Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Гаврилюк Богдан Вікторович,
студент групи РЗ-181

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Розробка програми-клієнта для зберігання паролів у зашифрованому вигляді за допомогою сторонніх сервісів

Спеціальність:
125 Кібербезпека

Спеціалізація, освітня програма:
Кібербезпека

Керівник:
Зоріло Вікторія Вікторівна,
ст. викладач

Одеса – 2022

Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Спеціальність 125 – Кібербезпека
Освітня програма – Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри КБПЗ

_____
д.т.н.,проф. А.А.Кобозєва
____ _____2022р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Гаврилюку Богдану Вікторовичу

1.Тема роботи: *Розробка програми-клієнта для зберігання паролів у зашифрованому вигляді за допомогою сторонніх сервісів*,
керівник роботи *Зоріло Вікторія Вікторівна, ст. викладач,*
затверджені наказом ректора від 17.05.22 р. №168-в .

2.Зміст роботи*: аналіз проблемної області, постановка задачі та аналіз існуючих рішень, реалізація програмного продукту.*

3. Перелік ілюстративного матеріалу: *блок-схема алгоритму шифрування та дешифрування AES, ілюстрація програмного продукту, слайди презентації*.

4. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
Охорона праці	к.т.н, доцент Ярова І.А.	Завдання видав	Завдання прийняв
		6.05.2022	20.05.2022

5. Дата видачі завдання "_____"_____2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	*Аналіз джерел з теми випускної кваліфікаційної роботи*	*18.11.2021*	*виконано*
2	*Обґрунтування вибору рішення.* *Збір даних*	*15.01.2022*	*виконано*
3	*Аналіз необхідного функціоналу програмного продукту*	*20.02.2022*	*виконано*
4	*Реалізація програми-клієнт*	*13.05.2022*	*виконано*
5	*Підготовка тексту роботи*	*15.05.2022*	*виконано*
6	*Підготовка презентації та доповіді*	*8.06.2022*	*виконано*
7	*Попередній захист*	*9.06.2022*	*виконано*
8	*Нормоконтроль, рецензування*	*20.06.2022*	*виконано*
9	*Допуск до захисту у завідувача кафедри*	*22.06.2022*	*виконано*

Здобувач вищої освіти _____ *Гаврилюк Б.В.*

Керівник роботи _____ *Зоріло В.В.*

ЗАВДАННЯ

на розробку розділу «Охорона праці»

Гаврилюку Богдану Вікторовичу, група РЗ-181

Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій

Кафедра кібербезпеки та програмного забезпечення

Тема роботи: *Робоче місце інженера-програміста*

Зміст розділу:

1 Аналіз умов праці і вибір основних заходів виробничої безпеки.

2 Аналіз пожежної безпеки. Вибір заходів та засобів пожежної безпеки.

Керівник роботи

_____ (Зоріло В.В.)

«____» _____ 2022 р.

Консультант з охорони праці

_____ (Ярова І.А)

«____» _____ 2022 р.

АНОТАЦІЯ

Кваліфікаційна робота на тему «Розробка програми-клієнта для зберігання паролів у зашифрованому вигляді за допомогою сторонніх сервісів» на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 – Кібербезпека, спеціалізація, освітня програма: Кібербезпека, містить 11 рисунків, 1 таблицю, 1 додаток, 22 літературних джерела за переліком посилань. Робота виконана на 49 сторінках загального тексту і 33 сторінках основного тексту.

Метою роботи є підвищення ефективності зберігання та захисту даних користувача шляхом розробки програмного продукту та шифрування інформації.

Результати даної роботи можуть бути використані для індивідуального використовування особою для збереження своїх паролів чи іншої інформації.

Можливими напрямками подальших досліджень є збільшення функціоналу програми.

КІБЕРБЕЗПЕКА, КРИПТОГРАФІЯ, ПАРОЛЬНИЙ МЕНЕДЖЕР, PYTHON.

ANNOTATION

Qualification work on "Development of a client program for storing passwords in encrypted form using third-party services" for the first (bachelor's) level of higher education in the specialty 125 - Cybersecurity, specialization, educational program: Cybersecurity Management, contains 11 figures, 1 table, 1 appendix, 22 references according to the list of references. The work is performed on 49 pages of general text and 33 pages of main text.

The aim of the work is to increase the efficiency of storage and protection of user data by developing a software product and encrypting information

The results of this work can be used for individual use by a person to save their passwords.

Possible areas of further research are to increase the functionality of the program.

CYBER SECURITY, CRYPTOGRAPHY, PASSWORD MANAGER, PYTHON

ЗМІСТ

ВСТУП

З розвитком технологій суспільство перетворилося на так зване «Інформаційне суспільство». Майже всі люди користуються інформаційними технологіями, як програмними, так і апаратними. Інформація стала одним із найважливіших ресурсів. В кожного є деяка інформація, яку не варто розголошувати з метою безпеки. Наприклад, дані банківських карт або доступ до персонального кабінету якоїсь інтернет-платформи.

Зазвичай для ідентифікації користувача використовують паролі, які ж сам користувач і встановлює. Але існує велика кількість платформ, якими сучасна людина користується і в більшості кожна з платформ пропонує захистити доступ до своїх даних. Через це виникає деяка проблема, що велику кількість паролів важко завжди тримати в голові, встановлювати всюди один і той самий пароль небезпечно, записувати на листочок також не є раціональним з міркувань безпеки та зберігання.

Існуючі рішення цієї проблеми потребують матеріальних витрат на купівлю програмного чи апаратного забезпечення. Тому в цій кваліфікаційній роботі було розглянуто створення власного програмного продукту для вирішення поставленої проблеми.

Розроблений програмний продукт забезпечує спрощення задачі збереження паролів до запам’ятовування лише одного, це набагато легше ніж тримати в голові велику кількість різних паролів.

Сутність програми полягає у тому, що в одному з найпопулярніших веб-додатків «Telegram» розміщується зашифрований файл. Отримати доступ до цього файлу можливо лише за допомогою розробленої програми. Навіть якщо хтось отримує доступ до вашого акаунту, то відкрити файл з паролями неможливо. Великим плюсом є те, що програма є безкоштовною та кросплатформеною, тобто вона може функціонувати на різних пристроях та операційних системах.

В будь який момент користувач може скористатися необхідним паролем, для цього треба лише запам’ятати один основний пароль для доступу до зашифрованого файлу.

Основними задачами для досягнення поставленої в кваліфікаційній роботі мети є наступні:

- аналіз існуючих рішень вирішення проблеми;
- вибір середовища для реалізації, мови програмування та алгоритму шифрування для збереження даних;
- реалізація програми-клієнта та впровадження її в експлуатацію.

Результатом кваліфікаційної роботи є розроблена програма-клієнт, яка виконує збереження, шифрування, дешифрування файлу з даними користувача за допомогою алгоритму шифрування AES.MODE EAX-файлу та забезпечує комунікацію користувача з даними через інтерфейс.

Отримані результати опубліковано у фаховому виданні «Інформатика та математичні методи в моделюванні» [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблематика. Захист даних в інформаційному просторі

Інформаційний простір (надалі ІП) – це простір, в якому циркулює інформація. В ІП інформація створюється, модифікується та споживається основними учасниками та паралельно творцями – людьми.

З розвитком технологій контактування з ІП людини стає невід'ємною частиною життя. Внаслідок цього більшість важливих даних для людини було перенесено в ІП. Це має великі плюси, оскільки деякі повсякденні задачі, що раніше займали багато часу, можна вирішити в один клік. Та, як завжди, є зворотній бік медалі. Він пов'язаний з інформаційною безпекою.

Інформаційна безпека – сукупність правил та практичних методів, спрямованих на захист певної інформації. Це поняття є універсальним і розповсюджується на будь-які данні, незалежно від їх форми.

Захист інформації – це діяльність, спрямована на те, щоб запобігти витоку, несанкціонованому доступу, втратам, підробці (модифікації) захищеної інформації.

Витік інформації – це певна діяльність в системі, в результаті якої інформація стає доступною для осіб, які не повинні мати до неї доступу.

Несанкціонований доступ до інформації – це протиправна діяльність, результатом якої стає доступ до інформації зі сторони зловмисника.

Втрата інформації – діяльність, в результаті якої інформація знищується частково або повністю.

Підробка (модифікація) інформації – діяльність, в результаті якою зміст чи об'єм інформації став відрізнятися від оригінала.

Зловмисник – це людина чи програма, діяльність яких спрямована на викрадення, модифікацію, несанкціонований доступ інформаційних даних ПК.

Треба також зважати на те, що чим інформація цінніша для зловмисника, тим більше вірогідність, що він спробує забрати її у власника.

Тому в інформаційній безпеці вже існують та продовжують розроблятися нові методи захисту, та контролю доступу до інформації. Одним із таких методів є використання паролю. Паролі умовно поділяються на три категорії: слабкий, середній, сильний. «Сила» паролю залежить від багатьох факторів, наприклад:

- оригінальність структури формування паролю;
- його розмір (кількість символів);
- різноманіття символів, що використовуються (літери, цифри, спецсимволи, регістр).

Використання достатньо сильного паролю забезпечить надійний захист даних, які закриваються обраним текстовим паролем.

1.2 Аналіз поняття «сильний пароль»

Пароль – це таємний певний набір символів, головна функція якого є забезпечення перевірки особи, що бажає отримати дані. Перевірка відбувається з тим розрахунком, що тільки певна людина чи група людей повинні знати пароль, так би мовити «ключ доступу» для отримання захищених даних.

Перевірка ключа доступу, який надає користувач, і є основна функція автентифікації. Тобто задача автентифікації – це надання необхідної інформації лише тій людині чи групі людей, яка/які мають на неї право.

Враховуючи вищевикладене можливо зробити висновок, що одним із найпопулярніших методів забезпечення автентифікації є текстовий пароль. Це досить традиційний метод, який має як деякі недоліки, так і переваги серед інших видів автентифікації.

