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

Чепурной Кирило Вячеславович,
студент групи РЗ-181

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Алгоритм та реалізація хмарного сейфа з подвійним дном на основі IM-мереж

Спеціальність:
125 Кібербезпека

Спеціалізація, освітня програма:
Кібербезпека

Керівник:
Соколов Артем Вікторович,
к.т.н., доцент

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

Чепурному Кирилу Вячеславовичу

1. Тема роботи: *Алгоритм та реалізація хмарного сейфа з подвійним дном на основі IM-мереж*
2. Керівник роботи: *Соколов А.В. к.т.н., доцент*.

затверджені наказом ректора від „17” 05.2022р. №168-в.

3. Зміст роботи: *Підвищення захищеності користувацьких файлів шляхом розробки та імплементації алгоритму хмарного сейфа з подвійним дном на основі IM-мереж*
4. Перелік графічного матеріалу*: блок-схема алгоритму, рисунки інтерфейсу програмного продукту, рисунки програмного коду*.

4. Консультанти розділів роботи

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Охорона праці	*доц. Ярова І.А.*		

Дата видачі завдання “_____”_____2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	*Аналіз літератури за темою кваліфікаційної роботи*	*02-03-2022*	*виконано*
2	*Аналіз програмного забезпечення, що надає можливість створення криптосейфів*	*22-03-2022*	*виконано*
3	*Аналіз сервісів для хмарного зберігання інформації*	*06-04-2022*	*виконано*
4	*Розробка алгоритму*	*28-04-2022*	*виконано*
5	*Імплементація алгоритму*	*10-05-2022*	*виконано*
6	*Підготовка пояснювальної записки.*	*16-05-2022*	*виконано*
7	*Підготовка презентації та доповіді*	*23-05-2022*	*виконано*
8	*Попередній захист*	*02-06-2022*	*виконано*
9	*Нормоконтроль, рецензування*	*18-02-2022*	*виконано*

Здобувач вищої освіти _____ *Чепурной К.В.*

Керівник роботи _____ *Соколов А.В.*

ЗАВДАННЯ

на розробку розділу «Охорона праці» у кваліфікаційній роботі бакалавра

студенту *Чепурному Кирилу Вячеславовичу*

Інститут інформаційної безпеки радіоелектроніки та телекомунікацій

Кафедра кібербезпеки та програмного забезпечення

Дата отримання завдання 13.04.2022

Консультації 16.05.2022, 23.05.2022

Дата закінчення розділу 16.06.2022

Тема роботи *Алгоритм та реалізація хмарного сейфа з подвійним дном на основі IM-мереж*

Зміст розділу

1. Аналіз умов праці і вибір основних заходів виробничої безпеки
2. Аналіз пожежної безпеки і вибір заходів і засобів пожежної безпеки

Керівник роботи	Консультант з охорони праці
_____ (А.В. Соколов)	_____ (Ярова І.А.)
«____» _____ 2022 р.	«____» _____2022 р

АНОТАЦІЯ

Дипломна робота на тему «Алгоритм та реалізація хмарного сейфа з подвійним дном на основі ІМ-мереж» на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 – Кібербезпека, спеціалізація, освітня програма: Кібербезпека, містить 27 рисунків, 1 додаток, 14 літературних джерел за переліком посилань. Робота виконана на 52 сторінках загального тексту і 43 сторінках основного тексту.

Основною метою даної дипломної роботи є підвищення захищеності користувацьких файлів шляхом розробки та імплементації алгоритму хмарного сейфу з подвійним дном на основі ІМ-мереж.

Розроблено та реалізовано у програмному продукті алгоритм хмарного сейфу з подвійним дном на основі Telegram.

Дана розробка може використовуватись громадянами і військовими як під час військового стану, так і у мирний час, для зберігання своєї приватної інформації.

СЕЙФ, КРИПТОГРАФІЯ, ПОДВІЙНЕ ДНО, СИМЕТРИЧНЕ ШИФРУВАННЯ, PYTHON, СТЕГАНОГРАФІЯ, LSB, ЗОБРАЖЕННЯ.

ANNOTATION

Thesis on the topic "Algorithm and implementation of a cloud safe with a double bottom based on IM-networks" for the first (bachelor's) level of higher education specialization in 125 – Cybersecurity educational program: Cybersecurity, contains 27 figures, 1 appendix, 14 references links. The work is performed on 52 pages of general text and 43 pages of main text.

The main purpose of this thesis is to increase security of user files by developing and implementing a double bottom cloud safe algorithm based on IM-networks.

Cloud-safe algorithm based on Telegram was developed and implemented in the software product.

Present development can be used by citizens and the military both during martial law and in peacetime to store their private information.

SAFE, CRYPTOGRAPHY, DOUBLE BOTTOM, SYMMETRIC ENCRYPTION, PYTHON, STEGANOGRAPHY, LSB, IMAGE.

ЗМІСТ

ВСТУП

В останні роки саме месенджери стали невід’ємною частиною життя сучасної людини. Користувачі проводять багато часу у месенджерах, адже вони дозволяють їм обмінюватись не тільки текстовими повідомленнями, а й зображеннями, відео, аудіо, та багатьма іншими видами медіа файлів. Збільшення попиту на месенджери призвело до їх неминучого розвитку, завдяки чому компанії почали одна за одною підвищувати зручність користування, рівень безпеки аккаунтів та чатів, виділяти більше місця на серверах під зберігання відправлених медіа файлів.

Одночасно із зростанням споживання медіа контенту, цифровізацією усього суспільства – виникла необхідність розвитку технологій зберігання інформації. Саме зараз найбільшого поширення набувають сервіси хмарного зберігання даних, що дозволяють користувачам завантажити свої файли у хмару, та отримати до них доступ з будь-якого пристрою, та з будь-якого місця, якщо там є доступ до інтернету.

Завантаження великої кількості користувацьких файлів у мережу очікувано призвело до випадків отримання зловмисниками несанкціонованого доступу до чутливих персональних файлів користувачів, адже отримати логін та пароль від хмарного сейфу в інтернеті значно легше, ніж скачати інформацію з персонального комп’ютера жертви. Отримання даних авторизації до акаунту користувача автоматично означає скачування файлів зловмисниками, та їх перегляд, адже хмарні сервіси зберігають інформацію у незашифрованому вигляді.

Деякі програми надають користувачам можливість шифрувати свою інформацію, та навіть створювати файли, що являють собою сейфи з подвійним дном, що дуже позитивно відзначається на безпеці, але ці програми не надають можливості отримання хмарного доступу до зашифрованої інформації.

Метою цієї дипломної роботи є підвищення захищеності користувацьких файлів шляхом розробки та імплементації алгоритму хмарного сейфу з подвійним дном на основі ІМ-мереж.

Для досягнення мети поставлені такі задачі:

- Дослідити існуючі файлові сейфи, їх переваги та недоліки;
- Розробити алгоритм роботи файлового сейфу, із забезпеченням хмарного зберігання даних та можливістю організації подвійного дна;
- Виконати імплементацію розробленого алгоритму засобами мови програмування Python;

Об’єкт дослідження – процеси захисту конфеденційної інформації під час її зберігання

Предмет дослідження – алгоритми побудови файлових сейфів.

Дипломний проект складається з вступу, чотирьох розділів, висновків, списку джерел і додатків.

Перший розділ дипломної роботи присвячено аналізу існуючих програм ті сервісів, що надають можливість створення криптосейфу, або хмарного зберігання даних.

У другому розділі розглянуто теоретичні засади створення криптографічних сейфів та побудовано алгоритм, який буде реалізовано у програмному продукті.

Третій розділ описує програмну реалізацію алгоритму, та тестування програмного проекту.

Четвертий розділ містить аналіз робочого місця програміста з питань охорони праці, та деякі обов’язкові для виконання рекомендації.

1 АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ ПРОДУКТІВ, ЩО РЕАЛІЗУЮТЬ КРИПТОГРАФІЧНІ СЕЙФИ

1.1 Історія сучасної криптографії

Вивчення історії і розвитку криптографії допомагає зрозуміти що це таке, та чому вона стала невід'ємною частиною технологій збереження даних.

Наївна криптографія з'явилася ще до початку XVI століття, у ній характерним є використання будь-яких примітивних способів заплутування противника відносно змісту шифрованого тексту. На початковому етапі для цього використовувалися кодування та стеганографія, що є спорідненими, але не тотожними криптографії [1].

Наприкінці XV – початку XX століття з'явилася формальна криптографія. Цей етап пов'язаний з появою формалізованих та відносно стійких до ручного криптоаналізу шифрів. Поява надійних способів захисту інформації пов'язані з розвитком науки та торгівлі. Саме у цей час італійський архітектор Леон Батист Альберті один із перших запропонував багатоалфавітну підстановку. Його шифр у майбутньому отримав назву «Шифр Віженера».

Отже криптографія – це наука про математичні методи забезпечення конфіденційності, цілісності та автентичності інформації [2].

Сучасна криптографія з кінця 20-го століття використовує складні математичні розрахунки, для чого задіє розрахункові потужності комп'ютерів.

