка от алгоритъма за криптиране.

- Методът приема два параметъра, които са списък от символни низове за паролата, с която се криптира файла и за самият файл.
- В първият for-цикъл, който се завърта в диапазона на дължината на паролата(key.Count) извършва завъртане на паролата взимайки последният елемент от ключа (key[key.Count-1]) и се премахва от списъка (key.RemoveAt(key.Count – 1)) , след което премахнатия елемент се вмъква на първата позиция(key.Insert(0, lastElement)), по този начин паролата се обръща наляво. (Виж Фиг. 3, Фиг.4)
- След като ключът е ротиран, вторият for-цикъл събира елементите от обърнатия ключ и ги добавя към съдържанието на файла, като ги вмъква в края на списъка 'file'.(Виж Фиг.3, Фиг.4)
- Така, този метод съчетава ключа със съдържанието на файла, като ротира ключа преди това. Това може да се разглежда като начин за "замаскиране" на съдържанието на файла, като се използва ключ за промяна на последователността на данните. Този процес може да допринесе за повишаване на сигурността на данните, като прави криптоанализа по-труден.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

```
//1st part of my encryption algorithm -> shuffle the file with the key
1 reference
private void firstStepOfEncryption(List<string> key, List<string> file)
{
    //We rotate the key for example if its 12 34 56 , now is 56 34 12
    for (int i = 0; i < key.Count; i++)
    {
        if (key == null || key.Count == 0)
            return;

        string lastElement = key[key.Count - 1];
        key.RemoveAt(key.Count - 1);
        key.Insert(0, lastElement);
    }
    //Then the rotated password is added to the file content
    for (int i = 0; i < key.Count; i++)
    {
        file.Add(key[i]);
    }
}
```

Фигура 3 – Първата трансформация при криптиране на файл.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

Пример за по-лесно разбиране на първата стъпка от криптирането

File: 64 a8 b3 41 11

Key(Password): 46 12 23 bb

Key(Password): 46 12 23 bb

Паролата се обръща
----->

Rotated Key: bb 23 12 46

Сега смесицата от файлът и паролата се използва в следващите стъпки от криптирането

File + Rotated Key = 64 a8 41 11 bb 23 12 46

Фигура 4 – Визуализация на първата трансформация при криптиране на файл *secondStepOfEncryption (List<string> file)* – този метод представлява втората стъпка от алгоритъма за криптиране. Тя има за цел да размести позициите на елементите, така че да не се знае оригиналната подредба на елементите от файла.

- Методът има един параметър и той е смесицата от файлът и паролата, който първият метод създава.
- Първият for -цикъл променя позициите на всеки трети елемент с елемента, който се намира на позицията му минус две. (Виж Фиг.5, Фиг.6)
- Вторият for -цикъл разменя позициите на всеки трети елемент започващ от втория индекс '1'. Това са елементите, които се намират между числата, които първият for -цикъл разменя. (Виж Фиг.5, Фиг.6)



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

```
//second step of my encryption is every 3rd hex number to change position
//with the element with his's index - 2 and change positions
//to the other numbers left which are with index i and i+3
2 references
private void secondStepOfEncryption(List<string> file)
{
    //Change's every third elemnt with the elemnt with his index-2
    //For example "84" "21" "51" "54" "78" "96" -> "51" "21" "84" "96" "78" "54"
    for (int i = 2; i < file.Count; i += 3)
    {
        string a = file[i - 2];
        file[i - 2] = file[i];
        file[i] = a;
    }

    //16 32 14 60 50 30 we need to scramble i=1 i+=3 numbers so we just
    //change positions of 32 with 50 and it will look like this
    //16 50 14 60 32 30 the step over this will change the other numbers
    //so finnaly will look like this 14 50 16 30 32 60 which is well scrambled
    //and it can be easily reversable for decryption.

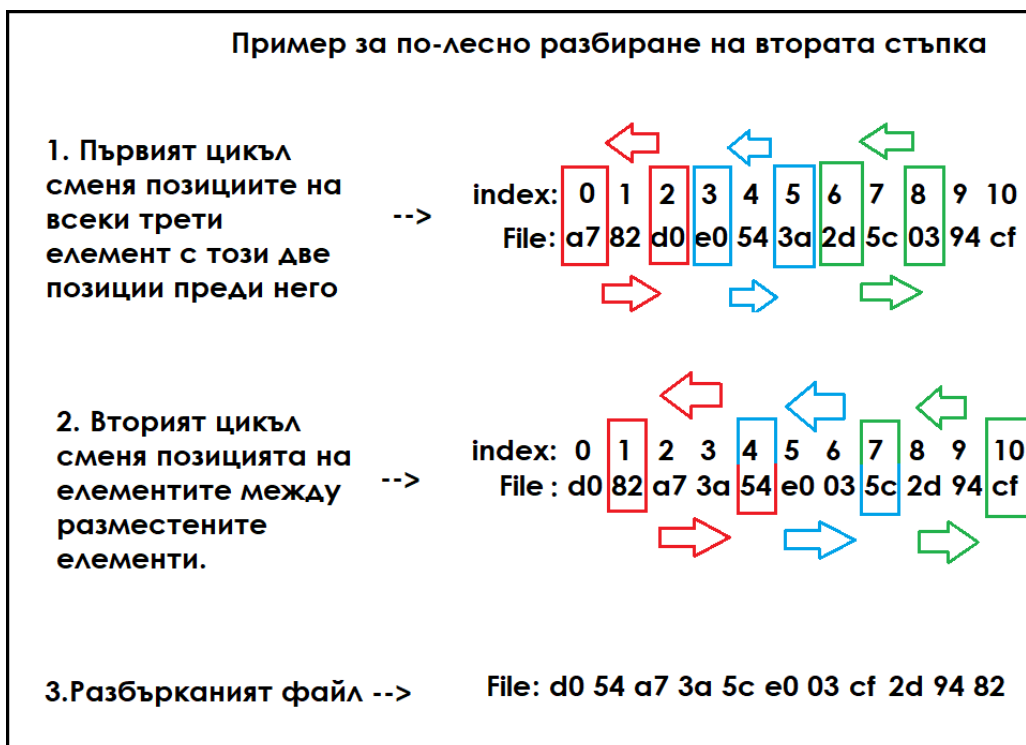
    for (int i = 1; i < file.Count; i += 3)
    {
        if (i + 3 >= file.Count) //if i + 3 doesn't exist we break
        {
            break;
        }
        string element = file[i + 3];
        file[i + 3] = file[i];
        file[i] = element;
    }
}
```

Фигура 5 – Втората трансформация при криптиране на файл



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов



Фигура 6 – Визуализация на втората трансформация от алгоритъма за криптиране

thirdStepOfEncryption (List<string> file) – този метод представлява третата стъпка от алгоритъма за криптиране.

- Методът има един параметър и той е списък от низове съдържащ смесицата от файлът и паролата, който втория метод генерира, като елементите са представени чрез шестнадесетичната бройна система.

- Третата стъпка разменя символите на всички двойки последователни елементи от списъка в зависимост от това дали сумата на двете шестнадесетични числа е четна или нечетна.