З переваг можна виділити простоту та доступність введення символів текстового паролю для автентифікації користувача. Як результат простота та доступність дозволяє залишатися текстовому виду паролів одним із найпопулярніших методів автентифікації.

Що до недоліків текстового паролю, то пароль повинен легко запам’ятовуватись, але важко «вгадуватись», щоб ніхто інший крім користувача не зміг отримати «випадково» доступ до даних.

Але, як показує практика, паролі, які легко запам’ятати, зазвичай короткі чи базуються на якійсь значущий події для користувача, наприклад – день народження, або день закінчення університету. Короткі паролі підбираються методом перебору символів, це потребує великої кількості часу, але все ж є можливим. Паролі, що встановленні користувачем і співпадають з якоюсь важливою подією чи чимось іншим подібного характеру, можна підібрати, прослідкувавши за користувачем, або проаналізувавши його особисті дані, що знаходяться у відкритому доступі (анкети соціальних мереж, мессенджерів тощо).

Паролі вважаються одним із елементів безпеки в інформаційних системах, який найбільше вразливий до підбору або зламу, що робить його основною ціллю для здійснення успішної кібератаки.

В основному така вразливість є причиною людської поведінки і практично не має відношення до самої системи. Спеціалісти називають це «людським фактором». Людський фактор є причиною більшості проблем та успішних кібератак, що відбуваються в інформаційному просторі.

Що до встановлення сильного паролю, то більшість людей ігнорує прості правила відносно вибору паролю. Прийнято вважати, що пароль повинен включати в себе поєднання різних символів клавіатури (букви, спеціальні символи, цифри). Пароль не повинен містити значущих слів зі словника.

Незважаючи на доступність правил та порад щодо встановлення сильного паролю, більшість користувачів вибирають легкі паролі, які не дають належного захисту.

Далі будуть наведені приклади створення «сильного» паролю самостійно:

- візьмемо просте слово, яке легко перекладається на англійську та має не менше 4-6 символів, наприклад «яблуко»;
- транслітеруємо це слово або перекладаємо на англійську мову: «yabloko» чи «apple»;
- тепер підберемо певну комбінацію цифр, достатньо буде від 3 до 4 цифр. Бажано, щоб цифрі не йшли підряд одна за одною, а були випадкові, без повторень;

- далі обираємо ще декілька символів, можна взяти перші три символи з назви сайту, чи назви якогось предмету, тощо. Наприклад оберемо: «TEL» чи «RAM»;

- тепер оберемо декілька спеціальних символів, наприклад: «!» і «$»;

- далі скомпонуємо всі отримані елементи паролю в будь-якому порядку: «yabloko!$RAM601», «748TEL!apple$».

Вочевидь, що короткі та осмислені паролі, краще запам’ятовуються, а вибір довгих і хаотичних призводить до того, що людині необхідно зберігати десь паролі.

Хтось записує на папірці, але папірець чи губиться, чи потрапляє до зловмисника. Хтось зберігає їх в нешифрованому файлі на флешці, чи ПК, що також має негативні наслідки. Флешка чи ПК можуть вийти з ладу або зловмисник отримає доступ до вашого ПК чи флешки, і якщо файли з паролями будуть не шифровані, то зловмисник «зірве куш».

Одне із результативних рішень для захисту великої кількості паролів – використання менеджеру паролів.

1.3 Огляд та аналіз існуючих програм, спрямованих на захист паролів

Одним із найзручніших рішень для зберігання та захисту великої кількості паролів є використання менеджеру паролів.

Менеджер паролів – програмне забезпечення, що дозволяє користувачу працювати з паролями. В основному використовується для збереження паролів в одному місці: базі даних чи на якомусь пристрої.

На сьогоднішній день існує велика кількість програмного забезпечення спрямованого на захист паролів користувача. Перед початком розробки свого програмного продукту, необхідно розглянути вже існуючі на ринку, та проаналізувати їх роботу.

Одними із найпопулярніших менеджерів паролів є наступні програмні продукти: «1Password», «Dashlane», «Bitwarden», «LastPass» «Keeper», «Zoho Vault».

Проаналізувавши дані зазначені в таблиці 1.1, маємо можливість зробити висновок, що навіть найпопулярніші менеджери паролів не мають особливих відмінностей в послугах, що надаються безкоштовно. А більшість корисних та унікальних послуг необхідно придбавати окремо.

Таблиця 1.1 – Аналіз та порівняння найпопулярніших програмних продуктів

Функції	1Password	Dashlane	Bitwarden	LastPass	Keeper	Zoho Vault
Підтримка Chrome	Так	так	Так	так	так	так
Підтримка Firefox	Так	так	Так	так	так	так
Підтримка Mac OS	Так	так	Так	так	так	так
Підтримка Windows	Так	так	Так	так	так	так
Підтримка Linux	слабо	слабо	Так	так	так	так
Консольний клієнт Mac OS	слабо	ні	Так	так	так	ні
Консольний клієнт Windows	слабо	ні	Так	слабо	так	ні
Консольний клієнт Linux	слабо	ні	Так	слабо	так	ні
Двофакторна автентифікація	так	так	Так	так	так	так
Синхронізація між пристроями	так	так	Так	так	так	так
Підтримка 2FA при авторизації у сховищі паролів	ні	ні	так	ні	ні	ні
Корпоративна ліцензія,	7,99$	4$	3$	6$	3,75$	3,6$

місяць						
Кількість падінь сервісу за останні 6 місяців	1	12	0	12	0	2

Слід зауважити, що в вище зазначеній таблиці розглядались лише послуги, що надаються сервісами на безкоштовній основі. Для отримання доступу до більш корисних та зручних послуг, необхідно оформити підписку, вартість якої також вказана в таблиці.

1.4 Обґрунтування створення свого програмного додатку

Більшість існуючих рішень для збереження паролів не є зовсім безкоштовними. За унікальні або необхідні послуги треба платити гроші. Тому створення свого персонального програмного продукту, який буде задовільняти потребам одного користувача є логічним рішенням для захисту особистих паролів.

Використання такого додатку має певні переваги, а саме: повна безкоштовність; отримання саме того функціоналу, який потрібен; вибір свого алгоритму шифрування; збереження бази даних саме там, де обирає користувач, що не дає зловмиснику точної цілі для атаки; шифрування і дешифрування відбувається на рівні пристрою, а не на сторонньому сервері; можливість модифікації програмного продукту, якщо з’явиться така необхідність.

У підсумку, можна сказати, що розроблення програмного продукту для індивідуального використання є одним із найкращих рішень для захисту своїх даних, оскільки це не дає змогу зловмиснику успішно спланувати хід кібератаки.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОД ДОСЛІДЖЕННЯ

2.1 Мета роботи

Метою роботи є підвищення ефективності зберігання та захисту даних користувача шляхом розробки програмного продукту та шифрування інформації. Головна задача – це програмна реалізація такого продукту. Щоб впевнитись в необхідності реалізації, було проведено аналіз вже існуючих рішень розглянутої проблеми. Із аналізу зроблено висновок, що жодний із вже існуючих рішень не задовольняє в повному обсязі потреби користувача.

Перед початком реалізації програмного продукту необхідно поставити та виконати такі задачі: вибір сервісу, який буде виступати в якості серверу, де буде розміщуватися шифрований файл з паролями; вибір мови програмування на якому буде реалізований програмний продукт; вибір середовища розробки в якій буде відбуватися проектування програми; вибір алгоритму шифрування.

Після чіткої постановки задач маємо можливість приступати до пошуку методів реалізації програми.

2.2 Обґрунтування вибору при вирішенні поставлених задач

2.2.1 Обґрунтування вибору сервісу

Оскільки розробка програмного продукту орієнтована на одного користувача, то для вибору сервісу, який буде виступати в якості бази даних, застосовувалися такі критерії:

- безкоштовність;
- кросплатформність, тобто щоб доступ до сервісу можна було отримати з будь-якого пристрою та з різних операційних систем;
- захищеність. Під цим будемо розуміти, що сервіс повинен мати свій власний захист даних, який ми можемо використати в свою користь;
- популярність. Використання відомих вже багатьом сервісів дозволить уникнути ускладнень відносно вивчення користувачем нового сервісу.

Враховуючи все вищевикладене було обрано веб-додаток «Telegram».

Telegram – це веб-додаток, головний функціонал якого полягає в забезпеченні швидкого та безпечного обміну повідомленнями та невеликими файлами між користувачами. Telegram являє собою повністю безкоштовним сервісом, який пропонує захищений доступ до даних та збереження листування в хмарному сервісі. На даний момент Telegram є одним із найпопулярніших веб-додатків, в день в ньому реєструється приблизно 350 тисяч облікових записів, а кількість доставлених повідомлень через Telegram досягає 14 мільярдів в день [2].

При реєстрації Telegram запрошує номер телефону користувача, на який прив’язується обліковий запис. Після чого відбувається активація облікового запису, на ваш номер приходить повідомлення від сервісу з кодом підтвердження номеру. Після вводу коду активація пропонується ввести дані про користувача (ім’я та прізвище), але вказувати справжні дані не обов’язково. Telegram також є кросплатформним веб-додатком, що дозволяє встановити його на будь-який пристрій чи операційну систему, які є в наявності у користувача. Принцип захисту даних в Telegram реалізується на базі власного розробленого протоколу MTProto [3]. Користувачі обмінюються інформацією за методом Діффі-Хеллмана [4]. Тобто створюється загальний ключ, який використовується для передачі даних. Комунікація користувачів із сервером здійснюється за встановленим публічним RSA-ключом [5], який прописаний усередині клієнта Telegram та змінюється досить рідко.