1.2 VeraCrypt

VeraCrypt – це безкоштовний інструмент для шифрування з відкритим кодом для Windows, MacOS та Linux. Він використовує технологію шифрування "на льоту", яка шифрує або дешифрує файли, до яких звертаються, без потреби втручання користувача [3].

VeraCrypt надає користувачам можливість створювати зашифровані файлові контейнери всередині файлів, додавати до них таємні розділи (подвійне дно), шифрувати несистемні зовнішні диски, створювати на них скритий том, зашифрувати розділ, або весь диск з системою, на якій знаходиться операційна система. Також є можливість створити приховану операційну систему. Можливості застосунку наведені на рисунку 1.1.

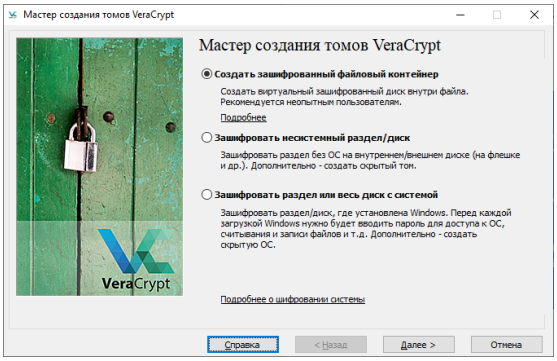


Рисунок 1.1 – Основні можливості застосунку VeraCrypt

При виборі опції створення зашифрованого файлового контейнеру програма надає можливість до файлу скритий том, або – подвійне дно. Подвійне дно потрібно у випадку, коли просто немає фізичної можливості не видати пароль від криптографічного контейнера, наприклад при вимаганні. У такому випадку з’являється можливість назвати пароль від відкритого тому, не надаючи доступу до дійсно цінної та чутливої інформації.

Том VeraCrypt може знаходитись у файлі, на жорсткому диску, або будь-якому іншому внутрішньому та зовнішньому накопичувачі. Контейнер можна переміщувати та видаляти як звичайний файл.

VeraCrypt надає можливість обрати алгоритм шифрування зовнішнього тому, в якому буде зберігатися контейнер багатьма алгоритмами шифрування, включаючи AES. (рис 1.2)

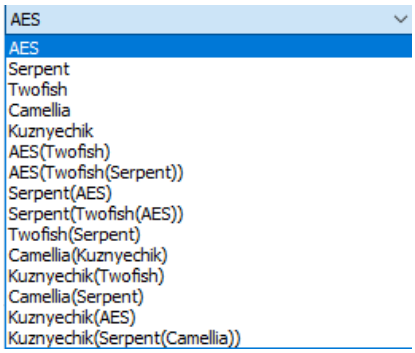


Рисунок 1.2 – Доступні у VeraCrypt алгоритми шифрування

Також можливо обрати алгоритм хешування, який буде використовуватися для збереження таємності паролю.

Після цього програма надає можливість обрати розмір контейнера, встановити паролі для зовнішнього та скритого томів, або навіть створити ключові файли, які можна зберігати на зовнішньому носії, та використовувати замість паролю, як додатковий рівень захисту. Також можливо обрати файлову систему, та розмір кластеру для форматування кожного із томів.

Серед плюсів програми можна виділити:

– Безкоштовне використання;
– Можливість створення контейнера з подвійним дном;
– Можливість обрати алгоритм шифрування;

Суттєвими мінусами слід зазначити:

– Відсутність можливості отримання доступу до інформації без наявності на комп’ютері програми VeraCrypt;
– Складний для розуміння середньостатистичного користувача інтерфейс з багатьма функціями;
– Відсутність можливості отримання доступу до файлу у хмарному режимі з будь-якого пристрою;

1.3 Google Drive

Google Drive (або Google Диск) – це сховище даних, яке дозволяє збереження користувацьких даних у хмарі на серверах компанії Google [4].

Сховище не використовує жодних алгоритмів шифрування, тому для отримання доступу до усієї інформації у файлах достатньо лише дізнатися логін та пароль від акаунту.

Google Drive надає користувачу широкий функціонал для комфортного перегляду файлів без скачування, створення та розподілення файлів за папками, та надання до них доступу іншим користувачам. Кожен користувач безкоштовно отримує 15 GB місця у хмарі, куди він може завантажити усі свої файли. Увесь доступний функціонал зазначено на рисунку 1.3.

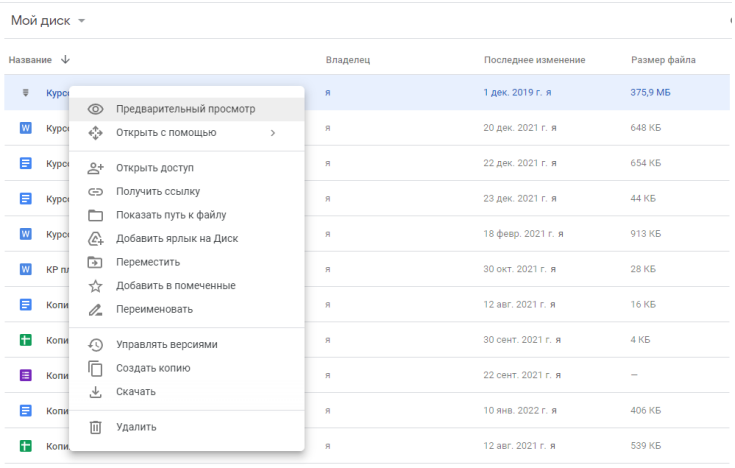


Рисунок 1.3 – Функціонал Google Drive

Серед плюсів програми слід відмітити:

– Безкоштовне використання;
– Простий та зрозумілий для середньостатистичного користувача інтерфейс;

– Можливість отримати доступ до інформації з будь-якого пристрою в будь-якому місці, де є доступ до інтернету;

Серед мінусів необхідно зазначити:

– Відсутність можливості шифрувати файли;
– Відсутність функціоналу для створення подвійного дна;

1.4 MEGA

MEGA – це надійний хмарний криптосейф для зберігання файлів користувача. Для шифрування файлів MEGA використовує алгоритм AES. Дані шифруються на боці клієнта. Усі дані користувачів зберігаються в хмарі и розшифровуються "на льоту", якщо власник бажає отримати до них доступ. Також MEGA надає можливість поділитися своїми файлами з іншими користувачами, або навіть надати посилання на один із своїх контейнерів (папок), що дозволить його перегляд усім, хто перейде за посиланням. Якщо контейнер зашифрований – окрім посилання користувачам необхідно буде ввести ключ, який повинен надати власник файлів.

MEGA надає новим користувачам 20 GB безкоштовного місця у хмарі, та має дружній інтерфейс за багатьма функціями. Окрім зберігання та розповсюдження файлів – сервіс надає можливість створити текстові, або голосові чати, щоб підтримувати спілкування з іншими користувачами. MEGAchat використовує технологію наскрізного шифрування даних. У чаті є можливість перевірити відбитки пальців користувачів з якими ви спілкуєтеся, передавши їх по окремому каналу, щоб впевнитися, що це саме та людина, за кого вона себе видає. Доступний функціонал хмарного сейфу наведено на рисунку 1.4.

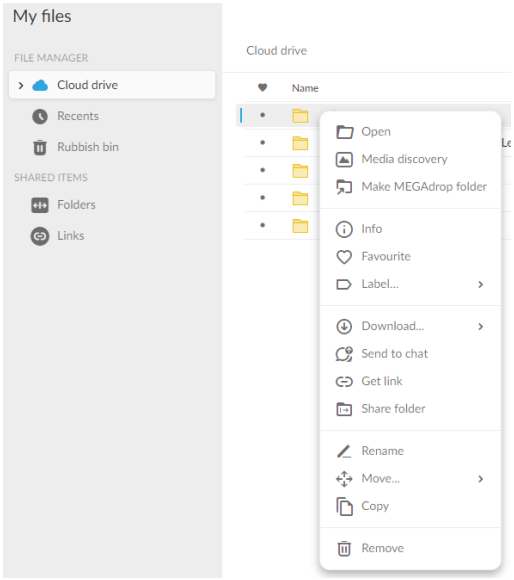


Рисунок 1.4 – Функціонал сейфу MEGA

Деякі користувачі критикують сервіс за те, що пароль від аканту являє собою ключ шифрування файлів, через що немає можливості скинути, або відновити пароль без втрати даних у сейфі. Також експерти криптографії зауважують, що шифрування на боці клієнта за допомогою javascript, код якого кожного разу знову завантажується з серверу може бути підмінено, або відключено в результаті отримання доступу до серверу третьою стороною. Через це користувач повинен повність довіряти сервісу.

Не зважаючи на усі переваги сервісу – користувачі не мають можливості створити контейнер з подвійним дном. Увесь функціонал сервісу MEGA наведено на рисунку 1.5.