- Първите два масива 'char[] firstNumberValues' и 'char[] nextNumberValues' съдържат двата символа от първия елемент и следващия. По този начин можем да достъпваме като отделни променливи всяка част от елемента.(Виж. Фиг. 7)



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

- Сумата на двойките числа 'sum' намираме чрез сбора на символите на първия елемент 'sumFristHex' и символите на вторият елемент 'sumSecondHex'. Тук разбъркването на символите зависи от това дали 'sum' е четно или нечетно. (Виж. Фиг. 7)
- Методът 'int. Parse' има параметър: 'System.Globalization.NumberStyles.HexNumber', който превръща символа от шестнадесетичната бройна система в десетична, за да може да се извърши операцията събиране.
- Ако сумата 'sum' е четна първият символ от първият елемент сменя позицията си с вторият символ от вторият елемент. (Виж. Фиг. 7, Фиг. 8)
- Ако сумата 'sum' е нечетна вторият символ от първият елемент сменя позицията си с първият символ от вторият елемент. (Виж. Фиг. 7, Фиг. 8)

```
//If the sum between 2 of the hex is odd or even they trade some of them values
2 references
private void thirdStepOfEncryption(List<string> file)
{
    for (int i = 0; i < file.Count - 1; i++)
    {
        //Takes the first hex number for example "16" and separate the symbols -> "1" and "6" and converts them to int
        char[] firstNumberValues = file[i].ToCharArray();
        int firstValueFromFirstNumber = int.Parse(firstNumberValues[0].ToString(), System.Globalization.NumberStyles.HexNumber);
        int secondValueFromFirstNumber = int.Parse(firstNumberValues[1].ToString(), System.Globalization.NumberStyles.HexNumber);

        //Takes the second hex number for example "13" and separate the symbols -> "1" and "3" and converts them to int
        char[] nextNumberValues = file[i + 1].ToCharArray();
        int firstValueFromNextNumber = int.Parse(nextNumberValues[0].ToString(), System.Globalization.NumberStyles.HexNumber);
        int secondValueFromNextNumber = int.Parse(nextNumberValues[1].ToString(), System.Globalization.NumberStyles.HexNumber);

        //Makes the summary of numbers in the hex
        int sumFirstHex = firstValueFromFirstNumber + secondValueFromFirstNumber; //for example if the first hex is 16 sum = 1 + 6 = 7
        int sumSecondHex = firstValueFromNextNumber + secondValueFromNextNumber; //for example if the second hex is 13 sum = 1 + 3 = 4
        int sum = sumFirstHex + sumSecondHex; //Makes the sum of the sums of the hex symbols -> 7 + 4 = 11
        //If the sum is even we change the symbols from the hex -> we change frist symbol
        //from the first hex with the second symbol from the second hex
        //Like this "16" "13" -> "36" "11"
        if (sum % 2 == 0)
        {
            string newValueFori = secondValueFromNextNumber.ToString("X") + secondValueFromFirstNumber.ToString("X");
            string newValueForiplusone = firstValueFromNextNumber.ToString("X") + firstValueFromFirstNumber.ToString("X");
            file[i] = newValueFori;
            file[i + 1] = newValueForiplusone;

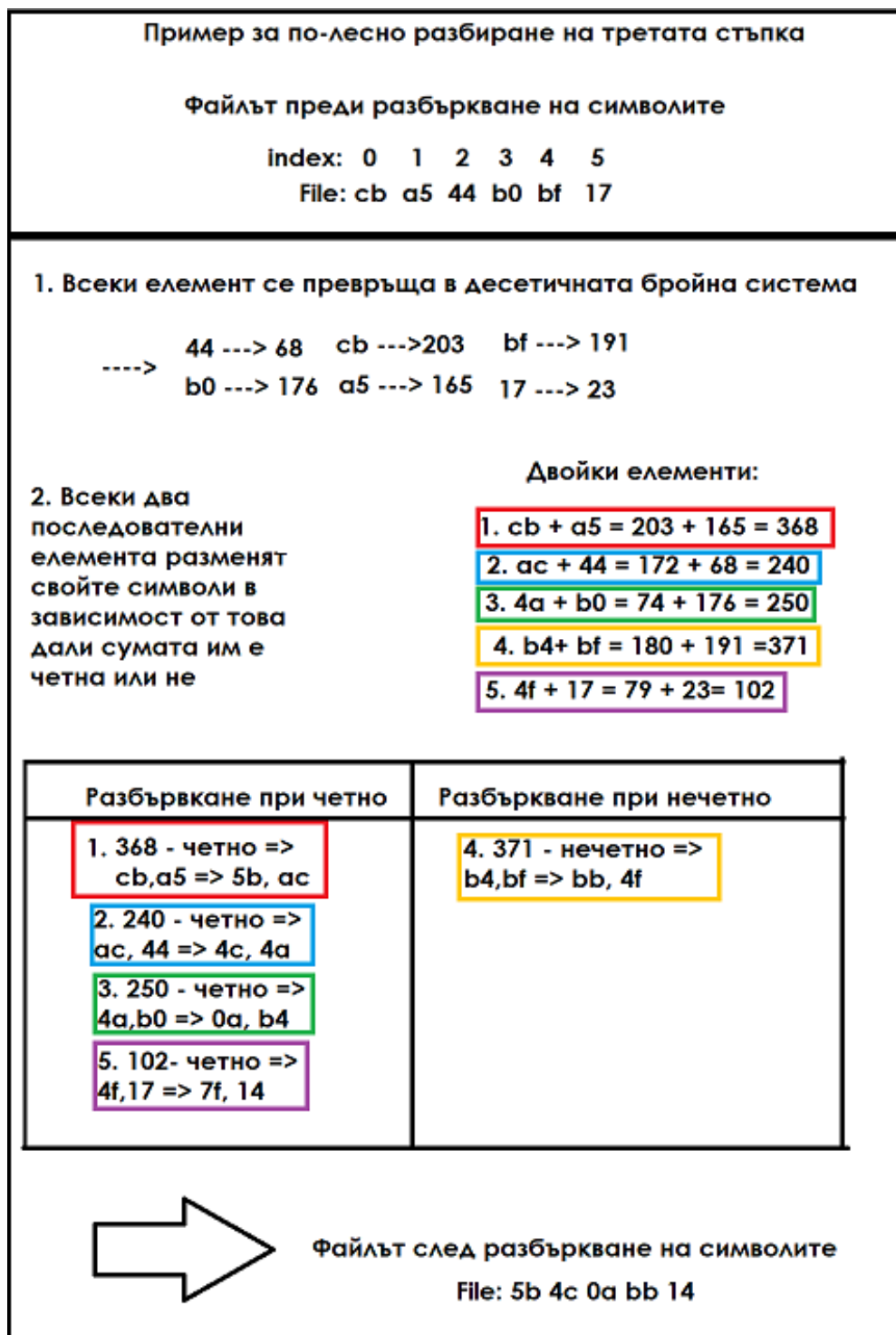
            //If the sum is odd we change the symbols from the hex -> we change second symbol from the first hex with the
            //first symbol from the second hex
            //Like this "16" "13" -> "13" "63"
        } else
        {
            string newValueFori = firstValueFromFirstNumber.ToString("X") + firstValueFromNextNumber.ToString("X");
            string newValueForiplusone = secondValueFromFirstNumber.ToString("X") + secondValueFromNextNumber.ToString("X");
            file[i] = newValueFori;
            file[i + 1] = newValueForiplusone;
        }
    }
}
```

Фигура 7 – третата стъпка от алгоритъма за криптиране



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов



Фиг. 8 – Визуализация на третата стъпка от алгоритъма за криптиране



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

fourthStepOfEncryption (List<string> file) – този метод представлява четвъртата стъпка от алгоритъма за криптиране.