Як і в будь-якій технології, в Telegram є свої недоліки і їх також треба виділити, щоб впевнитись, що вони не мають негативного впливу на збереження паролів. Можна виділити наступні недоліки:

- як і інші веб-додатки подібного характеру, Telegram використовує хмару для збереження даних. Тобто якщо зловмисник отримає контроль на сервером, то отримає доступ (як мінімум) до незашифрованих даних. Оскільки наш файл з паролями буде зашифрований, то цей недолік не має критичного значення під час вибору сервісу;
- функція наскрізного шифрування не використовується за умовчанням [6]. Через це більшість користувачів, які не мають технічної освіченості

використовую веб-додаток без функції секретного чату і вважають, що повідомлення шифруються. Цей недолік також не має критичного значення під час вибору сервісу, оскільки файл не буде нікуди пересилатись, а лише розміщається в додатку;

- Telegram використовує протокол MTProto власної розробки. Більшість експертів скептично відносяться до такого методу і вважають, що команді телеграм не треба «винаходити велосипед». Як сказав спеціаліст із криптографії Метт Грін: «Telegram має 10 мільйонів деталей, які підтримують одиничний неавторизований обмін ключами за методом Діффі-Хеллмана» [7]. Ця проблема також не має значного впливу, оскільки розміщений файл з паролями зашифрований і нікуди не перекидається;

- третім недоліком є те, що Telegram з самого початку запрошує доступ до списку контактів користувача. Велика база даних номерів зберігається на сервері веб-додатку. Що дає змогу зловмиснику отримати її при зломі сервера або збої в роботі сервісів компанія Telegram призведе до витоку зазначеної інформації без відома як компанії так і користувачів. Але зазначений недолік також не має критичного значення під час вибору сервісу.

Отже, проаналізувавши всі дані відносно веб-додатку, можна підсумувати, що Telegram повність задовольняє всім критеріям які були поставлені для вибору сервісу. А визначені недоліки не мають жодного негативного впливу на працездатність та реалізацію програмного продукту для збереження паролів. Надалі можна переходити до вибору мови програмування на якому буде відбуватися реалізація програми.

2.2.2 Обґрунтування вибору мови програмування.

Перед вибором мови програмування слід виділити критерії за якими буде проводитися вибірка, а саме:

- простий синтаксис та наявність інтуїтивно зрозумілого написання коду;
- велика кількість вже вирішених задач на обраній мові програмування від простих до складних;

- за необхідності, можливість легко модифікувати програмний код чи його окрему частину;
- кросплатформність. Можливість компіляції коду на будь-якому пристрої чи операційній системі;
- наявність великої кількості бібліотек які можна використати для спрощення написання програми;
- підтримка об’єктно-орієнтованого програмування, для написання простого інтерфейсу.

Для реалізації програмного продукту був обраний язик програмування Python.

Python – це високорівнева скриптова мова програмування, яку використовують для рішення великої кількості різноманітних задач у безлічі сучасних галузей. Також Python орієнтований на легке розуміння коду розробником. Python використовують для розробки комп’ютерних програм та мобільних додатків, для взаємодії з великим обсягом даних та в розробці інших різноманітних проектів [8].

Python підтримує функціональне, структурне, об’єктно-орієнтоване програмування [8].

Із можливостей мови програмування можна виділити :

- декоратори, регулярні вирази;
- обробка винятків, ітератори та генератори;
- інтроспекція. Це можливість запросити дані про тип, структуру об’єкта під час виконання програми;
- у Python є можливість отримати інформацію про внутрішню структуру будь-якого об’єкта;
- управління контекстом виконання;
- програмне забезпечення можна оформити в якості модулів, які можуть бути зібрані в пакети;

- специфічна реалізація об’єктно-орієнтованого програмування, якщо порівнювати з іншими об’єктно-орієнтованими мовами, але досить добре реалізована та детально продумана.

До основних архітектурних рис Python можна віднести:

- автоматичне керування пам’яттю;
- динамічну типізацію;
- механізм обробки виключень;
- підтримка багатопотокових обчислень;
- високорівневі структури даних.

Можна виділити такі переваги Python:

- характерність мови достатньо логічним синтаксисом, внаслідок чого програма написана на Python легко сприймається при читанні та аналізі коду;
- «умовна легкість», тобто Python вважається найбільш вдалим варіантом для вивчення починаючими програмістами;
- також можна вважати за перевагу - широку інтернет спільноту. Тобто, якщо розробник стикається з труднощами, які не може вирішити самостійно, то можна запитати рішення у інших користувачів Python десь на форумі. Це значно прискорює процес вивчення мови та розробки програми;
- Python є мовою програмування, що інтерпретується. Це означає, що до того, як файл з програмою запуститься на виконання, він представляє із себе звичайний текстовий файл. Відповідно програмувати можна на будь-яких платформах.
- легке масштабування навіть складних програм.
- доступність великої кількості ресурсів.
- навіть зі стандартною бібліотекою можна виконувати складні функції.

Із недоліків слід виділити такі аспекти Python:

- досить повільний;
- неможливість реалізації 3D-графіки з високої роздільною здатністю;
- не підходить для роботи з багатоядерними чи багатопроцесорними потоками;

- захист який забезпечується моделлю пам’яті Python наближує майже до нуля можливість процесорної оптимізації;

Проаналізувавши всі отриманні дані про мову програмування Python та порівнявши їх з заданими критеріями отримаємо у підсумку, що вибрана мова програмування повністю підходить для розробки запланованого програмного продукту. Із недоліків можна виділити лише повільність, але це не є критичним. Так як нам не потрібна високошвидкісна програма, оскільки основа задача програмного продукту це надати належний рівень безпеки даних користувача та доступ до них. Для вирішення зазначеної задачі мова програмування Python підходить повністю. Після остаточного визначення мови програмування слід перейти до вибору середовища розробки.

2.2.3. Обґрунтування вибору середовища розробки

Перед пошуком середовища розробки треба задати критерії по яким буде йти вибірка, а саме:

- безкоштовність;
- зручна реалізація внутрішнього інтерфейсу при роботі з кодом програми;
- орієнтованість на Python;
- можливість зручного рефакторингу [9]: перейменування чи вилучення методу, введення константи чи змінної, зміна розташування методу (підйом/спуск);

Це всі основні критерії по яким буде здійснюватися пошук та вибір середовища розробки на мові програмування Python.

В якості середовища розробки був обраний PyCharm.

PyCharm – це інтегроване середовище розробки для мови Python.

Редактор був розроблений чеською компанією JetBrains спеціально для мови програмування Python. При роботі з цим редактором користувач отримує широкий набір можливостей:

- автозавершення та інспекція коду;
- підсвічування, автоматичне виправлення, налагодження помилок;

- система контролю версії;
- рефакторинг;
- доступність на Windows, Linux, macOS;
- має платну версію, але безкоштовної достатньо для вирішення більшості задач;
- наявність файлової структури проекту;
- швидкий перехід між класами, файлами, методами.

PyCharm має один невеликий недолік, якщо у вас малопотужний комп'ютер який має менше ніж 8гб оперативної пам'яті то середовище розробки буде працювати повільно. Але це незначний недолік.

IDE PyCharm – повність задовольняє висунутим критеріям.

В процесі роботи з цим середовищем, проблем не було виявлено, функціоналу безкоштовної версії вистачило для вирішення поставлених задач.

Повна орієнтованість під Python, зручна навігація в проекті статичний аналіз коду, підсвічування помилок дозволяє заощадити багато часу та сил при розробці програмного продукту.

Комбінація коректно підібраної мови програмування та IDE в рази спрощує роботу під час реалізації програмного продукту чи вирішення необхідних задач.

Після вирішення трьох з поставлених задач перейдемо до останньої поставленої задачі, а саме вибір алгоритму шифрування.

2.2.4 Обґрунтування вибору алгоритму шифрування

Встановимо критерії для вибору алгоритму шифрування:

- стійкість до зламу;
- відносно невеликий час роботи.

Для реалізації програмного продукту був обраний метод шифрування AES, а саме його модифікація AES.MODE EAX.

Advanced Encryption Standard або Rijndael (далі – AES) – алгоритм симетричного блокового шифрування з різним розміром блоку (є три варіації 128/192/256 біт). У 2002 році Агентство Національної Безпеки США постановило,

що алгоритм шифрування AES є досить надійним щоб захищати інформацію, яка відноситься до державної таємниці [10]. Узагальнена схема роботи алгоритму шифрування вказана на рисунку 2.2.1.