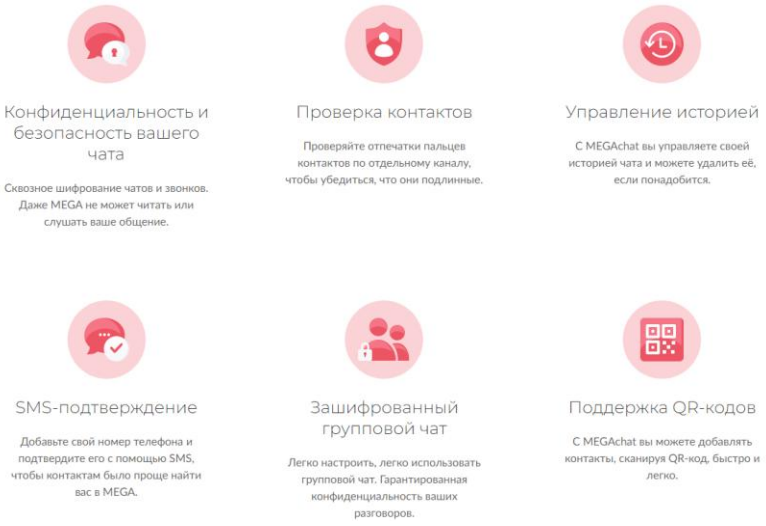


Рисунок 1.5 – Повний функціонал сервісу MEGA

Серед плюсів сервісу слід відмітити:

– Можливість безкоштовного використання;
– Шифрування файлів та діалогів надійним алгоритмом;
– Можливість спілкування з іншими користувачами;
– Зрозумілий середньостатистичному користувачу інтерфейс;

Серед мінусів потрібно зазначити:

– Відсутність можливості створення подвійного дна;
– Можливість компрометації шифрування у разі отримання зловмисниками доступу до сервера;
– Відсутність можливості відновити, або змінити пароль без втрати даних;

2 ТЕОРЕТИЧНІ ЗАСАДИ СТВОРЕННЯ КРИПТОГРАФІЧНИХ СЕЙФІВ

2.1 Системи парольного захисту

Одним із найважливіших процесів для дотримання конфіденційності інформації є обмеження доступу. Найбільш розповсюджений процес автентифікації – використання паролю.

Основна перевага парольного захисту — це його простота і звичність більшості користувачів. Паролі вже давно вбудовують в операційні системи та інші сервіси. При правильному використанні паролі можуть забезпечити прийнятний рівень безпеки, але за своєю природою їх слід вважати найслабшим засобом автентифікації. Для легкості запам’ятовування – паролі зазвичай роблять простими. Однак простий пароль легко вгадати, особливо якщо ви знаєте користувача, або провели мінімальну розвідку у відкритих джерелах. Іноді паролі не тримаються в секреті з самого початку, оскільки вони мають стандартні значення, зазначені в документації, і вони не завжди змінюються після встановлення обладнання. Введення паролю також можна підглянути, стоячи позаду користувача, або переглянувши записи з камер спостереження, якщо вони наявні у приміщені. Паролі часто повідомляють колегам, які, наприклад, можуть тимчасово змінити власника пароля на його робочому місці. В теорії правильним було б використання контролю доступу, але в реальності цього зазвичай ніхто не робить.

Одним із найпоширеніших механізмів підбору паролів є Brut-force, або "Метод грубої сили". Метод грубої сили є найбільш затратним по часу, але у кінці він 100% досягне результату. Цей метод перебирає усі можливі комбінації букв та символів, доступних до вводу, а підбір довгого, або складного паролю може займати десятки, або сотні років.

Метод підбору по словнику – практично ідеальний і швидкий метод для підбору легких, або типових паролів. Для цього метода використовується файл із сотнями тисяч варіантів паролів, які вводяться програмою один за одним в надії, що один з них підійде.

Rainbow table, або "райдужні таблиці" – це метод підбору паролів, створений в 2003 році, який став суттєвим ступенем в збільшенні швидкості підбору паролів. Цей метод - це спеціальний варіант таблиць пошуку, використовуючий механізм розумного компромісу між часом пошуку по таблиці та займаємою пам'яттю (time-memory tradeoff). Райдужна таблиця створюється побудовою ланцюгів можливих паролів. Кожен ланцюг починається із випадкового можливого паролю, після чого підвергається дії хеш-функції і функції редукції. Функція редукції перетворює результат хеш-функції в деякий можливий пароль. Після цього до можливого пароля знов застосовується хеш-функція і ланцюг повторюється знов. В ланцюг записуються лише початкова та кінцеве значення. Спрощений приклад роботи райдужних таблиць наведено на рисунку 2.1.

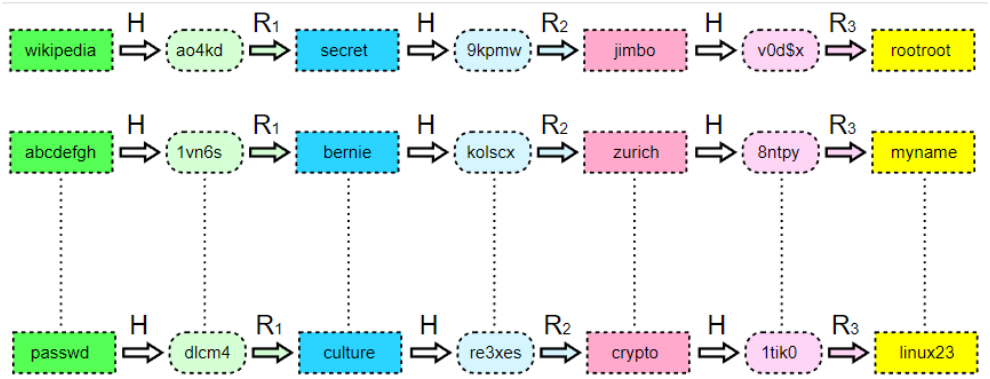


Рисунок 2.1 – Схема спрощеної райдужної таблиці

Єдиний недолік райдужних таблиць полягає лише у часі генерації самих таблиць [5].

Перші системи парольного захисту на операційних системах з'явилися ще у Windows 95/98, де пароль зберігався у PWL файлі [6]. Такі файли зберігалися у каталозі операційної системи у зашифрованому вигляді, але декілька серйозних недоліків:

- Усі паролі були перетворені до верхнього регістру, що значно зменшує кількість можливих паролів;
- Використані алгоритми шифрування MD5 та RC4 дозволяли швидке шифрування паролю, але достовірний пароль Windows повинен бути довжиною не менш 10 символів;
- Паролі зберігалися у зашифрованому файлі у відкритому вигляді;

Для вирішення проблеми можливого розкриття паролю при дешифруванні файлу, або його перехоплення під час автентифікації у наш час використовуються хеш-функції.

Хеш-функція – функція, яка здійснює перетворення вхідних даних довільного розміру в дані з фіксованим розміром [7].

Алгоритм хешування – це процес ін’єкції з множини M масивів двійкових вхідних даних m довільної довжини в двійковий рядок h = H(m) вихідних даних, де H(m) – хеш-дайджест, M – прообраз [8].

Для хеш-функції існують вимоги, яким вона повинна задовольняти:

а) Для будь-якого $m \in M$ легко обчислити h = H(m);

б) Для будь-якого h складно обчислити m таке, шо h = H(m);

Через те, що хеш-алгоритм перш за все спрямований на максимальне стиснення даних, можлива поява однакових хешів, тобто колізії. Через це хеш-функція повинна відповідати ще одній вимозі:

в) Для будь-якого $m \in M$ складно знайти $m' \in M$ таке, що $h = H(m) = H(m')$;

Використання хеш-функцій є ідеальним для зберігання паролів. На серверах паролі зберігаються у вигляді хешу, тому після вводу паролю користувачем – він хешується і хеш-код паролю порівнюється з хеш-кодом на сервері. Якщо хеш повністю співпадає – користувачу надається доступ.

2.2 Симетрична криптографія. Advanced Encryption Standard

Симетричні криптосистеми – це спосіб шифрування, у якому в шифруванні та дешифруванні використовується один і той самий криптографічний ключ. Ключ шифрування є таємний і повинен зберігатися в тайні обома сторонами. На

всьому маршруті слідування криптограми повинні виконуватися міри захисту доступу до каналу [9]. Алгоритм шифрування зазначається сторонами до початку обміну повідомленнями.

Блочний шифр є однією із різновидів симетричного шифру. Він оперує блоками біт фіксованої довжини, розмір яких знаходиться в межах 64-256 біт. Якщо початковий текст менше розміру блоку, його доповнюють перед шифруванням. Блочний шифр здатен шифрувати одним ключем одне, або декілька повідомлень, які сумарно більші ніж довжина ключа. Передача по каналу малого в порівнянні з повідомленням ключа – задача значно більш проста.

До переваг блочних шифрів відносять схожість процедур шифрування і дешифрування, які зазвичай відрізняються лише порядком дій. Це спрощує створення пристроїв шифрування, дозволяючи використовувати одні й ті самі блоки для шифрування і дешифрування.

Одним із найпоширеніших алгоритмів шифрування є Advanced Encryption Standard (AES), ще відомий під назвою Rijndael. Це симетричний алгоритм блокового шифрування, який набув своєї розповсюдженості після прийняття його американським стандартом шифрування у результаті конкурсного відбору. AES замінив собою застарілий стандарт DES [10].