- Методът приема списък от низове съдържащ съдържанието на файла с паролата разбъркани от предишните методи, като елементите в списъка са представени чрез шестнадесетичната бройна система.(Виж Фиг.9, Фиг.10)

- Първият for -цикъл разменя символите на всеки втори елемент започващ от нулевия индекс от масива. Ако например нулевият елемент представлява “4a” след размяна на символите би изглеждал така “a4”. След обръщането на символите текущият ‘file[i]’ елемент сменя позицията си с елемента отдясно от него ‘file[i + 1]’. Ако броят на елементите е нечетен, последният елемент няма елемент отдясно, затова в началото на методът има if оператор, който следи когато текущият елемент е последният от низа да спре ‘for’-цикъла оставяйки последният елемент със същите символи на същото място. (Виж Фиг.9, Фиг.10)

- Вторият if оператор в края на методът следи за това когато файлът съдържа нечетен брой елементи след като са разбъркани последният елемент остава на същото място и със същите символи. За да компенсира това, методът разменя първия елемент с последния, като по този начин тази слабост бива отстранена и всички елементи успешно сменят позицията си и някой от тях позицията на своите символи. (Виж Фиг.9, Фиг.10)



```
//fourth step changes every element with even index to change the position of his hex symbols (e5 -> 5e)
//and position with the element on the right(index+1)
1 reference
private void fourthStepOfEncryption(List<string> file)
{
    for (int i = 0; i < file.Count; i += 2)
    {
        //If file.Count is odd the last element can't get replaced because is alone, so it breaks
        if (i + 1 >= file.Count) break;
        char[] hex_value = file[i].ToCharArray();

        //Rotating the symbols of an every second elemnet
        char first_hex_value = hex_value[0];
        hex_value[0] = hex_value[1];
        hex_value[1] = first_hex_value;

        //Actualizing the actual file with the edited elements which had roatated their symbols
        file[i] = "";
        foreach (char hex_char in hex_value)
        {
            file[i] += hex_char;
        }

        //Every elemnt change position with the elemnt on the right
        string element = file[i + 1];
        file[i + 1] = file[i];
        file[i] = element;
    }

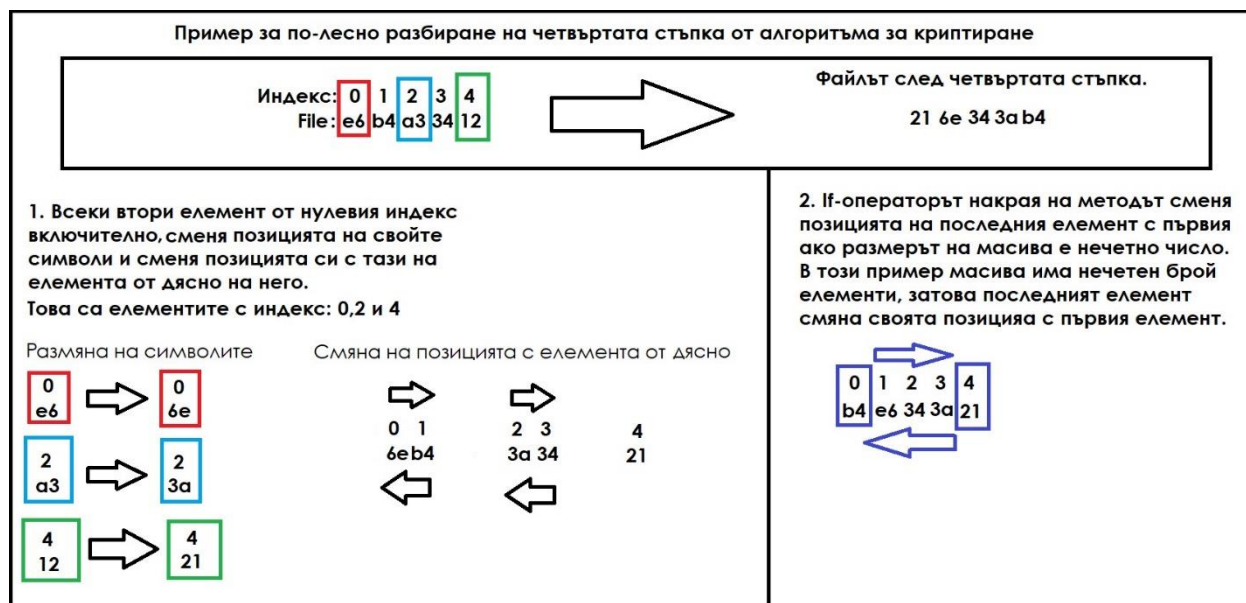
    //When the file count is odd number the last hex number can't change position with number,
    //because is the last number, so the
    //last hex number after encryption stays the same and in the same position, that's why we
    //change it with the first element
    if (file.Count % 2 != 0)
    {
        string firstEl = file[0];
        file[0] = file[file.Count - 1];
        file[file.Count - 1] = firstEl;
    }
}
```

Фиг. 9 – Четвъртата стъпка от алгоритъма за криптиране



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов



Фигура 10 – Визуализация на четвъртата стъпка от алгоритъма за криптиране.



3.5.2. Декриптиране

firstStepOfDecryption (List<string> file) – този метод представлява първата стъпка от процеса на декриптиране. Първата стъпка има за цел да разшифрова четвъртата стъпка от процеса на криптиране. При процеса на декриптиране всичко се извършва в обратен ред спрямо криптирането.

- Методът има един параметър, който е списък от низове съдържащ данните на файла в представени в шестнадесетичната бройна система. (Виж фиг. 11, фиг. 12)
- Първият if - оператор проверява дали броят на елементите е нечетно число, защото при нечетен брой елементи четвъртата стъпка сменя позицията на първия и последния елемент, затова ако е нечетно първото действие е отново да разменим първият с последният елемент. (Виж фиг. 11, фиг. 12)
- Първият for - цикъл сменя отново позициите на елементите, които четвъртата стъпка сменя, по този начин елементите се нареждат по правилният начин. (Виж фиг. 11, фиг. 12)
- Вторият for - цикъл сменя символите, които четвъртата стъпка разменя по този начин символите се нареждат в правилният ред. (Виж фиг. 11, фиг. 12)



```
//Method for decryption the fourth step of decryption
1 reference
private void firstStepOfDecryption(List<string> file)
{
    //In the encryption step when the count of all of the
    //hex numbers are odd, the last number can change its
    //position because its the last, so the encryption
    //changes the last element with the first so here first
    //before other reverse decryption we need to change the
    //first with the last element
    //!(THIS IS ONLY FOR ODD file.Count)!
    if (file.Count % 2 != 0)
    {
        string firstEl = file[0];
        file[0] = file[file.Count - 1];
        file[file.Count - 1] = firstEl;
    }
    //First reverse the shuffle and make the elements in
    //the right order
    for (int i = 0; i < file.Count; i += 2)
    {
        if (i + 1 >= file.Count) break; //When the file.Count
        //is odd it has to break so the last elemnt doesn't change or edit
        //in some way because its changed with the first element look the if above
        string element = file[i + 1];
        file[i + 1] = file[i];
        file[i] = element;
    }
    //Change the rotated symbols on every second element
    for (int i = 0; i < file.Count; i += 2)
    {
        if (i + 1 >= file.Count) break; //When the file.Count is odd it has
        //to break so the last elemnt doesn't change or edit
        //in some way because its changed with the first element look the if above
        char[] hex_value = file[i].ToCharArray();
        char first_hex_value = hex_value[0];
        hex_value[0] = hex_value[1];
        hex_value[1] = first_hex_value;

        file[i] = new string(hex_value);
    }
}
```

Фигура 11 – Първата стъпка от процеса на декриптиране



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

Визуализация за по-лесно разбиране на четвъртата стъпка от процеса на декриптиране

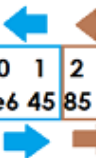
1. Първата стъпка проверява дали броя на елементите е нечетен, в този пример броят на елементите е нечетен (5), следователно сменяме позициите на първия и последния елемент.

Encrypted File: 12 45 85 57 e6



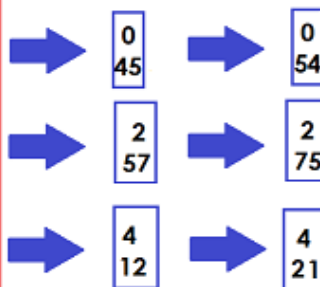
2. Втората стъпка размества позициите на всеки втори елемент започвайки от нулевия елемент с този отясно на него.

Index: 0 1 2 3 4
Encrypted File: e6 45 85 57 12



3. Третата стъпка сменя позициите на символите на всеки втори елемент започвайки от нулевия елемент.

Index: 0 1 2 3 4
Encrypted File: 45 e6 57 85 12



Original File: 54 e6 75 85 21



След четвъртата стъпка от криптирането

Encrypted File: 12 45 85 57 e6

4. По този начин декриптирахме четвъртата стъпка от алгоритъма за криптиране.