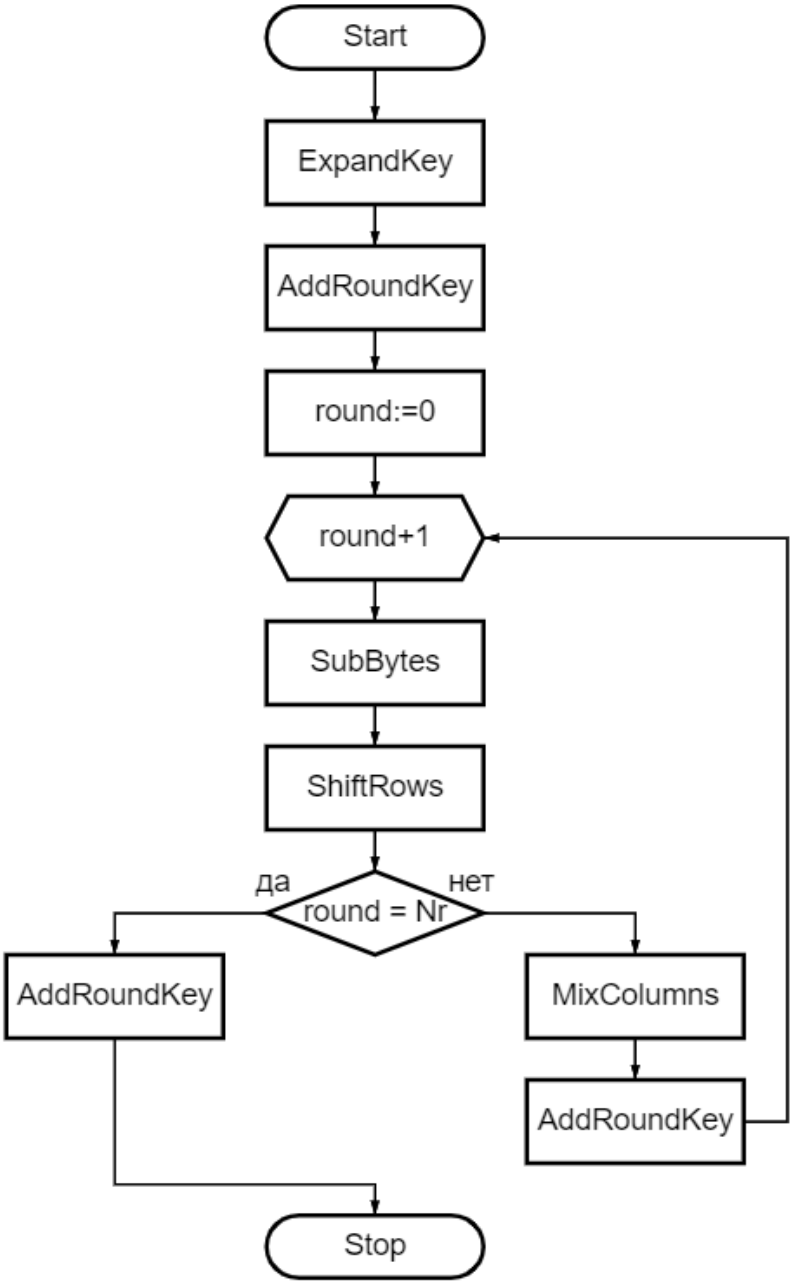

Рисунок 2.2.1 – Загальна схема алгоритму шифрування AES

Загальна схема алгоритму дешифрування вказана на рисунку 2.2.2

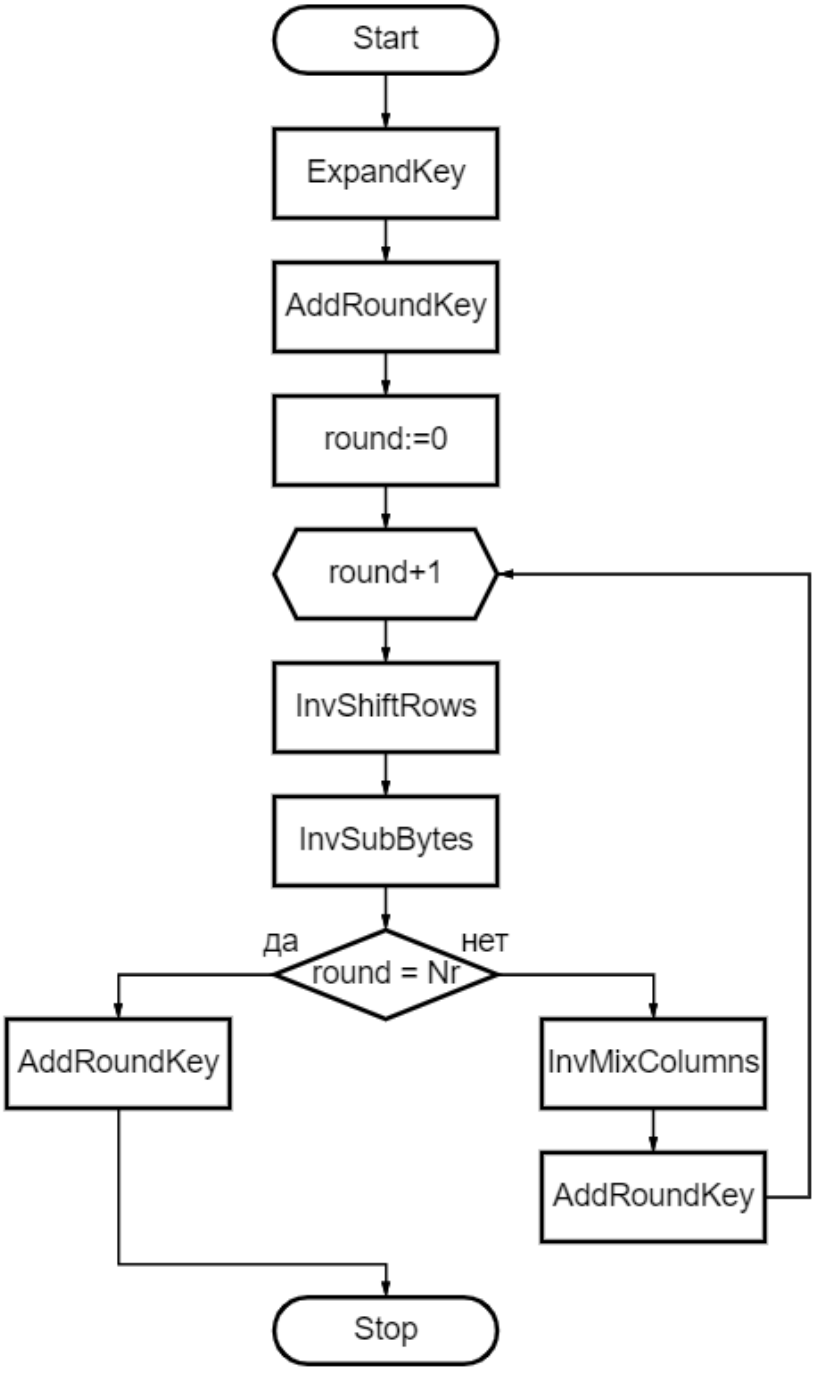

Рисунок 2.2.2 – Схема алгоритму дешифрування AES

Більш детальне пояснення роботи алгоритму AES розглядається у [11].

Перейдемо до модифікації яка використовується в програмному продукті: «AES.MODE EAX».

EAX – це гнучка двопрохідна схема AEAD «автентифіковане шифрування з приєднаними даними» яка функціонує в режимі CTR. Більше інформації про режими можна отримати в [12].

Саме тому була обрана ця модифікація алгоритму для реалізації в програмному продукті.

Важливо зазначити, що шифрування і дешифрування файлу буде реалізовано на стороні клієнта, а не сервера.

У підсумку можна сказати, що всі поставлені задачі на початку розділу були вирішені. Після цього можна приступати до написання та повної реалізації програмного продукту.

3 ОПИС ТА РЕАЛІЗАЦІЯ ФУНКЦІОНУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Опис виконаної роботи

Для початку розберемо яким чином файл був створений і розміщений у Telegram.

Процедура розміщення файлу відбувається наступним чином:

- для початку створюється Python словник (dict). Як його створити можна подивитися тут [16];
- далі з отриманого словника беремо дамп. Детальніше можна прочитати тут [17];
- отриманий дамп шифруємо за допомогою AES.MODE EAX (у якості ключа виступає хеш-функція MD5);
- отриманий результат відправляємо у Telegram;
- вивантажуємо зашифрований дамп і дешифруємо. Бачимо, що все функціонує корректно;
- знов зашифровуємо наш дамп і завантажуємо у Telegram;
- далі на наше повідомлення встановлюємо «прапор»[18] (#passwords) (див. рис. 3.1.1).

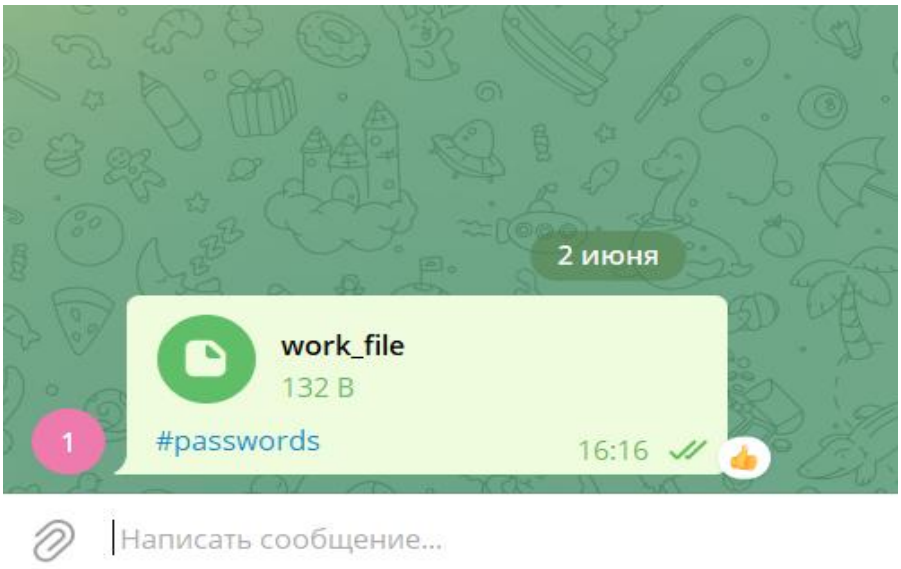

Рисунок 3.1.1 – Встановлений "прапор" на повідомлення

Словник у Python – це невпорядкована колекція деяких довільних об’єктів, доступ до яких здійснюється через ключ.

Дамп – це файл з повним або частинним вмістом пам’яті комп’ютера чи бази даних в якій був розміщений файл у момент створення цього файлу.

Таким чином файл був підготовлений до роботи з програмою. Після цього можна приступати до написання інтерфейсу та функцій, які дозволять програмі організувати взаємодію користувача з файлом на сервері.

Почнемо з написання функцій для початку сформулюємо чіткий алгоритм який дозволить програмі взаємодіяти з сервером.

Порядок функцій які повинна виконувати програма:

- вивантаження файлу з серверу;

- видалення старого файлу;

- перегляд або редагування змісту файлу;

- завантаження файлу назад.