У алгоритма AES встановлена фіксована довжина у 128 біт, у той час коли розмір ключа може варуватися у значеннях 128, 192 та 256 біт. Фіксований розмір надає алгоритму можливість оперувати масивом 4 x 4 байти, це називають станом [11].

Для 128 бітного ключа алгоритм застосовує 10 раундів, послідовно використовуючи операції subBytes(), shiftRows(), mixcolumns(), xorRoundKey().

SubBytes() проводить обробку кожного байту стану незалежно. Для цього проводиться нелінійна заміна байтів через таблицю замін S-box. Завдяки операції subBytes() забезпечується нелінійність алгоритму шифрування.

До кожного байту b у S-box застосовується наступна операція

$$B'_i = b_i \oplus b_{(i+4) mod\ 8} \oplus b_{(i+5) mod\ 8} \oplus b_{(i+6) mod\ 8} \oplus b_{(i+7) mod\ 8} \oplus C_i$$

Примечание [КЧ1]: Поменять шрифт в формуле на Courier New

де $0 \leq i < 8$, і де b_i це і-ий біт константи $c = 63_{16} = 99_{10} = 01100011_2$.

Таким чином досягається забезпечення алгоритмом стійкості до атак, що засновані на простих алгебраїчних властивостях. Приклад S-box буде наведено на рисунку 1.1.

\	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Рисунок 2.2 – S-box

ShiftRows() циклічно трансформує рядки таблиці State, зсуваючи рядки на r байтів по горизонталі в залежності від його номера. Номера кожного рядка розуміються як r = 0, r = 1 і так далі. Після цього перетворення кожна вихідна колонка стану буде зберігати у собі байти кожної початкової колонки стану [12]. Алгоритм трансформації наведено на рисунку 1.2.

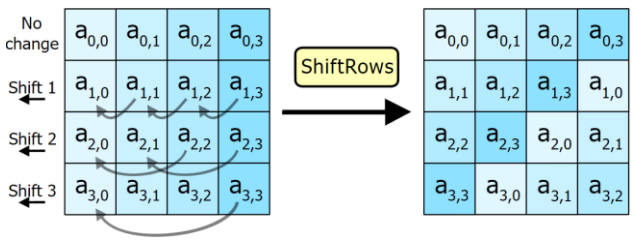


Рисунок 2.3 – Алгоритм роботи функції ShiftRows()

MixColumns() використовує зворотну лінійну трансформацію для змішування чотирьох байтів кожної колонки стану. Кожна з колонок трактується функцією як поліном четвертого степеня. Над ними виконується множення в $GF(2^8)$ по модулю $x^4 + 1$ на многочлен $c(x) = 3x^3 + x^2 + x + 2$. Поєднання функцій mixColumns() та shiftRows() вносить в шифр дифузію [12]. Алгоритм роботи функції наведено на рисунку 1.3.

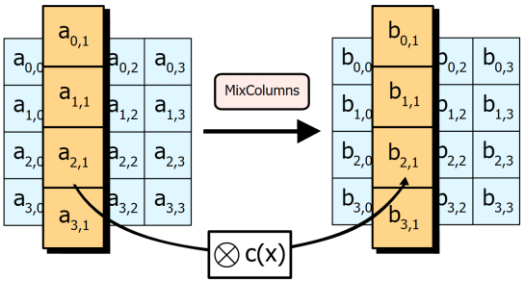


Рисунок 2.4 – Алгоритм роботи функції MixColumns()

AddRoundKey() об’єднує RoundKey кожного рядка з таблицею стану. При кожному раунді RoundKey виходить із CypherKey через процедуру KeyExpansion. RoundKey завжди має той самий розмір, що и State. Кожен байт RounKey проходить процедуру побітового XOR з кожним байтом State [12]. Це наведено на рисунку 1.4.

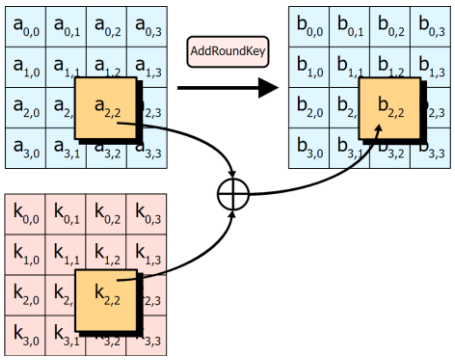


Рисунок 2.5 – Алгоритм роботи функції AddRoundKey()

2.3 Взаємодія з Telegram

Для налагодження взаємодії програми з Telegram було обрано бібліотеку Pyrogram. Pyrogram – це сучасна асинхронна бібліотека, яка дозволяє легко взаємодіяти з API Telegram через обліковий запис користувача, або з телеграм-ботами через ідентифікатор бота.

Програма буде побудована на взаємодії користувача через графічний інтерфейс додатку з юзер-ботом у Telegram. Юзер-бот – це акаунт у Telegram, який зареєстрований у мережі як звичайний користувач, тобто не має ідентифікатора бота, але виконує заданий йому алгоритм дій.

Для взаємодії з юзер-ботом потрібно задати параметри api_id та api_hash, які можна отримати на офіційному сайті Telegram.

Отже першим кроком для користувача буде запуск програми, відображення графічного інтерфейсу та запит введення паролю, який і визначить до якого із рівнів сейфу звертається користувач. Цей алгоритм наведений на рисунку 2.6.

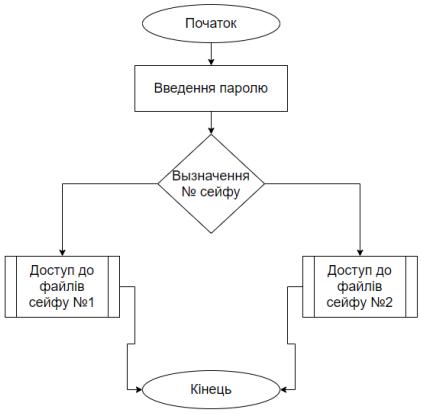


Рисунок 2.6 – Алгоритм обрання користувачем відкритого, чи прихованого рівня сейфу.

Наступним кроком незалежно від введення відкритого, чи прихованого пароля користувач повинен побачити однаковий графічний інтерфейс з можливістю побачити вміст цього рівня сейфу, можливість скачати, завантажити, видалити інформацію, перезавантажити вікно відображення вмісту рівня сейфу,

та можливість закрити програму. Обидва рівні сейфу повністю ідентичні за можливостями (рис 2.7).

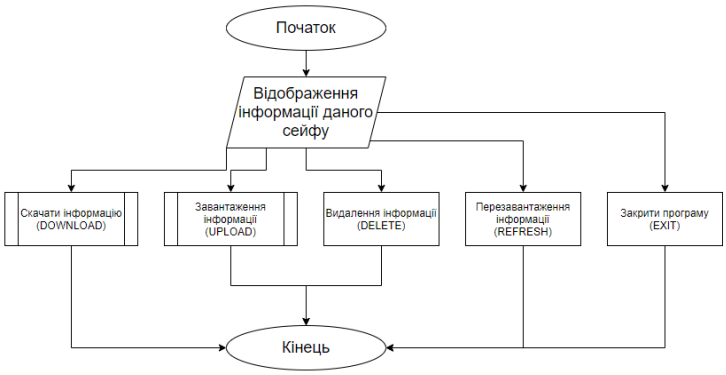


Рисунок 2.7 – Отримання користувачем доступу до будь-якого рівня сейфу

Для збільшення можливостей для зберігання чутливої інформації – було вирішено надати користувачу можливість імплементації стеганоповідомлення в картинку. Імплементація має виконуватися методом LSB, який є вразливим до перетворень зображення, тому зображення будуть відправлятись в Telegram без стиснення [13], [14].

Також користувачу має надаватися можливість шифрувати, або не шифрувати зображення із стеганоповідомленням. Файли без розширення .jpg і .png мають бути зашифровані і відправлені в Telegram. Стандартним алгоритмом шифрування був обраний AES.

Отже опишу запропонований алгоритм завантаження файлів. Крок 1. Надати користувачу можливість обрати файл зі списку файлів на комп’ютері. Крок 2. Перевірити, чи є обраний файл зображенням. Якщо це зображення – перейти до кроку 4. Якщо це не зображення – виконати крок 3 і перейти до кроку 8. Крок 3. Зашифрувати файл алгоритмом AES. Крок 4. Запропонувати користувачу зашифрувати зображення алгоритмом AES. Якщо користувач погоджується – виконати крок 3 і перейти до кроку 5. Якщо відмовляється – перейти до кроку 5. Крок 5. Запропонувати користувачу додати у зображення

стеганоповідомлення. Якщо користувач згоден – перейти до кроку 6. Якщо користувач не бажає додавати стеганоповідомлення – перейти до кроку 8. Крок 6. Надати користувачу можливість ввести стеганоповідомлення. Крок 7. Надати користувачу можливість зберегти ключ від стеганоповідомлення. Крок 8. Завантажити файл у Telegram.

Більш наглядно описаний алгоритм надано у блок-схемі на рисунку 2.8.