Encrypted File: 54 e6 75 85 21

Original File: 54 e6 75 85 21

Фигура 12 – Визуализация на първата стъпка от процеса на декриптиране.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

secondStepOfDecryption (List<string> file) – този метод представлява втората стъпка от процеса на декриптиране. Втората стъпка има за цел да разшифрова третата стъпка от процеса на криптиране. При процеса на декриптиране всичко се извършва в обратен ред спрямо криптирането.

- Методът има един параметър, който е списък от низове съдържащ данните на файла в представени в шестнадесетичната бройна система.
- Тук принципа е един и същ и кодът е идентичен с третата стъпка от процеса на криптиране. Единствената разлика е, че `for` - цикълът започва от последният елемент `'int i = file.Count - 1'` и продължава до `'i > 0'`, което означава, че цикли до вторият елемент. Причината за това е, че третата стъпка от криптирането разбърква символите по двойки, затова декриптирането на тази стъпка става като обхваща двойките в обратен ред от последната двойка до първата. (Виж фиг. 12, фиг. 13)
- Размяната на символите става в зависимост от това дали сборът от двойка символи е четен или нечетен.
- Ако сумата `'sum'` е четна първият символ от първият елемент сменя позицията си с вторият символ от вторият елемент. (Виж фиг. 12, фиг. 13)
- Ако сумата `'sum'` е нечетна вторият символ от първият елемент сменя позицията си с първият символ от вторият елемент. (Виж фиг. 12, фиг. 13)
- Методът `'int. Parse'` има параметър: `'System.Globalization.NumberStyles.HexNumber'`, който превръща символа от шестнадесетичната бройна система в десетична, за да може да се извърши операцията събиране. (Виж фиг. 12, фиг. 13)



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

```
private void secondStepOfDecryption(List<string> file)
{
    for (int i = file.Count - 1; i > 0; i--)
    {
        //Takes the last hex number for example "16" and separate the symbols -> "1" and "6" and converts them to int
        char[] rightNumberValues = file[i].ToCharArray();
        int firstValueFromRightNumber = int.Parse(rightNumberValues[0].ToString(), System.Globalization.NumberStyles.HexNumber);
        int secondValueFromRightNumber = int.Parse(rightNumberValues[1].ToString(), System.Globalization.NumberStyles.HexNumber);
        //Takes the hex number before the last hex number for example "13" and separate the symbols -> "1" and "3" and
        //converts them to int
        char[] leftNumberValues = file[i - 1].ToCharArray();
        int firstValueFromLeftNumber = int.Parse(leftNumberValues[0].ToString(), System.Globalization.NumberStyles.HexNumber);
        int secondValueFromLeftNumber = int.Parse(leftNumberValues[1].ToString(), System.Globalization.NumberStyles.HexNumber);

        //Makes the summary of numbers in the hex
        int sumRightHex = firstValueFromRightNumber + secondValueFromRightNumber; //for example if the first hex
        //is 16 sum = 1 + 6 = 7
        int sumLeftHex = firstValueFromLeftNumber + secondValueFromLeftNumber; //for example if the second hex
        //is 13 sum = 1 + 3 = 4
        int sum = sumLeftHex + sumRightHex; //Makes the sum of the sums of the hex symbols - > 7 + 4 = 11

        //If the sum is even we change the symbols from the hex -> we change frist symbol from the first hex with the
        //second symbol from the second hex
        //like this "16" "13" - > "36" "11"
        if (sum % 2 == 0)
        {
            string newRightValues = firstValueFromRightNumber.ToString("X") + firstValueFromLeftNumber.ToString("X");
            string newLeftValues = secondValueFromRightNumber.ToString("X") + secondValueFromLeftNumber.ToString("X");
            file[i] = newRightValues;
            file[i - 1] = newLeftValues;

            //If the sum is odd we change the symbols from the hex -> we change second symbol from the first hex with
            //the first symbol from the second hex
            //like this "16" "13" - > "13" "63"
        } else
        {
            string newRightValues = secondValueFromLeftNumber.ToString("X") + secondValueFromRightNumber.ToString("X");
            string newLeftValues = firstValueFromLeftNumber.ToString("X") + firstValueFromRightNumber.ToString("X");
            file[i] = newRightValues;
            file[i - 1] = newLeftValues;
        }
    }
}
```

Фигура 12 – втората стъпка от процеса на декриптиране



„Десктоп приложение за засекретено съхраняване на файлове“

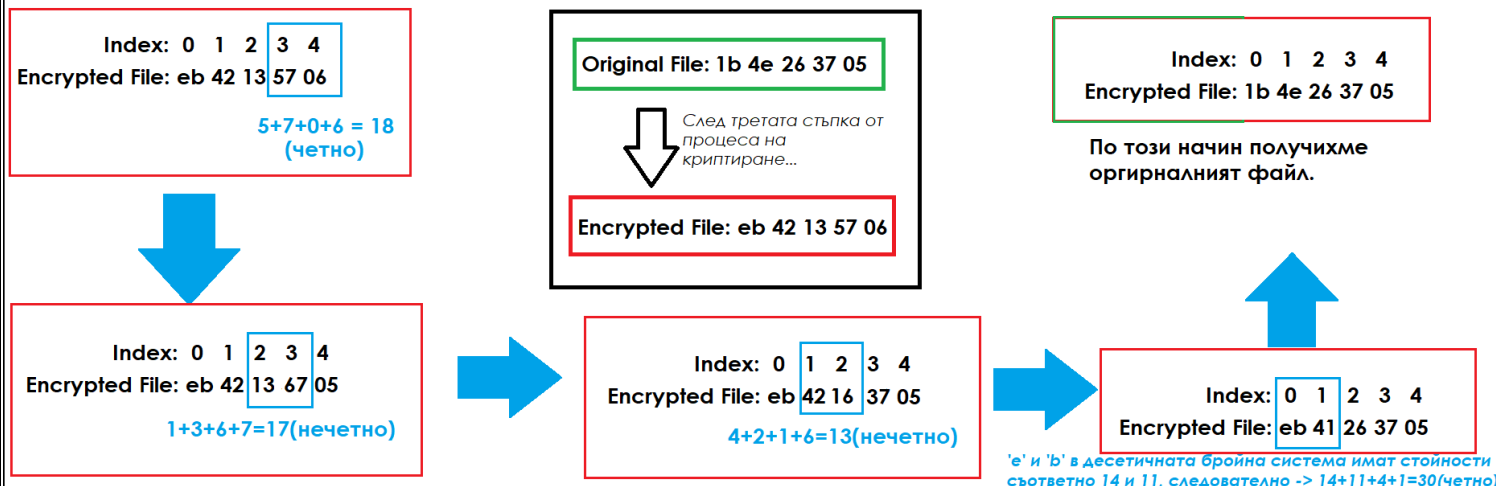
Виктор Дамянов Владинов

Визуализация на втората стъпка от процеса на декриптиране

1. Обхващаме първата двойка елементи, в случая те са последните два и започваме да обхождаме през тях като в зависимост от това дали сборът от елементите е четно или нечетно число се определя размяната на символите на елементите.

При нечетна сума втория символ от първия елемент сменя позицията с първия символ от втория елемент

При четна сума първият символ от първия елемент сменя позицията си с вторият символ от втория елемент



Фигура 13 – визуализация на втората стъпка от процеса на декриптиране.

thirdStepOfDecryption (List<string> file) – този метод представлява третата стъпка от процеса на декриптиране. Третата стъпка има за цел да разшифрова втората стъпка от процеса на криптиране. При процеса на декриптиране всичко се извършва в обратен ред спрямо криптирането.