Вивантаження файлу з серверу виконує наступна функція:

```
    def download_file(file_id, api_id, api_hash, keydir):
  with Client("my_account", api_id, api_hash,
workdir=keydir) as app:
    app.download_media(file_id,
file_name='./keys/work_file')
```

Видалення старого файлу виконує настпуна функція:

```
    def delete_message(message_id, api_id, api_hash,
keydir):
  with Client("my_account", api_id, api_hash,
workdir=keydir) as app:
    app.delete_messages('me', message_id)
```

За перегляд або редагування файлу відповідає наступна функція:

```
    def search_message(api_id, api_hash, keydir):
  with Client("my_account", api_id, api_hash,
workdir=keydir) as app:
    for message in app.search_messages("me",
query="#passwords"):
      return message.id, message.document.file_id
```

Завантаження файлу назад виконує наступна функція:

```
    def send_file(file, api_id, api_hash, keydir):
  with Client("my_account", api_id, api_hash,
workdir=keydir) as app:
    app.send_document("me", file, caption='#passwords')
```

Всі вищезазначені функції були об’єднані до одного модуля в програмі «telegram.py» (див. рис. 3.1.2).

```
from pyrogram import Client


def send_file(file, api_id, api_hash, keydir):
    with Client("my_account", api_id, api_hash, workdir=keydir) as app:
        app.send_document("me", file, caption='#passwords')


def search_message(api_id, api_hash, keydir):
    with Client("my_account", api_id, api_hash, workdir=keydir) as app:
        for message in app.search_messages("me", query="#passwords"):
            return message.id, message.document.file_id


def download_file(file_id, api_id, api_hash, keydir):
    with Client("my_account", api_id, api_hash, workdir=keydir) as app:
        app.download_media(file_id, file_name='./keys/work_file')


def delete_message(message_id, api_id, api_hash, keydir):
    with Client("my_account", api_id, api_hash, workdir=keydir) as app:
        app.delete_messages('me', message_id)
```

Рисунок 3.1.2 – Модуль програми "telegram.py"

Після написання вищезазначеного модулю, необхідно написати модуль, що буде забезпечувати автентифікацію користувача при вході в програму і шифрування та дешифрування файлу. Отже алгоритм програми набуває наступного вигляду:

Автентифікація користувача. При невдалій спробі програма нічого не робить/ при вдалій програма продовжує працювати.

- вивантаження файлу з серверу;
- видалення старого файлу;
- дешифрування файлу;
- редагування або перегляд файлу;
- шифрування файлу;
- завантаження нового файлу на сервер.

Враховуючи оновлений алгоритм роботи програми, напишемо функції які будуть це виконувати.

Почнемо з автентифікації користувача. Для початку опишемо, як вона працює.

Автентифікація користувача реалізується в програмному продукті за допомогою майстер-паролю (текстового паролю).

При реалізації програми був вибраний майстер-пароль. Для перевірки користувача в код програми був завантажений хеш MD5 цього паролю (див. рис. 3.1.3).

```
login_hash = '5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8'
```

Рисунок 3.1.3 – Встановлений MD5 хеш

Коли користувач здійснює вхід до програми він повинен ввести майстер-пароль (див. рис. 3.1.4).

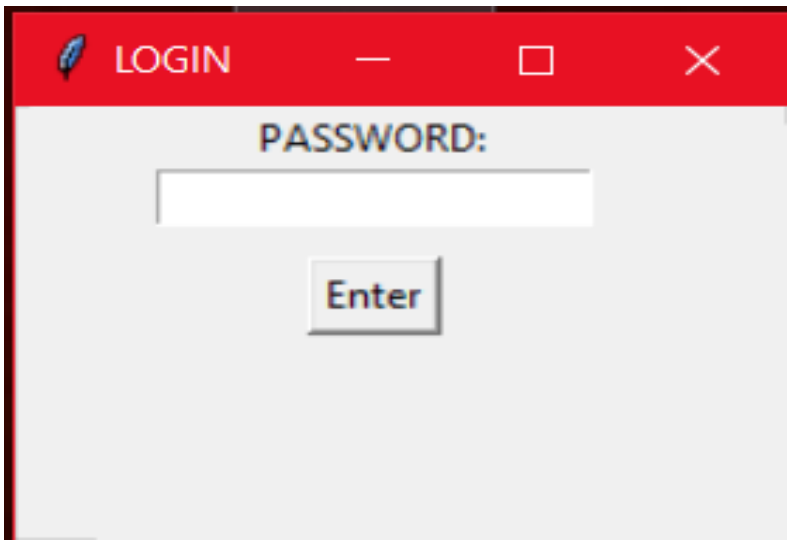

Рисунок 3.1.4 – Вікно для вводу майстер-паролю

Порівняння хеш майстер-паролю, яку ввів користувач зі встановленим відбувається у функції:

```
    def vereficate_password(password):
  return True if get_sha256_hash(password) == login_hash
else False
```

Якщо користувач вводить невірний пароль, то отримає повідомлення про це (див. рис. 3.1.5).

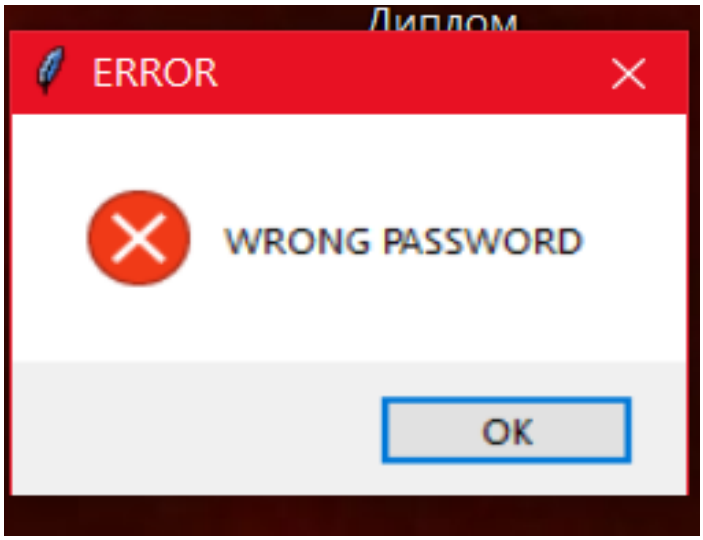

Рисунок 3.1.5 – Повідомлення, що пароль невірний

Після успішної авторизації користувача, настає черга вивантаження файлу з серверу та дешифрування його.

За дешифрування файлу відповідає функція:

```
    def decrypt_AES(files, password):
  for file in files:
    key = password.encode('UTF-8')
    file_in = open(file, "rb")
    nonce, tag, ciphertext = [file_in.read(x) for x in (16,
16, -1)]
    cipher = AES.new(key, AES.MODE_EAX, nonce)
    file_data = cipher.decrypt_and_verify(ciphertext, tag)
    file_out = open(file, 'wb')
    file_out.write(file_data)
    file_out.close()
```

Як тільки дешифрування відбудеться, користувач отримає діалогове вікно через яке зможе переглянути необхідні дані чи щось змінити (див. рис. 3.1.6).

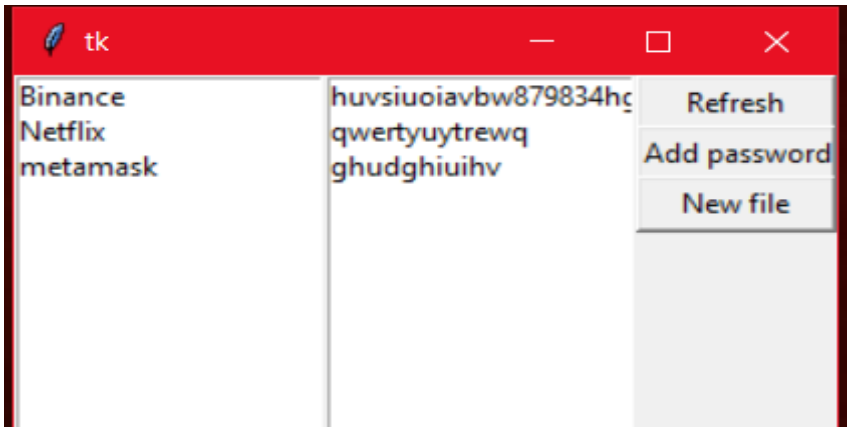

Рисунок 3.1.6 – Вікно для взаємодії користувача з файлом

В кінці сеансу взаємодії користувача з файлом. Програма шифрує отриманий новий файл.

Функція яка виконує шифрування:

```
    def crypt_AES(files, password):
  for file in files:
    file_in = open(file, 'rb')
    file_data = file_in.read()
    file_in.close()

    key = password.encode('UTF-8')
    cipher = AES.new(key, AES.MODE_EAX)
    ciphertext, tag = cipher.encrypt_and_digest(file_data)

    file_out = open(file, "wb")
    [file_out.write(x) for x in (cipher.nonce, tag,
ciphertext)]
    file_out.close()
```

В завершені шифруванні програма відправляє отриманий новий файл на сервер, де файл зберігається до наступного запуску програми.

3.2 Опис процесу шифрування файлу.

При розробці програмного продукту була використана модифікація алгоритму шифрування AES.MODE EAX, у якості ключа шифрування виступала хеш-функція.

Хеш-функція – функція яка здійснює перетворення даних вхідного масиву довільного розміру в бітову строку певної довжини[13].

В програмному продукті були використані дві хеш-функції: SHA256; MD5.

SHA256 – представляє собою односпрямовану функцію для створення цифрових відбитків фіксованої довжини (256 біт). Максимальне значення вхідних даних, які ця хеш-функція може перетворити дорівнює 2^{64}біт[14].

MD5 – це хеш-функція яка дозволяє хешувати якесь повідомлення L з деяким ключом K. Використовується для створення автентифікації цифрового підпису. Максимальне значення отриманого хеша дорівнює 128 біт [15].