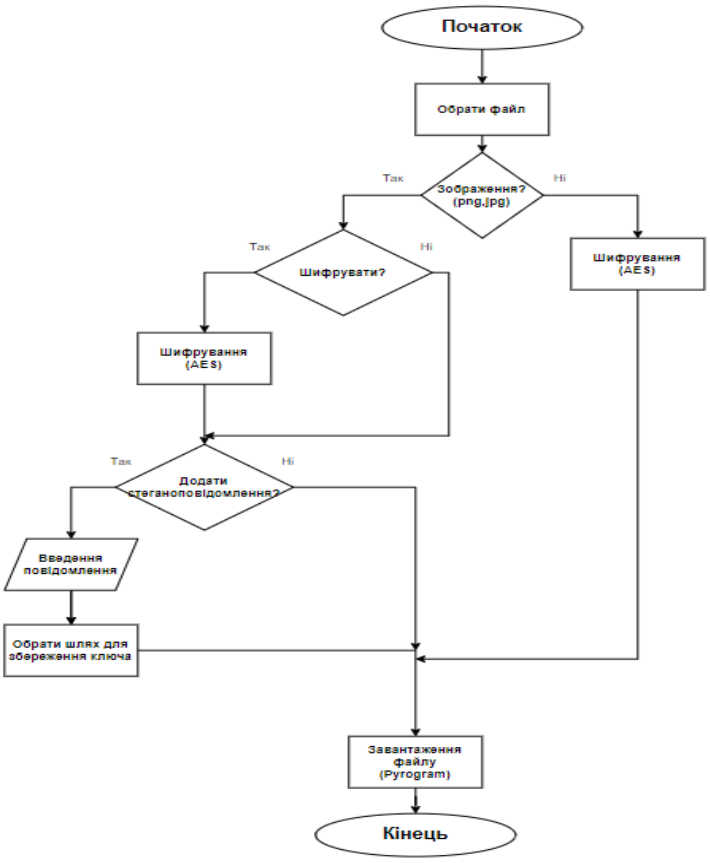


Рисунок 2.8 – Алгоритм завантаження файлу в сейф

Для скачування файлу з сейфу треба виконати дзеркальні дії з відміною лише у тому, що замість шифрування буде виконуватися дешифрування, а при виборі зображення – користувачу має надаватися можливість обрати файл ключа для отримання стеганоповідомлення. У разі обрання правильного ключа – користувач повинен отримати можливість прочитати повідомлення.

3 ПРАКТИЧНА ІМПЛЕМЕНТАЦІЯ АЛГОРИТМУ ХМАРНОГО СЕЙФА З ПОДВІЙНИМ ДНОМ

3.1 Вибір середовища для розробки програми

3.1.1 Python

Python – мова програмування високого рівня з динамічною строгою типізацією і автоматичним керуванням пам’яттю, орієнтованою на підвищення продуктивності розробника, легкості читання коду, та підвищення його якості.

Python надає можливість створення і використання сторонніх користувацьких модулів, що значно спростить взаємодію з API Telegram, та прискорить роботу.

Для імплементації алгоритму мовою Python будуть використані наступні модулі:

– Random;
– Hashlib;
– OS;
– Pillow;
– PyCryptodome;
– Tkinter;
– Pyrogram;

Random – це модуль, який надає функції для генерації випадкових чисел, букв, або випадкових елементів послідовності.

Hashlib – це модуль, який надає функціонал для реалізації хешування. Хешування SHA 256 буде використане для хешування паролів.

OS – це модуль, який надає функціонал для роботи з операційною системою. Цей модуль буде використаний для взаємодії з файлами на комп’ютері користувача.

Pillow (PIL) – модуль, який додає функціонал обробки зображень різних форматів, розширює та спрощує варіанти взаємодії з ними.

PyCryptodome – це модуль, який дозволяє розробнику спростити для себе процес шифрування, тому що модуль зберігає у собі велику кількість готових до

викростання шифрів. Цей модуль буде використаний для шифрування файлів алгоритмом AES.

Tkinter – модуль, який додає у Python функціонал для створення графічного інтерфейсу, та налагодження взаємодії з ним користувача.

Pyrogram – модуль для взаємодії з API Telegram. Буде використаний для взаємодії з юзер-ботом для зберігання та скачування інформації користувачів.

3.1.2 PyCharm

PyCharm – безкоштовне кросплатформне середовище розробки мовою Python від компанії JetBrains. PyCharm має приємний та зрозумілий графічний інтерфейс, окрім того програма надає можливість завантаження відсутніх модулів у один клік. На рисунку 3.1 зображено графічний інтерфейс PyCharm.

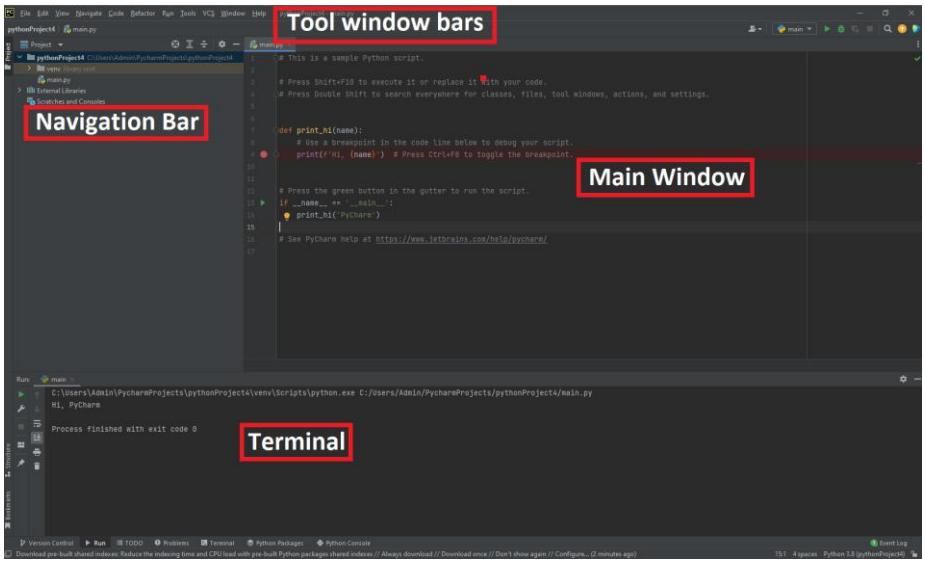


Рисунок 3.1 – Інтерфейс PyCharm

Terminal – це вікно емуляції терміналу, яке дозволяє виконувати системні команди, спрямовані до Python, тим самим взаємодіяти із програмою через консоль. Також вікно терміналу являє собою вікно виводу інформації, запитів та сповіщення про помилки, допущені під час роботи програми.

Navigation bar – це вікно навігації. У цьому вікні відображається поточний проект, та усі пов’язані із ним файли. Також при бажанні у цьому вікні є можливість обрати інші файли Python, проекти та відкрити їх.

Tool window bars – це вікно з меню інструментів. Завдяки цьому вікно у користувача є можливість створювати, відкривати, додавати проекти, розділяти програму на декілька вікон, виконувати налаштування PyCharm, а також обирати відображення додаткових вікон із меню View.

Main Window – вікно набору коду. Саме у цьому вікні відбувається написання коду програми.

На рисунку 3.2 наведено пустий скрипт у PyCharm.

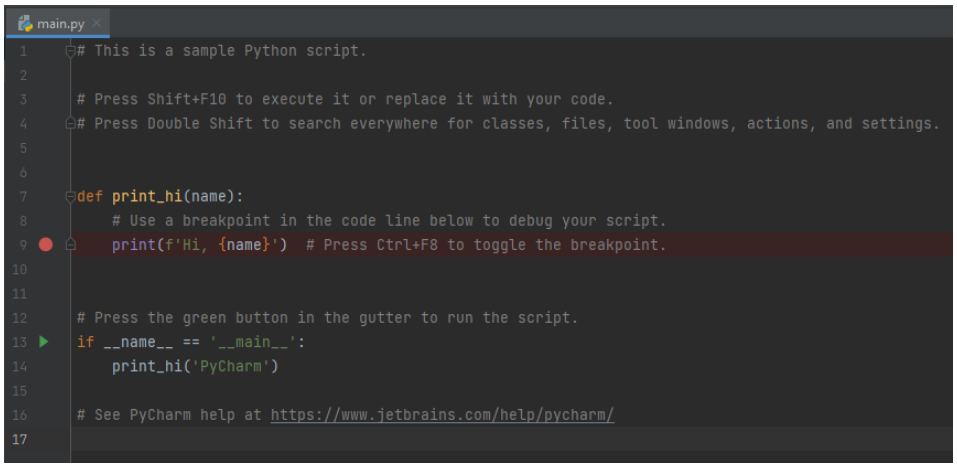


Рисунок 3.2 – Пустий скрипт

3.2 Опис програми

Готова програма на 100% підтримує увесь запланований на етапі створення алгоритму функціонал. Користувач взаємодіє з графічним інтерфейсом програми, не підозрюючи, що його файли зберігаються саме у Telegram. Це додає програмі рівень захисту, у випадку якщо зловмисники спробують знайти ваші вайли.