- Методът има един параметър, който е списък от низове съдържащ данните на файла в представени в шестнадесетичната бройна система. (Виж фиг.12, фиг. 13)
- Първият for - цикъл намира последното число, което ще участва в размяната, защото при втората стъпка от криптирането след разместването на всеки трети елемент с този две позиции преди него, между разместените елементи остават елементите почващи от първи индекс и са разположени през три позиции. Тези елементи втората стъпка ги разменя затова при декриптирането трябва да започнем от зад напред и 'index' променливата ще запази индекса на това последно число участвало в размяната. (Виж фиг.12, фиг. 13)



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

- Вторият for – цикъл извършва размяната на тези числа след като вече знаем кое е последното число участвало в размяната и започваме да ги разменяме през три позиции назад, ако индекса 'i' в цикъла стане по-малко от 3, спираме цикъла, защото това означава, че сме обходили всички числа и сме стигнали до първото. (Виж фиг.12, фиг. 13)
- Третият 'for' – цикъл сменя позициите на всеки трети елемент с този две позиции преди него. По този начин се възстановява правилният ред на елементите. (Виж фиг.12, фиг. 13)

```
private void thirdStepOfDecryption(List<string> file)
{
    //Here we change the numbers that are being rotated in the encryption metod, so the number i and i+3 have to be changed
    //and if i+3 doesn't exist we break
    int index = 0;
    for (int i = 1; i < file.Count; i += 3)
    {
        index = i; //Here we find the index of the last element which participated in the shuffle so next for cycle we
        //can know from which element to start reversing
    }
    for (int i = index; i >= 0; i -= 3)
    {
        if (i == 1 || i == 0 || i == 2) //if the index is below 3 we break, because this
        //means that the last two elements have changed and there are no other elements to change
        {
            break;
        }
        string element = file[i - 3];
        file[i - 3] = file[i];
        file[i] = element;
    }
    //Change's every third elemnt with the elemnt with his index-2
    //For example "84" "21" "51" "54" "78" "96" -> "51" "21" "84" "96" "78" "54"
    for (int i = 2; i < file.Count; i += 3)
    {
        string a = file[i - 2];
        file[i - 2] = file[i];
        file[i] = a;
    }
}
```

Фигура 13 – третата стъпка от процеса на декриптиране



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

Визуализация за по-лесно разбиране на третата стъпка от процеса на декриптиране

1. Първият 'for'-цикъл задава стойност на променливата `index`, която съдържа индекса на последното число участвало в размяната. Чрез това число сменяме позициите на числото на позиция `index` и на `index-3`. Когато `i` от вторият цикъл стане равно на 0, 1 или 2, това означава, че вторият цикъл е обходил целият файл.

Index: 0 1 2 3 4
Encrypted File: 68 bc e4 7a 32

Променливата `index` има стойност 4, защото последно 4-тият елемент е участвал в размяната, затова 4-тият елемент се разменя с този три позиции преди него. Това ще е единствената размяна, защото след това `i` от вторият 'for'-цикъл, където `i=index` след тази размяна придобива стойност 1, което означава, че за този пример само това ще бъде размяната.

2. Третият 'for'-цикъл разменя позициите на всеки трети елемент с този три позиции преди него.

Index: 0 1 2 3 4
Encrypted File: 68 32 e4 7a bc

Original File: e4 32 68 7a bc



След втората стъпка от процеса на криптиране

Encrypted File: 68 bc e4 7a 32

3. По този начин получаваме отново оригиналния файл

Encrypted File: e4 32 68 71 bc

Original File: e4 32 68 7a bc

Фигура 14 – визуализация на третата стъпка от процеса на декриптиране

fourthStepOfDeryption (*List<string> file, List<string> key*) - този метод представлява четвъртата стъпка от процеса на декриптиране. Четвъртата стъпка има за цел да разшифрова първата стъпка от процеса на криптиране. При процеса на декриптиране всичко се извършва в обратен ред спрямо криптирането.

- Методът има два параметъра, които са списъци от низове съдържащи данните на файла и на паролата, представени в шестнадесетичната бройна система. (Виж фиг.14, фиг.15)
- `for` – цикълът цикли през целият файл, докато броячът не стане по-голям или равен на броят на елементите на паролата. Понеже идеята на този метод е да премахне паролата от файлът и да остане само дешифрираният файл, а паролата е залепена на края на файла, затова премахваме един по един елементите започвайки от последният и при всеки премахнат елемент броячът се увеличава с едно, по този начин можем да



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

следим колко символи от паролата са премахнати. Понеже ние знаем броят на елементите на паролата, този брой го сравняваме всеки път с броячът и когато броячът стане по-голям или равен на броя на паролата спираме цикъла и метода, защото паролата вече е премахната от списъка и по този начин остава само декриптираният файл. (Виж фиг.14, фиг.15)

```
//Removes all the symbols from the key so only the original file will be left
1 reference
private void fourthStepOfDeryption(List<string> file, List<string> key)
{
    //so we remove the last symbols which represent the key from the
    //file so only the file content to stay
    int br = 0;
    for (int i = file.Count - 1; i >= 0; i--)
    {
        if (br < key.Count)
        {
            file.RemoveAt(i);
            br++;
        } else
        {
            return;
        }
    }
}
```

Фигура 15 – четвъртата стъпка от процеса на декриптиране



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

Визуализация на четвъртата стъпка от процеса на декриптиране

1. За да премахнем паролата и да остане само файлът, започваме да махаме елементи отзад напред, като при всеки премахнат елемент променливата 'br', която има функционалността на брояч отброява колко елементи са изтрети. Дължината на паролата има стойност 'key.Count', следователно когато броячът стигне броят на елементите 'key.Count', цикълът спира и по този начин успешно премахваме паролата от файла, като остава само декриптираният файл.

	File	RotatedPass
File + Rotated Password:	e4 56 23	b7 74 82 71



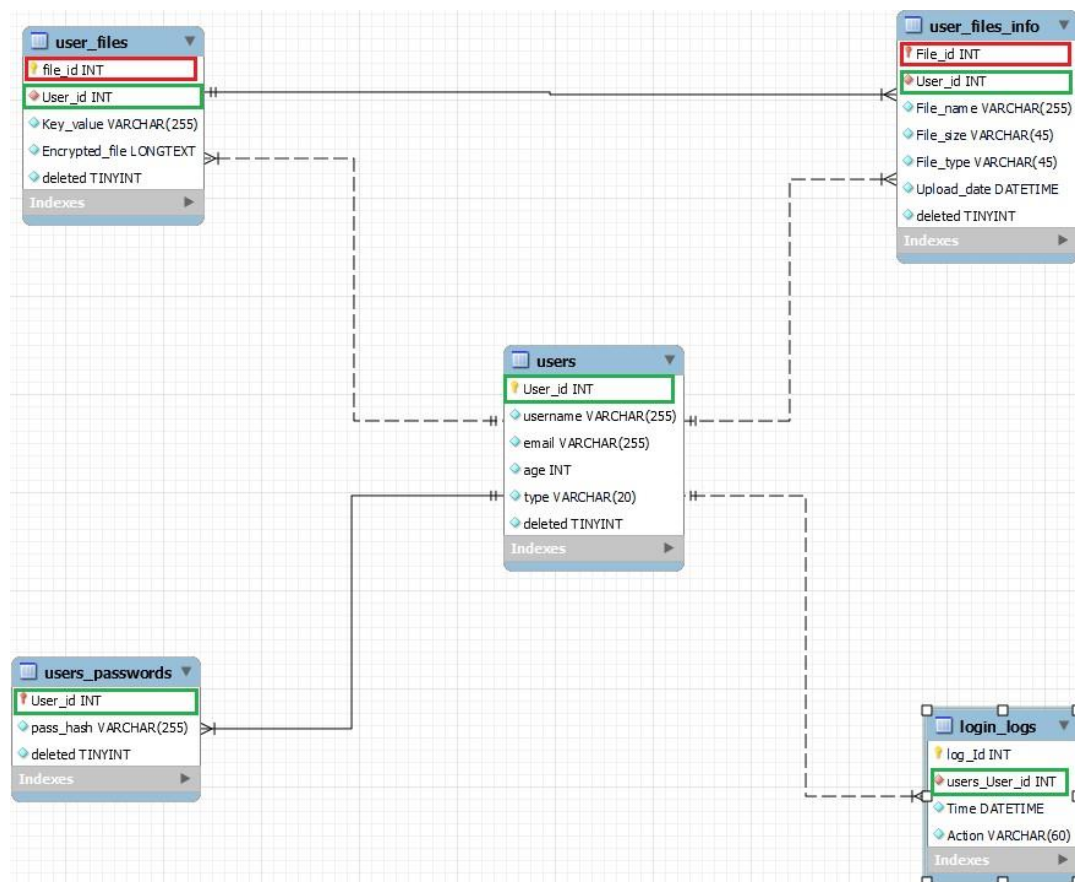
Decrypted File: e4 56 23

Фигура 16 – визуализация на четвъртата стъпка от процеса на декриптиране



3.6. Дизайн на базата данни

Базата данни се състои от 5 таблици: (Виж фиг.17)



С червено са показани връзките между file_id.