Шифрування файлу відбувається наступним чином: користувач вводить майстер-пароль; з правильного майстер-паролю береться хеш-функція SHA256; отриманий результат перетворюється на хеш-функцію MD5; останній отриманий хеш і є ключом для шифрування та дешифрування файлу.

Такий метод був реалізований для того, щоб якщо зловмисник вкраде хеш майстер-паролю, він не зміг би дешифрувати файл за допомогою власної програми.

3.3 Опис повністю реалізованої програми

Розберемо всі модулі які були створенні для роботи програми (рис. 3.1.7).

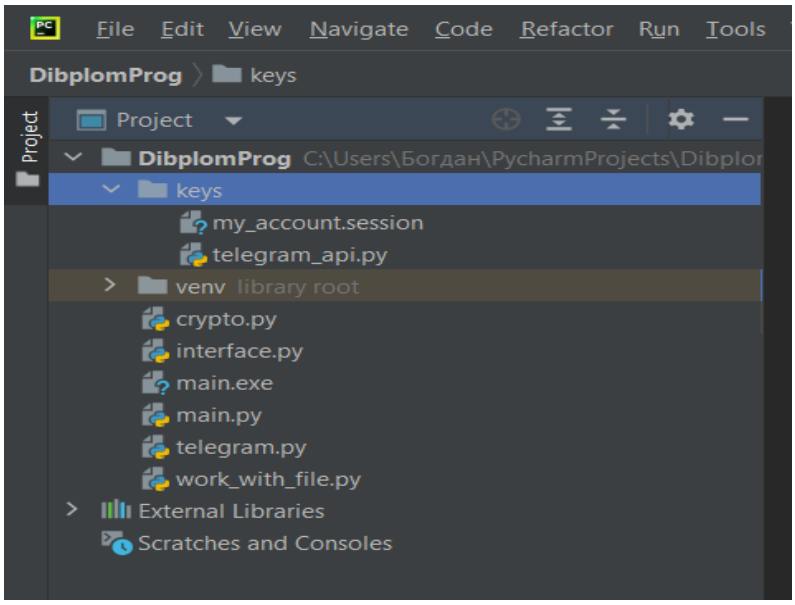

Рисунок 3.1.7 – Файлова структура проекту

«My_account.session» – це службовий файл. В цьому файлі записані ключі від сесії телеграму, щоб програма мала можливість підключитися до сесії, яку надає телеграм.

«Telegram_api.py» – файл, який зберігає в собі ключі від облікового запису телеграм в зашифрованому вигляді, щоб програма могла ідентифікувати обліковий запис користувача.

Ці два фали дуже важливі, оскільки без них програма не зможе взаємодіяти з сервером.

Надалі перейдемо до основних модулів, які мають в собі прописані функції.

Першим на черзі розглянемо модуль «crypto.py» (див. Додаток А).

Функціонал цього модулю відповідає за:

- створення MD5 функції;
- створення SHA256 функції;
- перевірки хеш майстер-паролю з вписаним заздалегідь у цей модуль хешем;
- шифрування файлу;
- дешифрування файлу;
- також в цьому модулі встановленні бібліотеки для роботи з AES та хеш-функціями.

Наступним розглянемо модуль «interface.py» (див. додаток А). Як зрозуміло з назви, цей модуль відповідає за інтерфейс програми.

В ньому прописані не лише код під інтерфейс, а також асоціації певних функцій інших модулів з елементами інтерфейсу.

Третім буде розглянутий вище модуль «telegram.py». В ньому прописані функції, які вказують програмі, що робити з файлом.

Четвертим буде розглянутий модуль «work_with_file.py». Основна функція цього розділу – це забезпечити роботу користувача з файлом. Тобто цей розділ відповідає за: внесення та збереження змін у файл; пошуку файлу за «прапором»; та повернення файлу в те місце де його було взято.

Останнім з модулів буде «main.py». Це модуль, основна функція якого забезпечити взаємодію всіх інших модулів між собою. За допомогою цього модулю відбувається компонування модульної програми в пакети, які повністю функціонують як повноцінна програма.

Для простоти запуску програми був створений «main.exe» із однойменного модулю. Тепер користувач не має потреби запускати середовище розробки для запуску програми, потрібно лише зробити подвійний клік по файлу з назвою «main.exe» на робочому столі.

Щоб перетворити модуль на .exe файл треба зайти в командний рядок Python і прописати там: «pip install PyInstaller» (див. рис. 3.1.8).

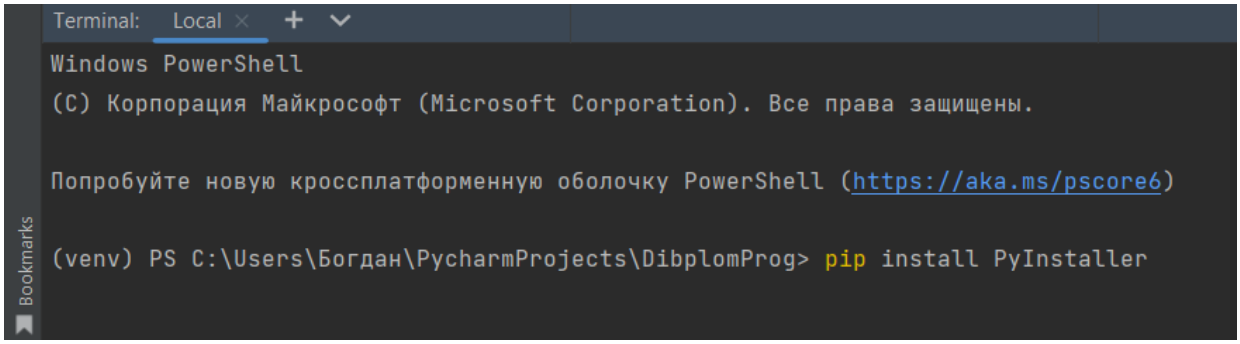

Рисунок 3.1.8 – Введення команди

Після інсталяції треба скомпонувати .exe файл. Для цього треба в командний рядок Python прописати наступну команду: «pyinstaller – -onefile main.py» (див. рис. 3.1.9).

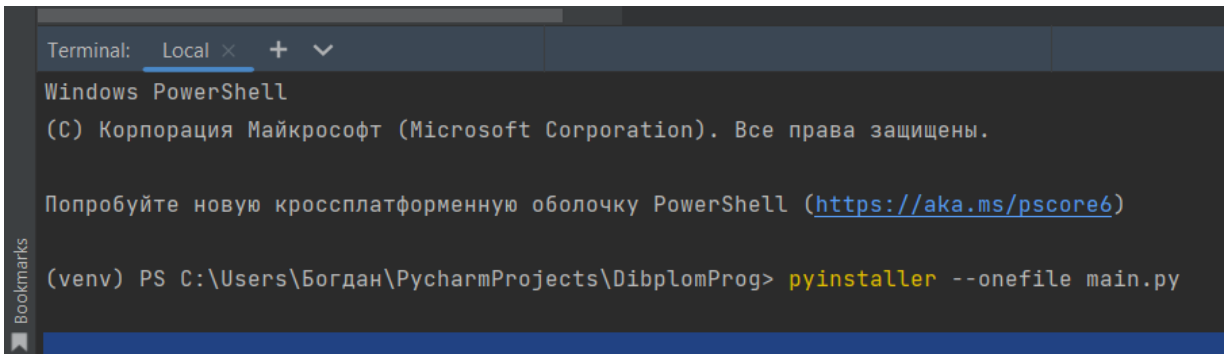

Рисунок 3.1.9 –Введення команди

У результаті отримаємо exe-файл, який можна запустити з командного рядку чи створити ярлик на робочому столі для запуску. В підсумку можна вважати, що задача з реалізації програми-клієнта для збереження паролів за допомогою сторонніх сервісів була повністю виконана.

4 ОХОРОНА ПРАЦІ

4.1 Аналіз умов праці та вибір основних заходів виробничої безпеки

В цьому розділі кваліфікаційної роботи були розглянуті поставлені задачі, вирішення яких необхідно для встановлення норм праці на робочому місці інженера-програміста.

В першу чергу було виявлено які саме небезпечні фактори можуть бути на робочому місці інженера-програміста. Можливі небезпечні фактори зазначені у ДСТ 12.0.003-74* [18], тому треба провести аналіз документа і виділити їх:

- недостатня освітленість робочого місця;
- підвищений рівень шуму на робочому місці;
- підвищена чи знижена температура повітря на робочому місці;
- підвищена чи знижена вологість повітря на робочому місці;
- підвищена чи знижена швидкість руху повітря на робочому місці;
- підвищене значення напруги в електричному ланцюзі.

Отримавши основні фактори небезпеки при робочій діяльності, їх треба дослідити та запропонувати можливі рішення.

Робота інженера-програміста полягає в розробці програмних продуктів з використанням персонального комп’ютеру. Тому найбільше навантаження, під час виконання робочих завдань, випадає на очі інженера-програміста.

З метою уникнення можливих негативних наслідків розглянемо нормативний документ ДБН В.2.5-28-2006 [19]. Відповідно до документа отримаємо, що рівень освітлення робочого місця під час роботи за монітором комп’ютера повинен бути не менший за 500лк.

Освітлення в робочих приміщеннях та на робочих місцях поділяється на наступні три види:

- природне;
- штучне;
- комбіноване.

Інженер-програміст може працювати, як і в денний період часу, так і в вечірній. У більшості офісних приміщень необхідна норма освітлення в денний

період часу забезпечується за рахунок природного світла, водночас для роботи в вечірній період часу необхідно обов’язково передбачити у достатній кількості джерела штучного освітлення.

До інших проблем з освітленням можна віднести:

- віддзеркалення світла від поверхонь складових частин персонального комп’ютеру та/або предметів інтер’єру, що оточують робоче місце інженера-програміста;
- великий контраст між темними і світлими зображеннями монітора або між монітором та навколишнім середовищем.