При запуску програма запитує у користувача пароль від хмарного сейфу (рис 3.3).

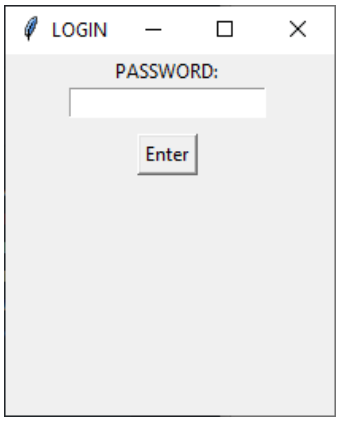


Рисунок 3.3. – Меню введення паролю

Введений пароль визначить, до якого із рівнів сейфу звертається користувач, або відхилить авторизацію, якщо такого пароля не існує.

Обидва паролі зберігаються у файлі data.py у вигляді хеш-функції (рис. 3.4).

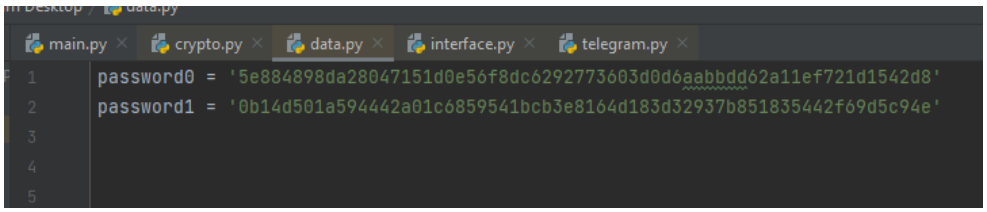
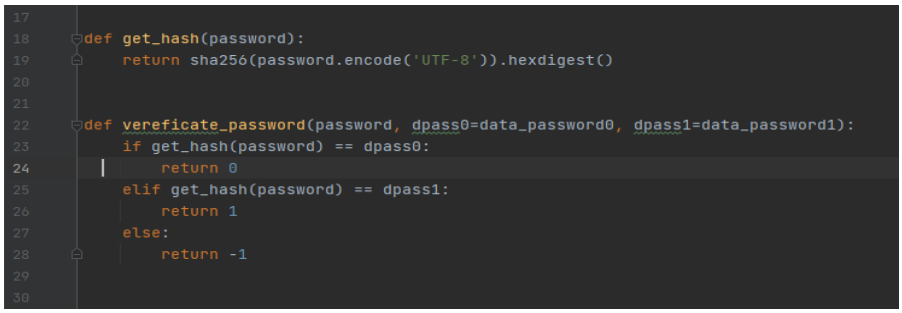


Рисунок 3.4 – Паролі від обох рівнів сейфу

Після вводу користувачем паролю відбувається його хешування за допомогою SHA 256 і порівняння хешу введеного паролю із хешем існуючих паролів. Якщо один із паролів співпадає – користувачу встановлюється режим доступу до першого чи другого рівня сейфу. На рисунку 3.5 наведено реалізацію цієї процедурі у коді.



```
def get_hash(password):
    return sha256(password.encode('UTF-8')).hexdigest()


def vereficate_password(password, dpass0=data_password0, dpass1=data_password1):
    if get_hash(password) == dpass0:
        return 0
    elif get_hash(password) == dpass1:
        return 1
    else:
        return -1
```

Рисунок 3.5 – Функція парольного доступу

Отримавши доступ до сейфу – користувач повинен побачити список свої файлів та набір функціональних кнопок по взаємодії з ними, що відповідно і відбувається (рис. 3.6).

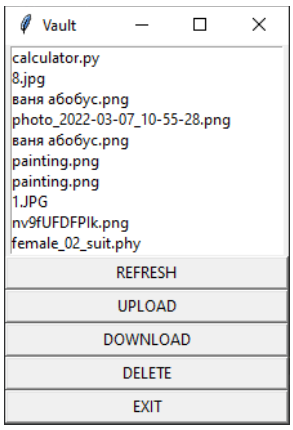


Рисунок 3.6 – Меню сейфа

Наступним кроком було реалізовано надання користувачу можливість завантажити у сейф свої файли. Завантажені файли повинне бути перевірене на те, чи це зображення. Якщо це зображення – окрім надання можливості шифрування користувачу надається можливість додати стеганоповідомлення. Якщо це не зображення – файл шифрується алгоритмом AES і відправляється в Telegram. Основна частина функції відправки файлу відображена на рисунку 3.7.

```
def send_file(): # відправка файлу
    def image_param(patch_to_file, mode): # якщо зображення
        def work(encrypt, message): # варіанти по обробці фото
            if message == '' and encrypt:
                crypto.crypt(path_to_file, mode)
            elif message == '' and encrypt == False:
                telegram.send_file(patch_to_file, mode)
            elif message != '' and encrypt:
                crypto.stega_encrypt(patch_to_file, message,
                                     filedialog.askdirectory())
                crypto.crypt(f'./TEMP/{patch_to_file.split("/")[-1].replace("jpg", "png")}', mode)
            elif message != '' and encrypt == False:
                crypto.stega_encrypt(patch_to_file, message,
                                     filedialog.askdirectory())
                telegram.send_file(f'./TEMP/{patch_to_file.split("/")[-1].replace("jpg", "png")}', mode)
```

Рисунок 3.7 – Функція відправки файлу

Також на рисунку 3.8 відображена функція вбудови стенагоповідомлення в зображення.

```
def stega_encrypt(path_to_image, message, key_patch): # додавання повідомлення в зображення
    keys = []
    img = Image.open(path_to_image)
    draw = ImageDraw.Draw(img)
    width = img.size[0]
    height = img.size[1]
    pix = img.load()
    with open(f'{key_patch}/{path_to_image.split("/")[-1].split(".")[0]}.key', 'w') as f:
        for elem in ([ord(elem) for elem in message]):
            key = (randint(1, width - 10), randint(1, height - 10))
            g, b = pix[key][1:3]
            draw.point(key, (elem, g, b))
            f.write(str(key) + '\n')
    img.save(f'./TEMP/{path_to_image.split("/")[-1].replace("jpg", "")}png', "PNG")
```

Рисунок 3.8 – Функція вбудови стеганоповідомлення в зображення

Шифрування та дешифрування файлів алгоритмом AES як і зазначалося виконано за допомогою модуля PyCryptodome (рис. 3.9).

```
def crypt(file, mode):
    file_in = open(file, 'rb')
    file_data = file_in.read()
    file_in.close()

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

    file_out = open(f'./TEMP/{file.split("/")[-1]}', "wb")
    [file_out.write(x) for x in (cipher.nonce, tag, ciphertext)]
    file_out.close()

    telegram.send_file(f'./TEMP/{file.split("/")[-1]}', mode)
    remove(f'./TEMP/{file.split("/")[-1]}')
```

Рисунок 3.9 – Шифрування файлів алгоритмом AES

Ключ шифрування зберігається у файлі програми data.py. Окрім нього там знаходяться і api_id і api_hash, необхідні для взаємодії з юзер-ботом (рис. 3.10).

```
3   file_password = 'yw3OjMQyJekh0'
4   api_id = 1181
5   api_hash = "b722a900a01a99891a728bf871c"
6
```

Рисунок 3.10 – Вміст файлу data.py

3.3 Тестування програми

Тест № 1. Введення неправильного паролю.

Очікування: програма виведе вікно з помилкою.

Результат (рис. 3.11):

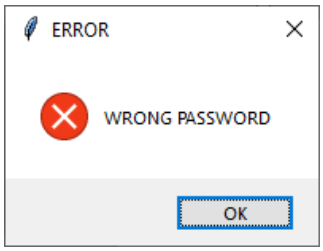


Рисунок 3.11 – Помилка при введені неправильного паролю

Тест № 2. Вбудова в зображення 900x900 пікселей занадто великого стеганоповідомлення.

Очікування: випадкові пікселі будуть пересвітлені синім кольором.

На рисунку 3.12 вказано оригінальне зображення.

Рисунок 3.12 – Оригінальне зображення для тесту № 2

Результат (рис. 3.12):

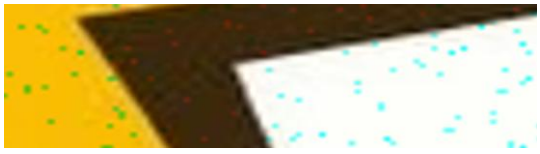


Рисунок 3.12 – Фрагмент зображення із стеганоповідомленням

На рисунку 3.13 зображена частина вбудованого повідомлення.

Рисунок 3.13 – Частина вбудованого в зображення стеганоповідомлення

Реалізація алгоритму LSB розбиває повідомлення на ASCII код і вбудовує цей код у синій елемент кожного випадкового пікселя. 1 символ = 1 байту = 8 біт. В один піксель вбудовується 2 біти, щоб уникнути висвітлення синього кольору, але оскільки повідомлення занадто велике – алгоритму не вистачає місця у вільних пікселях, тому значення синього для випадкових пікселів змінюється занадто сильно.