В зелено са показани връзките между User_id.

Фигура 17 – EER диаграма на базата данни



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

- **users** – съдържа данни за потребителите и се състои от следните колони:
 - **User_id** – първичен ключ на таблицата, който се инкрементира сам. Първичният ключ е уникален за всеки потребител и се използва като външен ключ в другите таблици.
 - **username** – уникално потребителско име с максимум 255 символа
 - **email** – уникален имейл на потребител с максимум 255 символа
 - **age** – години на потребителя тип ‘Integer’
 - **type** – привилегия на потребителя (normal user/admin) с максимум 20 символа.
 - **deleted** – показва дали потребителят е изтрит или съществува чрез TINYINT (0 = съществуващ потребител; 1 = изтрит потребител)
- **users_passwords** – съдържа паролата на потребителите, чрез която те влизат в своя потребителски акаунт. Състои се от следните колони:
 - **User_id** – първичен ключ на таблицата, който се инкрементира сам, като този идентификатор е външен ключ към таблицата ‘users’.
 - **pass_hash** – съдържа хешнатата парола на потребителят с максимум 255 символа.
 - **deleted** - показва дали потребителят е изтрит или съществува чрез TINYINT (0 = съществуващ потребител; 1 = изтрит потребител)
- **login_logs** – съдържа хронологията от извършените действия от потребителите. Състои се от следните колони:
 - **log_id** – първичен ключ на таблицата, който се инкрементира сам.
 - **users_User_id** – идентификатор, който е външен ключ към таблицата ‘users’, показващ кой потребител извършва даденото действие.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

- Time – часът и дата, в която потребителят е извършил действието. Данните се съхраняват чрез типа 'DATETIME'
- Action – действието, което потребителят е извършил с максимум 60 символа.
- **user_files** – съдържа криптираните файлове на потребителите, заедно с паролите, с които файловете са заключени. Състои се от следните колони:
 - file_id – първичен ключ на таблицата, който се инкрементира сам. Връзката между 'user_files' и 'user_files_info' се осъществява чрез този първичен ключ, който е външен ключ в 'user_files_info'. Чрез този идентификатор можем да достъпваме файла и информацията за файла през 'file_id'.
 - User_id – идентификатор, който е външен ключ към таблицата 'users', показващ кой потребител е криптирал този файл.
 - Key_value – съдържа паролата, с която потребителя е заключил своя файл, като паролата е хеширана.
 - Encrypted_file – съдържа криптираният от потребителя файл, кодирани в 'Base64'. 'Base64' представлява група от схеми за кодиране на двоични данни, като ги трансформира в символи лесни за отпечатване.
 - deleted - показва дали потребителят е изтрит или съществува чрез TINYINT (0 = съществуващ потребител; 1 = изтрит потребител)
- **user_files_info** – съдържа допълнителна информация за криптираните файлове от потребителите. Състои се от следните колони:
 - File_id – първичен ключ на таблицата, който се инкрементира сам, който е външен ключ към таблицата 'user_files', по този начин файла и информацията на файла, се достъпват през 'file_id'.
 - User_id – идентификатор съхраняващ идентификаторът от таблица 'users', показващ кой потребител е криптирал този файл.
 - File_name – съдържа името на криптирания файл както и неговото местоположение в директориите.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

- File_size – съдържа размерът на криптираният файл.
- File_type – съдържа азширението на файла, който потребителят е криптирал.
- Upload_date – съдържа часът и датата, в която потребителят е криптирал своя файл. Данните се съхраняват чрез типа 'DATETIME'
- deleted - показва дали потребителят е изтрит или съществува чрез TINYINT (0 = съществуващ потребител; 1 = изтрит потребител)

3.7 CRUD заявки

- SELECT заявка – този тип заявка се използва за извличане на информация от базата данни. Ще разгледам един от методите, който използва 'SELECT' заявка за проверка дали потребителят е администратор. (Виж фиг.18)
 - Методът 'checkIfUserIsAdmin(int userId)' – има един параметър и това е идентификаторът на потребителя, който ще проверяваме дали има администраторски привилегии. (Виж фиг.18)
 - Първо се създава връзка с базата данни чрез 'MySQLConnection', като се предоставя адреса на сървъра (в случая той е локален), името на базата и потребител и парола. (Виж фиг.18)
 - След което се формира SQL заявка. В случая тази заявка извлича всички потребители от таблицата 'users', който имат идентификатор равен на този, по който търсим и не са изтрети. По този начин извличаме информацията от таблица 'users' за даден потребител. (Виж фиг.18)



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

- Заявката се изпълнява с 'MySQLCommand' и резултатът се съхранява чрез 'MySQLDataReader'. Чрез 'MySQLDataReader' успяваме да прочетем какви привилегии има потребителят и ако те са администраторски, методът връща булева стойност 'true', ако не са връща 'false'. По този чрез CRUD заявката 'SELECT' извлякохме информация от базата и успяхме успешно да я използваме. (Виж фиг.18)

```
//Method for checking if the user is admin
1 reference
private bool checkIfUserIsAdmin(int userId)
{
    string connstring = "Server=localhost;Database=mydb;User=normaluser;Password=normalusernormaluser;";
    MySqlConnection CurrentConnection = new MySqlConnection(connstring);
    CurrentConnection.Open();

    string query = "SELECT type FROM users WHERE User_id=@userId AND deleted=@Deleted;";
    MySqlCommand cmd = new MySqlCommand(query, CurrentConnection);
    cmd.Parameters.AddWithValue("@userId", userId);
    cmd.Parameters.AddWithValue("@Deleted", 0);
    MySQLDataReader reader = cmd.ExecuteReader();

    if (reader.Read() && reader[0].Equals("admin"))
    {
        reader.Close();
        CurrentConnection.Close();
        return true;
    }
    else
    {
        reader.Close();
        CurrentConnection.Close();
        return false;
    }
}
```

Фигура 18 – употреба на CRUD заявката 'SELECT'.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