Проаналізувавши можливі проблеми з освітленням пропонується вжити наступні заходи:

- обов’язкова наявність комбінованого освітлення на робочому місці;
- облаштування робочого місця таким чином, щоб відбиття від поверхонь в приміщені не створювало додаткового навантаження для очей інженера-програміста;
- кількість світла повинна бути достатньою, щоб бачити зображення на моніторі, але не настільки яскравим, щоб віддзеркалювати від екрану в очі або засвітлювати його;
- правильно підібране та налаштоване штучне освітлення.

Наступним небезпечним фактором буде розглянутий шум. Шум на робочому місці інженера-програміста, виникає у наслідок роботи персонального комп’ютера або інших електроприладів розташованих на робочому місці. Також можливими шумоутворючими факторами може бути близьке розміщення робочого приміщення інженера-програміста до промислових об’єктів з великим рівнем шуму та вібрацій (будівництва, виробництво, цехи, тощо), або низький рівень шумоізоляції робочого приміщення від вуличних шумів (людяні місця, автотранспорт, тощо).

Для встановлення допустимого рівня шуму на робочому місці інженера-програміста проаналізуємо ДСН 43.3.6 037-99 [20] та з’ясуємо, що допустимий рівень шуму не має перевищувати 50 дБА.

Для забезпечення дозволеного рівня шуму в робочому приміщенні, у разі перевищення показників зашумлення, необхідно передбачити в приміщені додаткові звукоізоляційні засоби (шумоізолюючі склопакети в вікнах, шумопоглинаючі панелі на стінах).

Розташування та наявність електроприладів на робочому місці, часто є основним фактором при утворенні зайвого шуму. Електроприлади на робочому місці інженера-програміста, що створюють додатковий шум – є невід'ємною частиною його роботи (комп'ютер, принтер, джерело безперебійного живлення). Враховуючи вищевикладене, для забезпечення норм шуму необхідно передбачити наступні заходи:

- прилади, що неминуче створюють шум, за можливості розташувати подалі від робочого місця та використовувати їх лише за необхідності;
- використовувати технічні засоби, що мінімізують шум під час роботи комп'ютера (водяне або пасивне охолодження комп'ютера, кліматична техніка з інверторними електричними двигунами).

Поряд з вирішенням проблем з освітленням та шумом постає питання мікроклімату в приміщенні. Мікроклімат в робочому приміщенні визначається діючими на організм людини факторами температури, вологості та швидкості повітря. Не дотримання норм мікроклімату, може призвести до виникнення хронічних хвороб та в наслідок к незадовільним результатам роботи інженера-програміста.

Для отримання стандартних норм звернемося до документу ДСанПіН 3.3.2.007-98 [21]

За результатами аналізу ДСанПіН 3.3.2.007-98, маємо такі стандарти мікроклімату для робочого місця інженера-програміста:

- площа на одне робоче місце має становити не менше ніж 6,0 $м^2$, а об'єм - не менше ніж 20,0 $м^3$;
- температура повітря в приміщені має бути в діапазоні від 22 до 24 °С;
- вологість повітря повинна бути в межах 40-60%;
- встановлена у нормативному документі швидкість руху повітря - 0.1 м/с;

Для забезпечення необхідних норм мікроклімату необхідно:

- використовувати прилад для вимірювання вологості та температури повітря в приміщенні (такі прилади легко придбати в магазині);

- забезпечити регулювання температури, за допомогою кліматичної техніки, з обов’язковим дотриманням техніки безпеки при їх використані;

- для регулювання вологості повітря можна придбати спеціальний пристрій, або регулярно робити вологе прибирання в приміщенні;

- важливо забезпечити циркуляцію свіжого повітря в приміщенні. Для цього необхідно перебачити засоби вентиляції в приміщенні (пасивні чи примусові) або регулярно його провітрювати.

Правильно налаштований мікроклімат зробить робочий процес комфортним та безпечним для людини.

Робочий процес, пов’язаний із взаємодією інженера-програміста з комп’ютером, забезпечення штучного освітлення, підтримка стандартів мікроклімату – все це забезпечується за допомогою електричних приладів.

Використання електрики утворює величезну кількість небезпек для працівника, але без цього не можлива діяльність інженера-програміста. Тому є важливим передбачити можливі загрози та протидіяти їм.

Можливі загрози:

- аварії, що можуть призвести до пожежі чи ураження людини електричним струмом;

- електромагнітний вплив на людину.

Не зважаючи, на невеликий перелік загроз, результатами їх виникнення можуть бути серйозні проблеми зі здоров’ям або смерть.

Дотримання техніки безпеки з експлуатації електроприладів є першим, що повинен вивчити інженер-програміст перед допуском до роботи.

Для захисту від електричних загроз, під час виконання робіт, рекомендується:

- впевнитись, що навантаження на електромережу приміщення не перевищує її можливостей;

- вимикати прилади з розетки, у разі їх непотрібності або робочий процес завершено;

- перевіряти електричні вузли та шнури електроприладів на наявність пошкоджень;

- всі електричні шнури повинні бути обережно прокладенні (без перекручень та сильних вигинів), за робочим столом, вздовж стін або плінтусів. Не допускається розташування шнурів та подовжувачів під ногами людини;

- обов'язкова наявність стабілізаторів живлення або протизавадних фільтрів. Ці пристрої дозволять контролювати напругу в електричних ланцюгах при її перепадах;

- встановлення автоматичних вимикачів в електролічильнику. Це дозволить знеструмити все робоче приміщення за необхідності;

- при виявлені проблем, необхідно одразу визвати майстра та припинити роботи до усунення проблеми.

Вплив електромагнітного випромінювання на організм людини також є досить небезпечним. Він може стати каталізатором для багатьох проблем зі здоров'ям. Для зменшення негативного впливу такого випромінювання необхідно:

- дотримуватися відстані не менше ніж 600мм від ліній елетромережі та електроприладів.

- за можливості використовувати рідкокристалічні екрани монітору (вони мають нижчі рівні випромінювання навідмінно від інших).

Зазначенні правила дозволять зменшити негативний вплив випромінювань до мінімуму.

Окрім вищезазначених правил безпеки, необхідно також забезпечити правильне та комфортне положення тіла за робочим місцем. Для цього потрібно мати в наявності корректно підібрані стілець (крісло) та стіл. Правильне та комфортне положення тіла за робочим місцем зменшить навантаження на хребет та шию, та підвищить рівень продуктивності людини.

4.2 Аналіз пожежної небезпеки та вибір заходів і засобів пожежної безпеки

Пожежа - найнебезпечніша подія, що може статися в робочому приміщені. Наслідки від пожежі завжди трагічні, починаючи з економічних втрат та закінчуючи смертю людей. Тому пожежна безпека - це перше, що перевіряється перед початком експлуатації приміщення, незалежно від його призначення.

В цьому розділі буде розглянута пожежна безпека в робочому приміщені інженера-програміста. Тому для початку було встановлено клас пожежі, що може відбутися. Для цього було проведено аналіз ДСТУ EN 2:2014 [22] та визначено, що в приміщенні буде відбуватися горіння твердих речовин, а це клас пожежі «В».

Щоб зменшити ризик виникнення пожежі, треба для початку виявити можливі причини.

До можливих причин виникнення пожежі на робочому місці інженера-програміста відноситься:

- недотримання заходів з безпеки при роботі з електричними приладами;
- аварії, пов’язані з електрикою (коротке замикання, деформація обмотки електричного кабелю, вихід зі строю електроприладу);
- паління на робочому місці.

Проаналізувавши можливі причини виникнення пожежі, для захисту від них необхідно:

- забезпечити наявність вуглекислотного вогнегасника в кожному робочому приміщенні;
- забезпечити приміщення засобами пожежної сигналізації;
- забезпечити обов’язкове вивчення техніки безпеки та поведінки під час пожежі;
- при перших проявах горіння, одразу знеструмити приміщення;
- у разі не можливості самостійно загасити пожежу, викликати пожежних.

Вогнегасник повинен бути вуглекислотним, оскільки в при пожежі класу «В» коли відбувається горіння і тління твердих речовин, необхідно максимально швидко знизити температуру поверхні об’єкту займання.

ВИСНОВОК

В даній кваліфікаційній роботі був проведений аналіз вже існуючих рішень забезпечення зберігання паролів, який показав, що використання вже існуючих рішень не є цілком раціональним для одного користувача. Більшість інноваційних і дійсно корисних послуг, які можуть надати існуючи програми, мають чималу вартість. А безкоштовні послуги є однотипними в усіх сервісів та не покривають усіх потреб користувача.

Розроблено і реалізовано персональний програмний додаток, який буде виконувати необхідні функції для користувача. Переваги розробленого додатку в порівнянні з аналогами наступні:

- можливість самостійного вибору функціоналу;
- самостійний вибір сервісу, який буде виступати в ролі серверу для збереження даних;
- можливість модифікації програмного продукту за бажанням користувача;
- користування не потребує придбання ліцензії;
- самостійний вибір алгоритмів шифрування та методів його реалізації.

Даний програмний продукт повністю задовольняє потреби користувача, є інтуїтивно зрозумілим та простим у використанні. Всі задачі кваліфікаційної роботи Вирішено, мету досягнуто.