В третьому розділі було виконано останній пункт поставлених задач, тобто імплементація розробленого алгоритму мовою Python.

В третьому розділі детально описана інструкція з користування програмою, описані найважливіщі моменти у коді, а також проведено тестування вразливостей.

4 ОХОРОНА ПРАЦІ

З усіх можливих видів діяльності – саме на робочому місці людина зазвичай проводить чи не найбільше часу. Як для працівника, так і роботодавця дуже важливим є забезпечення безпечного середовища на робочому місці. Безпечне робоче місце позитивно впливає на працівника, підвищуючи його мотивацію, демонструючи його важливість для підприємства, створюючи комфортні умови, але й безпосередньо підвищує ефективність кожного окремого працівника, знижує витрати підприємства на лікування та відшкодування збитків здоров'ю, які працівники можуть зазнати на робочому місці.

Від дотримання вимог безпеки при організації робочого місця виграє і безпосередньо роботодавець, адже здоровий працівник працює більш продуктивно, проводить більше часу на робочому місці, кількість лікарняних відпусток знижується до мінімуму, а на більш довгому відрізку часу, обчислюваному роками, або десятиріччями, цей працівник зможе продовжувати виконувати свої робочі обов'язки на підприємстві, не залишаючи його через проблеми зі здоров'ям.

Всупереч деяким міркуванням, робота за комп'ютером при недотриманні відповідних заходів безпеки може дуже швидко завдати невиправних збитків здоров'ю працівника. Ось деякі із шкідливих факторів, що можуть впливати на працівника під час роботи за комп'ютером:

- Підвищена або понижена вологість повітря
- Відсутність, або недостача природного освітлення
- Недостатня освітленість робочої зони
- Підвищений рівень шуму на робочому місці.
- Підвищена температура поверхонь обладнання

Під час роботи людина зазвичай відтворює одноманітні рухи впродовж довгого проміжку часу. Для запобігання хронічним захворюванням, які викликає робота за комп'ютером і розроблені правила організації робочого місця. Згідно з документом ДСанПіН 3.3.2.007-98, конструкція стола працівника має відповідати сучасним ергономічним вимогам, при цьому забезпечуючи оптимальне

розміщення робочого обладнання на поверхні. Висота, ширина і глибина робочого столу регулюється у тому ж самому документі. Окрім того документ зазначає необхідні параметри стільців, висоту та ширину спинки, радіус кривизни його горизонтальної площини, довжину та ширину підлокітників, м'якість поверхні стільця, наявність підставки для ніг, та її параметри, та багато іншого, що можна знайти у документі. Дисплей має бути розташований на відстані 600-700мм від очей користувача, градус його нахилу, розташування клавіатури, інших пристроїв також регламентовано.

Захисту від усіх різновидів комп'ютерних випромінювань регламентовано досягати шляхом використання приекранних фільтрів, засобів індивідуального захисту очей та інших засобів, які були випробувані у акредитованих лабораторіях та отримали гігієнічний сертифікат. Якщо робоче місце оснащене лазерним принтером, воно повинно відповідати вимогам СанПіН №5804-91.

Документ ДСН 43.3.6 037-99 регламентує санітарні норми для шумів, інфра- та ультразвук. Ці норми повинні дотримуватись на усіх підприємствах та установах без виключення. Нормовані параметри постійного шуму на робочому місці працівника, являють собою рівні звукових тисків у октавних смугах з середньогеометричними частотами 31,5; 63; 125; 500; 1000; 2000; 4000; 8000 Гц в децибелах.

Також документом нормовані характеристики середнього рівня звукового тиску, параметри непостійного шуму, дози шуму, різноманітні параметрі інфра- та ультразвуку, що нормуються за рівнями звукового тиску у октавних смугах частот. Вимірювання таких рівнів шуму може проводитись за допомогою індивідуальних дозиметрів шуму з встановленими у документі параметрами та при умові, що усі прилади перевірені органами Держстандарту.

У офісах компаній зазвичай знаходиться багато комп'ютерної техніки, робота якої потребує охолодження внутрішніх елементів, для чого використовуються вентилятори. Неправильний підбір вентиляторів призведе до підвищення рівню шуму у приміщенні, тому важливим параметром при виборі комп'ютерів є характеристики внутрішніх вентиляторів. Окрім вентиляторів

можна використовувати водне охолодження, що складається із трубок всередині корпусу, по яким проходить охолоджена рідина. Це практично безшумний варіант, але він є дуже фінансово затратним, адже ця технологія не використовується настільки масово і є достатньо дорогою. Якщо у офісі є приміщення, яке виділяє підвищений рівень шуму, у якому постійно не знаходяться люди, як наприклад серверна – слід використовувати спеціальні звукопоглиначі, якими обшивається приміщення, аби не дозволити розповсюдження звуку за межі приміщення, або на етапі проектування стін подумати про більш просунуті технології звукопоглинання.

Параметри освітленості регулюються будівельними нормами, а саме документом ДБН В.2.5-28-2006. Мінімальна освітленість робочої поверхні при роботі з документами, або за комп'ютером складає 500 лк. В вечірній час природна освітленість не досягає такої відмітки, тому освітлення на робочому місці являє собою комбінацію природного та штучного освітлення. При цьому треба розуміти, що навіть у денний час природне освітлення може не досягати вказаної відмітки через погодні умови, або специфіку розташування будівлі, тому зазвичай штучне освітлення є невід'ємною частиною робочого місця навіть вдень. Окрім наявності самого освітлення, важливим залишається його розташування у приміщенні, вид джерела освітленості та його безпечність для працівників.

Загальне освітлення має бути рівномірним та не сліпучим. Локальне освітлення на робочому місці не повинно створювати відблисків. З цим можуть допомогти світильники з рефлекторами, які відбивають світло у потрібному напрямку. Комбінація освітлення повинна забезпечувати для працівників правильне сприйняття відтінків світла, не сліпити його. Професії, пов'язані з кропіткою роботою з невеликими елементами потребують відразу двох видів місцевого освітлення: спрямованого і розсіяного. Це відноситься і до фахівців зі збирання комп'ютерної техніки, адже при роботі з одним невеликим об'єктом йому доводиться використовувати інші, які знаходяться поруч. Зазвичай у таких випадках поруч із робочим місцем розташовують настільні лампи на струбцинах чи прищіпках, які можна легко адаптувати до потреб у освітленні.

Мікроклімат на робочому місці також являє собою невід'ємну частину безпеки. За оптимальних умов у працівника повинна підтримуватися постійна температура тіла на рівні 36,6 °С. Кількість утворюваного організмом тепла залежить від фізичного стану працівника та його фізичного навантаження, а рівень тепловіддачі – від умов мікроклімату у приміщенні. Найбільша кількість тепла втрачається організмом завдяки випромінюванню крізь шкіру, на що власне впливає багато мікрокліматичних факторів, як вологість у приміщенні, швидкість повітря та його температура, тощо. Норми мікроклімату вказані у документі ДСН 3.3.6.042-99. Документ розподіляє важкість робіт за категоріями I – III, залежно від витрати енергії під час їх проведення. Кожна з цих категорій потребує різних підходів до регуляції мікроклімату з боку роботодавця. Також документом зазначається поняття теплого та холодного періоду року, де середня температура зовнішнього середовища вище +10 °С та нижче +10 °С відповідно.

Для досягнення відповідних показників мікроклімату слід використовувати зволожувачі повітря, або вологопоглиначі, у залежності від початкової вологи у приміщенні. Оптимальна вологість повітря становить 40-60%, а швидкість повітря - 0,1м/сек. Відхилення від цих показників може призвести до хвороби, адже чим вище рівень вологості, тим більше тепла віддає організм людини, отже змінюється температура.

Норма температури на робочому місці для холодного періоду року складає 22-24°С, та 23-25°С для теплого. Для обігріву приміщень слід використовувати налагоджену вентиляцію, кондиціонування та спеціальні прилади-обігрівачі, які пройшли тестування і мають відповідний сертифікат. Важливо пам'ятати, що під час обігрівання зменшується вологість повітря, отже її треба підтримувати.

Деякі працівники під час роботи за комп'ютером схильні нехтувати правилами про заборону куріння на робочому місці. Згідно із законодавством України, куріння тютюнових виробів заборонено на території усіх підприємств та установ, крім спеціально виділених для цього місць. У таких місцях розміщуюсь табличку зі знаком «Місце для куріння».

Аналіз пожежної безпеки і вибір заходів і засобів пожежної безпеки

Ризик пожежі – один із найнебезпечніших факторів на будь-якому робочому місці, отже слід дуже відповідно ставитися до правил та норм пожежної безпеки. Виконання потреб для забезпечення пожежної безпеки однаково цікавить як працівників, так і роботодавців. Роботодавець зацікавлений у збереженні майна підприємства та здоров'я працівників, у той час як працівників також цікавить власне здоров'я, та збереження робочого місця.