- INSERT заявка – този тип заявка се използва за записване на информация в базата данни. Ще разгледам един от методите, който използва ‘INSERT’ заявка за да запише в таблицата ‘login_logs’, че нов потребител се е регистрирал. (Виж фиг.19)
 - Методът ‘createLog(int User_id, string action)’ – има два параметъра, които са идентификатора на потребителя и какво действие е извършил. (Виж фиг.19)
 - Първо се създава връзка с базата данни чрез ‘MySQLConnection’, като се предоставя адреса на сървъра (в случая той е локален), името на базата и потребител и парола. (Виж фиг.19)
 - След което се формира SQL заявка. В случая тази заявка ще записва идентификатор на потребителя, моментния час и дата и действието, което потребителят е извършил в таблицата ‘login_logs’. (Виж фиг.19)
 - Заявката се изпълнява с ‘MySQLCommand’, като се извиква методът ‘ExecuteNonQuery()’, който връща броя на засегнатите редове от операцията (в случая, броя на добавените записи). Ако броят на засегнатите редове е по-голям от 0, операцията е успешна и данните за записани успешно в базата, ако засегнатите редове са нула, това означава, че операцията се е провалила и методът връща булевата стойност ‘false’. (Виж фиг.19)



```
//Method for making a log in login_logs
2 references
private bool createLog(int User_id, string action)
{
    string connstring = "Server=localhost;Database=mydb;User=normaluser;Password=normalusernormaluser;";
    MySqlConnection CurrentConnection = new MySqlConnection(connstring);
    CurrentConnection.Open();

    string query = "INSERT INTO login_logs (users_User_id, Time, Action) VALUES (@User_id, @Time, @Action)";
    MySqlCommand cmd = new MySqlCommand(query, CurrentConnection);

    cmd.Parameters.AddWithValue("@User_id", User_id);
    cmd.Parameters.AddWithValue("@Time", DateTime.Now);
    cmd.Parameters.AddWithValue("@Action", action);

    int rowsAffected = cmd.ExecuteNonQuery();
    if (rowsAffected > 0)
    {
        Console.WriteLine("Data inserted");
    }
    else
    {
        MessageBox.Show("Failed to insert data");
        return false;
    }
    CurrentConnection.Close();
    return true;
}
```

Фигура 19 – употреба на CRUD заявката ‘INSERT’

- UPDATE заявка – този тип заявка се използва за промяна на информация записана в базата данни. Ще разгледам един от методите, който използва „UPDATE“ заявка за да редактира запис в таблицата ‘users_passwords’. (Виж фиг.20)
 - Методът ‘updateUserPassByUserId (int id, string pass_hash)’ – има два параметъра, които са идентификатора на потребителя и хешираната парола. (Виж фиг.20)
 - Първо се създава връзка с базата данни чрез ‘MySqlConnection’, като се предоставя адреса на сървъра (в случая той е локален), името на базата и потребител и парола. (Виж фиг.20)
 - След което се формира SQL заявка. В случая тази заявка ще презапише хешираната парола в таблицата ‘users_passwords’, с тази подадена в параметъра, за потребителя с идентификатор равен на този, който е подаден чрез параметъра и потребителят има



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

стойност 0 при колоната 'deleted', което означава, че потребителят съществува и не е изтрит. (Виж фиг.20)

- Заявката се изпълнява с 'MySQLCommand', като се извиква методът 'ExecuteNonQuery()', който връща броя на засегнатите редове от операцията (в случая, броя на добавените записи). Ако броят на засегнатите редове е по-голям от 0, операцията е успешна и данните за записани успешно в базата, ако засегнатите редове са нула, това означава, че операцията се е провалила и методът връща булевата стойност 'false'. (Виж фиг.20)

```
//Method for changing the password
1 reference
private bool updateUserPassById(int id, string pass_hash)
{
    string connstring = "Server=localhost;Database=mydb;User=normaluser;Password=normalusernormaluser;";
    MySqlConnection CurrentConnection = new MySqlConnection(connstring);
    CurrentConnection.Open();

    string query = "UPDATE users_passwords SET pass_hash=@hash WHERE User_id=@userid AND deleted=@delete;";
    MySqlCommand cmd = new MySqlCommand(query, CurrentConnection);
    cmd.Parameters.AddWithValue("@hash", pass_hash);
    cmd.Parameters.AddWithValue("@userid", id);
    cmd.Parameters.AddWithValue("@delete", 0);

    int rowsAffected = cmd.ExecuteNonQuery();
    if (rowsAffected > 0)
    {
        Console.WriteLine("Insert successful");
        CurrentConnection.Close();
        return true;
    }
    else
    {
        MessageBox.Show("Insert failed");
        CurrentConnection.Close();
        return false;
    }
}
```

Фигура 20 – употреба на CRUD заявката 'UPDATE'



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

- DELETE заявка - този тип заявка се използва за изтриване на записи в таблица или изтриване на цяла таблица в базата данни. Ще разгледам един от методите, който използва 'DELETE' заявката за да изчисти всички записи в таблицата 'login_logs'. (Виж фиг.21)
 - Методът 'clearAllLogs()' – няма параметри и е от типа void. (Виж фиг.21)
 - Първо се създава връзка с базата данни чрез 'MySQLConnection', като се предоставя адреса на сървъра (в случая той е локален), името на базата и потребител и парола. (Виж фиг.21)
 - След което се формира SQL заявката, която изтрива всички записи от таблицата 'login_logs'. (Виж фиг.21)
 - Заявката се изпълнява с 'MySQLCommand', като се извиква методът 'ExecuteNonQuery()', който връща броя на засегнатите редове от операцията (в случая, броя на добавените записи). Ако броят на засегнатите редове е по-голям от 0, операцията е успешна и данните за записани успешно в базата, ако засегнатите редове са нула, това означава, че операцията се е провалила и методът връща булевата стойност 'false'. (Виж фиг.21)



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

```
private void ClearAllLogs()
{
    string connstring = "Server=localhost;Database=mydb;User=adminuser;Password=adminuser1234!";
    MySqlConnection CurrentConnection = new MySqlConnection(connstring);
    CurrentConnection.Open();

    string query = "DELETE FROM login_logs;";
    MySqlCommand cmd = new MySqlCommand(query, CurrentConnection);

    int rowsAffected = cmd.ExecuteNonQuery();
    if (rowsAffected >= 0)
    {
        Console.WriteLine("Insert successful");
        CurrentConnection.Close();
    }
    else
    {
        Console.WriteLine("Insert failed");
        CurrentConnection.Close();
        MessageBox.Show("Errorr occured cleaning the logs!");
        return;
    }
}
```

Фигура 21 – употреба на CRUD заявката ‘DELETE’



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

4. Заключение

4.1. Постигнати цели

По време на разработката на десктоп приложението, се изправих пред значителни предизвикателства, особено при изграждането на криптиращия алгоритъм. Изработката на дизайна на този алгоритъм и неговата успешна имплементация представляваха основното предизвикателство. С радост мога да заявя, че успях да постигна тази цел. В процеса на работа придобих нови знания и умения в областта на разработката на десктоп приложения, интеграцията на приложението с база данни и използването на различни библиотеки.

4.2. Перспективи за развитие

Десктоп приложението за криптиране на файлове има голяма перспектива за развитие в бъдещето. Идеята за такова приложение не е измислена от мен и в интернет има доста реализирани такива приложения. Главната цел е да се увеличи сигурността на всеки един потребител и да се намали изтичането на чувствителна информация. Световните загуби от киберпрестъпления за 2025 се предвижда да бъдат 10.5 трилиона американски долара според *cybercrime magazine* (<https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/>) и повечето от тези престъпления стават възможни благодарение на чувствителни данни, които биват откраднати от потребителите. Много потребители държат уязвима информация на своя личен компютър/телефон, който ако бъде експлоатиран с дадена програма или вирус тези данни стават лесно достъпни за хакерите, но когато тези данни са криптирани и само потребителят знае паролата, чрез която може да декриптира тези файлове, може да запази своите данни. Включително ако физически откраднат личното устройство на даден потребител, неговите данни пак са в сигурност. Затова смятам, че този тип приложения могат да помогнат в бъдеще на хората да се преборят с кибератаките. От друга страна за момента приложението



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

може да криптира файлове до 2 GB размер, в бъдеще за комерсиална цел може да се изисква месечна такса от потребителя в замяна на която да получава възможност за криптиране на цяла папка с файлове. По-големи файлове/папки биха желали да криптират потребителите, които разработват проекти и искат да запазят неговата конфиденциалност, по този начин чрез такса веднъж месечно тази функционалност може да им бъде предоставена.