ПЕРЕЛІК ПОСИЛАНЬ

1. Зоріло В.В., Осколкова О.Є., Гаврилюк Б.В. Розробка програмного забезпечення зберігання та захисту даних для приватного та корпоративного застосування. Інформатика та математичні методи в моделюванні. Том 12, № 1-2. 2022. С.50-60.;

2. Опис Telegram URL: https://startpack.ru/application/telegram-messenger#:~:text=Telegram%20-%20это%20веб-приложение%20для,настольных%20ПК%2C%20планшетах%20и%20телефонах ;

3. Мобільний протокол MTProto URL: https://tlgrm.ru/docs/mtproto ;

4. Алгоритм Діффі – Хеллмана URL: http://kaf403.rloc.ru/POVS/Crypto/DiffieHellman.html ;

5. Алгоритм шифрування RSA URL: https://e-nigma.ru/stat/rsa ;

6. Наскрізне шифрування URL: https://academy.binance.com/uk/articles/what-is-end-to-end-encryption-e2ee ;

7. Твіт Метт Грін URL: https://twitter.com/matthew_d_green/status/582916365750669312;

8. Лутц, М. Программирование на Python, I том / М. Лутц. - СПб.: Символплюс, 2015. - 992 с;

9. Рефакторинг URL: https://dou.ua/lenta/articles/refactoring;

10. AES URL: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard;

11. Алгоритм блочного симметричного шифрования Advanced Encryption Standard (AES) URL: http://crypto.pp.ua/wp-content/uploads/2010/03/aes.pdf;

12. Режимы шифрования блоковых шифров URL: https://bit.nmu.org.ua/ua/student/metod/cryptology/лекция%207.pdf;

13. Алгоритм SHA256 URL:

https://medium.com/dtechlog/алгоритмы-хэш-функция-sha-256-9862302f942f#:~:text=SHA-256%20представляет%20собой%20однонаправленную,Version%202)%20опубликованным%20АНБ%20США ;

14. Хэш-функция MD5 URL: https://habr.com/ru/sandbox/26876;

15. Словари (dict) и работа с ними. Методы словарей URL: https://pythonworld.ru/tipy-dannyx-v-python/slovari-dict-funkcii-i-metody-slovarej.html ;

16. Модуль json URL: https://pythonworld.ru/moduli/modul-json.html;

17. Переменные-флаги URL: https://dvmn.org/encyclopedia/python_intermediate/flag-variables ;

18. ДСТ 12.0.003-74*. ССБТ. Небезпечні і шкідливі виробничі фактори. Класифікація. URL: https://budinfo.org.ua/doc/1810987/DST-12-0-003-74-SSBT-Nebezpechni-i-shkidlivi-virobnichi-faktori-Klasifikatsiia ;

19. ДБН В.2.5-28-2006. Природне і штучне освітлення. URL: http://kbu.org.ua/assets/app/documents/dbn2/95.1.%20ДБН%20В.2.5-28-2006.%20Природне%20і%20штучне%20освітлення.pdf ;

20. ДСН 3.3.6.037-99. Санітарні норми виробничого шуму, ультразвуку та інфразвуку. URL: https://zakon.rada.gov.ua/rada/show/va037282-99#Text ;

21. ДСанПІН 3.3.2.007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text ;

22. ДСТУ EN 2:2014. Класифікація пожеж. URL: http://online.budstandart.com/ua/catalog/topiccatalogua/fire-afety/06._dstu_(derzhavnyi_143116/2-2014+63091.

Додаток А. Лістинг програми

Main.py

```
import crypto
import interface
import sys
import os
import telegram

keys_directory = './keys/'

def full_patch(directoy, files):
    filee = []
    full_patch = map(lambda name: os.path.join(directoy,
name), files)
    for file in full_patch:
        if os.path.isfile(file):
            filee.append(file)
    return filee


def crypt_all():
    password = 'password'
    crypto.crypt_AES(full_patch(keys_directory,
os.listdir(keys_directory)), crypto.get_md5_hash(password))


def main():
    password = interface.login_window()
    crypto.decrypt_AES(full_patch(keys_directory,
os.listdir(keys_directory)), crypto.get_md5_hash(password))

    sys.path.insert(0, keys_directory)
    import telegram_api

    message_id, file_id =
telegram.search_message(telegram_api.api_id,
telegram_api.api_hash, keys_directory)

    telegram.download_file(file_id, telegram_api.api_id,
telegram_api.api_hash, keys_directory)
    crypto.decrypt_AES([f'{keys_directory}work_file'],
crypto.get_md5_hash(password))

    interface.main_window(telegram_api.api_id,
```

```
telegram_api.api_hash, keys_directory)

    telegram.delete_message(message_id,
telegram_api.api_id, telegram_api.api_hash, keys_directory)
    crypto.crypt_AES([f'{keys_directory}work_file'],
crypto.get_md5_hash(password))
    telegram.send_file('./keys/work_file',
telegram_api.api_id, telegram_api.api_hash, keys_directory)
    os.remove('./keys/work_file')

    crypto.crypt_AES(full_patch(keys_directory,
os.listdir(keys_directory)), crypto.get_md5_hash(password))


if __name__ == '__main__':
    main()
    #crypt_all()

    Crypto.py

from Cryptodome.Cipher import AES

import hashlib


login_hash =
'5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d
1542d8'


def get_md5_hash(data):
    return hashlib.md5(data.encode('utf-8')).hexdigest()


def get_sha256_hash(password):
    return hashlib.sha256(password.encode('UTF-
8')).hexdigest()


def vereficate_password(password):
    return True if get_sha256_hash(password) == login_hash
else False

def crypt_AES(files, password):
    for file in files:
```

```
        file_in = open(file, 'rb')
        file_data = file_in.read()
        file_in.close()

        key = password.encode('UTF-8')
        cipher = AES.new(key, AES.MODE_EAX)
        ciphertext, tag =
cipher.encrypt_and_digest(file_data)

        file_out = open(file, "wb")
        [file_out.write(x) for x in (cipher.nonce, tag,
ciphertext)]
        file_out.close()


def decrypt_AES(files, password):
    for file in files:
        key = password.encode('UTF-8')

        file_in = open(file, "rb")
        nonce, tag, ciphertext = [file_in.read(x) for x in
(16, 16, -1)]
        cipher = AES.new(key, AES.MODE_EAX, nonce)
        file_data = cipher.decrypt_and_verify(ciphertext,
tag)

        file_out = open(file, 'wb')
        file_out.write(file_data)
        file_out.close()
    Interface.py
    from tkinter import *
from tkinter import filedialog
from tkinter import messagebox

import crypto
import telegram
import work_with_file


def get_file():
    return filedialog.askopenfilename()


def get_directory():
    return filedialog.askdirectory()
```

```
def login_window():
    def accept_login(password):
        login_page.destroy() if
crypto.vereficate_password(password) else
messagebox.showerror(title='ERROR',

message='WRONG PASSWORD')

    login_page = Tk()

    password = StringVar()

    login_page.title('LOGIN')
    login_page.geometry('200x100')

    Label(login_page, text='PASSWORD:').grid(row=0,
column=0, padx=(40, 0))
    Entry(login_page, show='*',
textvariable=password).grid(row=1, column=0, padx=(40, 0))
    Button(login_page, text='Enter', command=lambda:
accept_login(password.get())).grid(row=3, column=0,
pady=(10, 0),

padx=(40, 0))
    login_page.mainloop()
    return password.get()


def main_window(api_id, api_hash, keys_directory):
    def add_new_line():
        def add_window():
            add_window = Tk()

            Label(add_window, text='Area').grid(row=0,
column=0)
            Label(add_window, text='Password').grid(row=0,
column=1)

            area = StringVar(add_window)
            password = StringVar(add_window)

            Entry(add_window,
textvariable=area).grid(row=1, column=0)
```

```
            Entry(add_window,
textvariable=password).grid(row=1, column=1)

            Button(add_window, text='ADD!', command=lambda:
[work_with_file.add_pass(area.get(), password.get()),

messagebox.showinfo(message='Add succesful!'),

add_window.destroy()]).grid(row=2, column=0, columnspan=2)

            add_window.mainloop()

        add_window()

    def refresh_func():
        passwords = work_with_file.get_passwords()
        PlACE_list.delete(0, END)
        PASSWORD_list.delete(0, END)
        for key in passwords:
            PlACE_list.insert(END, key)
            PASSWORD_list.insert(END, passwords[key])

    root = Tk()
    PlACE_list = Listbox(root, selectmode=SINGLE)
    PASSWORD_list = Listbox(root, selectmode=SINGLE)

    PlACE_list.grid(row=0, column=0, rowspan=3)
    PASSWORD_list.grid(row=0, column=1, rowspan=3)

    Button(root, text='Refresh', width=10, command=lambda:
refresh_func()).grid(row=0, column=2, sticky='n')
    Button(root, text='Add password', width=10,
command=lambda: [add_new_line(),
refresh_func()]).grid(row=0, column=2)
    Button(root, text='New file', width=10,
           command=lambda: [work_with_file.create_file(),
messagebox.showinfo(message='File create succesful!'),
                            refresh_func()]).grid(row=0,
column=2, sticky='s')
    refresh_func()

    root.mainloop()
    Telegram.py
    from pyrogram import Client
```

```
def send_file(file, api_id, api_hash, keydir):
    with Client("my_account", api_id, api_hash,
workdir=keydir) as app:
        app.send_document("me", file, caption='#passwords')


def search_message(api_id, api_hash, keydir):
    with Client("my_account", api_id, api_hash,
workdir=keydir) as app:
        for message in app.search_messages("me",
query="#passwords"):
            return message.id, message.document.file_id


def download_file(file_id, api_id, api_hash, keydir):
    with Client("my_account", api_id, api_hash,
workdir=keydir) as app:
        app.download_media(file_id,
file_name='./keys/work_file')


def delete_message(message_id, api_id, api_hash, keydir):
    with Client("my_account", api_id, api_hash,
workdir=keydir) as app:
        app.delete_messages('me', message_id)
    Work_with_file.py
    import pickle


def create_file():
    with open('./keys/work_file', 'wb') as new:
        pickle.dump({}, new)


def add_pass(area, password):
    with open('./keys/work_file', 'rb') as old:
        passwords = pickle.load(old)
    passwords[area] = password
    with open('./keys/work_file', 'wb') as new:
        pickle.dump(passwords, new)

def get_passwords():
    with open('./keys/work_file', 'rb') as old:
        return pickle.load(old)
```