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

5. Източници

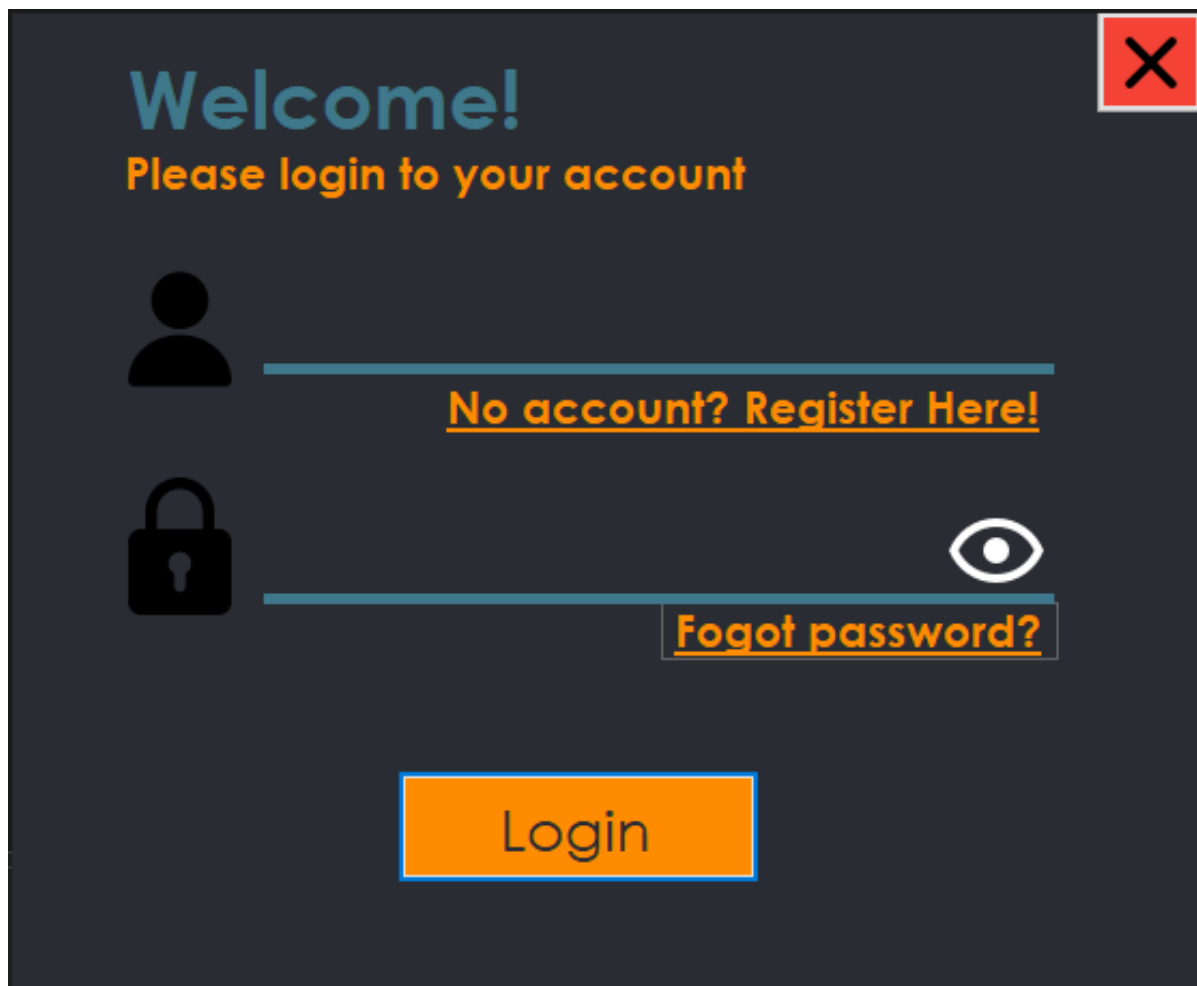
- <https://www.youtube.com/watch?v=K9Ps66GoD-k&t=61s> – Дизайн на десктоп приложение
- <https://www.youtube.com/watch?v=d1aLhXXEofU&t=2s> – Свързване с MySQL база от данни
- <https://www.youtube.com/watch?v=JSJ1JI2aIJg> – Създаване на десктоп приложение с C#
- <https://www.youtube.com/watch?v=tEhGIYN4vic> – Работа с MySQL Workbench и използването на EER диаграми
- <https://www.youtube.com/watch?v=xd9OeA2hYgY> – Връзки между таблиците в MySQL Workbench
- <https://www.youtube.com/watch?v=zLTAXQW96Do> – Основни принципи на алгоритъм за криптиране
- <https://www.youtube.com/watch?v=NuyzuNBFWxQ> – Основни понятия в криптирането
- <https://www.cryptool.org/en/cto/> - Сайт показващ как други алгоритми за криптиране работят
- <https://www.youtube.com/watch?v=aer8S1fFbNc&t=20s> - CRUD заявки с MySQL в C#



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

6. Приложения



Приложение 1 – Login Page



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

The image shows a 'Register' form with a dark grey background. At the top left is a white back arrow icon. The title 'Register' is in a light blue font. The form contains the following fields and controls:

- Username:** A text input field with an orange underline.
- Password:** A text input field with an orange underline and a white eye icon to its right for toggling visibility.
- Repeat password:** A text input field with an orange underline and a white eye icon to its right for toggling visibility.
- Email:** A text input field with an orange underline.
- Age:** A text input field with an orange underline.
- Role Selection:** Two radio buttons with labels:
• Administrator (selected)
• Normal user
- Register! Button:** An orange rectangular button with a blue border and the text 'Register!' in bold black font.

Приложение 2 – Register Page



„Десктоп приложение за засекретено съхраняване на файлове“

Виктор Дамянов Владинов

← **Forgotten password**

Please enter your username:

Reset password

Приложение 3 – Forgot Password Page

FilesAver
"Where Security Meets Simplicity"

Encrypt/Decrypt File

My Account

Admin tools

Log out

Encrypt file

1. Enter a password for encryption

The minimum length must be at least 16 characters!
- 2. Browse file that will be encrypted
 Browse File For Encryption
Chosen file:
- 3. Encrypt the file!
 Encrypt

Decrypt file

1. Enter a password for decryption

The minimum length must be at least 16 characters!
- 2. Browse file that will be decrypted
 Browse File For Decryption
Chosen file:
- 3. Decrypt the file!
 Decrypt


Приложение 4 – Main Page





„Десктоп приложение за засекретено съхраняване на файлове“


Виктор Дамянов Владинов

FilesAver
"Where Security Meets Simplicity"


Encrypt/Decrypt File


My Account


Admin tools


Log out

Your account


Change your current data

Username:

Email:

Age:

Account type: **admin**





These are your encrypted files


Your account doesn't have any encrypted files yet. Encrypted files will show up here.


Change password

Change your current password with new one

Old Password 

New password 

Confirm New Password 




Приложение 5 – My Account Page





„Десктоп приложение за засекретено съхраняване на файлове“


Виктор Дамянов Владинов

Files△ver
"Where Security Meets Simplicity"

Encrypt/Decrypt File

My Account

Admin tools

Log out

Edit account

Edit account's data


Select user:

Username:

Email:

Age:


Account type: **fafaf**

 Save changes

Delete account



Delete any account

Select user:

 Delete user

Users logs

See the actions performed by the users

 Remove logs  Show all logs [Apply filters](#)

Username: Az	Time at the action: 4/14/2024 2:49:44 PM	Action: Log out	<input type="checkbox"/> Log in
Username: Az	Time at the action: 4/14/2024 3:41:24 PM	Action: Logged in	<input type="checkbox"/> Log out
Username: Az	Time at the action: 4/14/2024 3:41:25 PM	Action: Logged in	<input type="checkbox"/> registration
Username: Az	Time at the action: 4/14/2024 3:45:03 PM	Action: Logged in	<input type="checkbox"/> Encrypted files
Username: Az	Time at the action: 4/14/2024 3:45:18 PM	Action: Log out	<input type="checkbox"/> Decrypted files
Username: Az	Time at the action: 4/14/2024 3:47:37 PM	Action: Logged in	<input type="checkbox"/> Failed to Log in
			<input type="checkbox"/> Account deleted
			<input type="checkbox"/> Changed password

Приложение 6 – Admin Tools Page