За собою найбільшу кількість ризиків для виникнення пожежі на робочому місці програміста звичайно несуть електроприлади, а саме – їх неправильна експлуатація, або помилки допущені на етапі монтажу обладнання. До небезпечних факторів, які можуть спричинити пожежу при роботі з електрообладнанням можна віднести:

- Недбалість під час монтажу розеток, прокладання дротів, порушення вимог до застосування матеріалів, або пошкодження ізоляції дроту;
- Використання в мережі більшої кількості електроприладів, ніж та, на яку вона розрахована, що може призвести до перенавантаження мережі;
- Використання несправного обладнання, пошкоджених приладів, неізольованих дротів, або потрапляння води у електромережу може призвести до короткого замикання;

Серед небезпечних факторів пожежі слід зазначити:

- Вірогідність вибуху;
- Порушення цілісності електричних, водних та інших комунальних мереж, що не призначені до використання у випадку порушення цілісності, та можуть становити загрозу для людини;
- Руйнування споруд;
- Відкрите полум'я;
- Підвищена температура повітря;
- Викид токсичних продуктів горіння;
- Згорання кисню та його подальший недостатній вміст у повітрі;
- Задимлення;

Документ НАПБ Б.03.002-2007 розділяє типи приміщень за категоріями, у залежності від вірогідності вибохо- або пожежонебезпеки через присутність у приміщені речовин, газів, горючого пилу, вибухонебезпечних рідин, температури, під якою перебувають перелічені компоненти.

Документ виділяє категорії А, Б, В (Горючі гази, легкозаймисті, горючі та складногорючі рідини, речовини, матеріали, які можуть при взаємодії з водою, киснем повітря, або один з одним вибухати та горіти), Г, Д.

Продовжуючи аналізувати робоче приміщення програміста, взявши його розмір 50м^2, можна визначити, що це приміщення підпадає під категорію В, через наявність виробів, що складаються з продуктів переробки нафти: пластмаса, стільці, ізоляція, та компоненті ПК. Компоненти комп'ютера під час роботи перебувають у стані нагріву, що також може спричинити пожежу. Клас пожежі у такому приміщені буде відповідати класу В (горючі рідини, або тверді речовини, що розплавляються при нагріванні).

Слід визначити тип вогнегасників, що будуть використовуватися у такому приміщені. Найкраще для використання у приміщеннях з ПЕОМ підійдуть порошкові та вуглекислотні вогнегасники, але порошковий слід застосовувати тільки у випадку, коли всі залишили приміщення, що не завжди може бути вчасно.

Вуглекислотні вогнегасники можуть бути застосовані для гасіння рідких та твердих речовин, що не можуть горіти без доступу повітря, електроустановок під напругою до 1000 В. У вогнегасник у рідкому стані закачана вуглекислота, що знаходиться там під тиском 6 – 7 МПа. При відкритті вентиля балона, газ одразу приймає вигляд снігоподобної маси і у такому вигляді викидається з дифузора. Час роботі такого вогнегасника 25 – 40 секунд, довжина струменя 1.5 – 3 метри. Для використання вогнегасника слід зірвати пломбу, висмикнути запобіжник, спрямувати розтруб на осередок пожежі та натиснути важіль, спрямовуючи струмінь «снігу» в осередок пожежі.

Під час дії вогнегасника суворо заборонено торкатися розтрубу, який у цей час набуває критично низьких температур, та може спричинити обмороження кінцівки.

Обравши вуглекислотний вогнегасник також слід зауважити, що він буде зменшувати кількість кисню у повітрі під час використання. Для офісу програміста підійде вуглекислотний вогнегасник ВВК-2, маса заряду якого становить 2 кг.

Приміщення 50 $м^2$ під час пожежі класу В, потребує 7 вогнегасників масою заряду вогнегасної речовини 2 кг.

Під час аналізу робочого міста програміста, були надані рекомендації щодо усунення найбільш поширених шкідливих факторів. Було надано аналіз пожежної безпеки кабінету програміста, обраний вид вогнегасної речовини і оптимальна кількість вогнегасників.

ВИСНОВКИ

В роботі було проведено дослідження існуючих програм і сервісів, що мають схожий із моїм задумом функціонал. Було зазначено їх переваги і недоліки, зроблені висновки, які були використані під час розробки.

Було розглянуто системи парольного захисту, їх переваги і недоліки, способи їх зламу. Також було розглянуто алгоритми симетричного шифрування, приділивши увагу алгоритму AES.

Був розроблений алгоритм роботи файлового сейфу, із забезпеченням хмарного зберігання даних та можливістю організації подвійного дна.

В результаті роботи алгоритм біло імплементовано засобами мови пайтон. Готовий програмний продукт на 100% діє за зазначеним алгоритмом. Програмний продукт було протестовано.

Готовий програмний продукт є як ніколи актуальний саме зараз, під час військового стану, легким у використанні, а графічний інтерфейс інтуїтивно зрозумілий.

ПЕРЕЛІК ПОСИЛАНЬ

1. Баричев С.Г, Гончаров В.В., Серов Р.Е. Основы современной криптографии. 2001. С 3 – 4.
2. Криптографія. *Матеріал з вікіпедії – вільної енциклопедії.* URL:https://uk.wikipedia.org/wiki/Криптографія
3. Knight G., Encrypt data using VeraCrypt. 2017. 5 с.
4. Google Drive. *Матеріал з вікіпедії – вільної енциклопедії.* URL:https://uk.wikipedia.org/wiki/Google_Drive
5. Снегуров А.В., Чакрян В.Х. Анализ устойчивости к взлому современных механизмов парольной защиты операционных систем. 2011.
6. Безмалий В.Ф. Парольний захист: справжнє, минуле, майбутнє. 2006. С 1 – 2.
7. Хеш-функція. *Матеріал з вікіпедії – вільної енциклопедії.* URL:https://uk.wikipedia.org/wiki/Хеш-функція
8. Жданова Ю.Д., Спасетєлєва С.О., Шевченко С.М., Кравчук К.В. Прикладні та методичні аспекти застосування хеш-функцій в інформаційній безпеці. 2020. С 27 – 28.
9. Шифрування з симетричними ключами. *Матеріал з вікіпедії – вільної енциклопедії.* URL:https://uk.wikipedia.org/wiki/Шифрування_з_симетричними_ключами
10. Muttaqin K., Rahmadoni J. Analysis and design of file security system AES (advanced encryption standard) cryptography based. *Journal of Applied Engineering and Technological Science (JAETS). 2020*. Т.1. №. 2. С. 113-123.
11. Atikah N. et al. AES-RC4 Encryption Technique to Improve File Security //2019 Fourth International Conference on Informatics and Computing (ICIC). – IEEE, 2019. С. 1-5.
12. Advanced Encryption Standard. *Матеріал з вікіпедії – вільної енциклопедії.* URL:https://uk.wikipedia.org/wiki/Advanced Encryption Standard

13. Dumre R., Dave A. Exploring LSB Steganography Possibilities in RGB Images. *12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, 2021. P. 1-7.
14. Rachael O. Image steganography and steganalysis based on least significant bit (LSB). *Proceedings of ICETIT 2019.* Cham: Springer, 2020. P. 1100-1111.

Додаток А. Лістинг програмного продукту

Файл №1 main.py

```
import interface

if __name__ == '__main__':
    interface.login()
    interface.main()
```

Файл №2 crypto.py

```
import data
import telegram

from re import findall
from random import randint
from hashlib import sha256
from os import remove, replace
from PIL import Image, ImageDraw
from Cryptodome.Cipher import AES

data_password0 = data.password0
data_password1 = data.password1
file_password = data.file_password
api_id = data.api_id
api_hash = data.api_hash


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


def vereficate password(password, dpass0=data password0,
dpass1=data_password1):
    if get_hash(password) == dpass0:
        return 0
    elif get_hash(password) == dpass1:
        return 1
    else:
        return -1
```

```
def crypt(file, mode):
    file_in = open(file, 'rb')
    file_data = file_in.read()
    file_in.close()

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

    file_out = open(f'./TEMP/{file.split("/")[-1]}', "wb")
    [file out.write(x) for x in (cipher.nonce, tag,
ciphertext)]
    file_out.close()

    telegram.send_file(f'./TEMP/{file.split("/")[-1]}',
mode)
    remove(f'./TEMP/{file.split("/")[-1]}')


def decrypt(file patch, file name):
    key = file_password.encode('UTF-8')

    file_in = open(f'./TEMP/{file_name}', "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(f'{file_patch}/{file_name}', 'wb')
    file_out.write(file_data)
    file_out.close()



def stega_encrypt(path_to_image, message, key_patch):
    keys = []
    img = Image.open(path_to_image)
    draw = ImageDraw.Draw(img)
    width = img.size[0]
    height = img.size[1]
    pix = img.load()
    with open(f'{key_patch}/{path_to_image.split("/")[-
1].split(".")[0]}.key', 'w') as f:
```

```
        for elem in ([ord(elem) for elem in message]):
            key = (randint(1, width - 10), randint(1,
height - 10))
            g, b = pix[key][1:3]
            draw.point(key, (elem, g, b))
            f.write(str(key) + '\n')
    img.save(f'./TEMP/{path_to_image.split("/")[-
1].replace("jpg", "")}png', "PNG")


def stega_decrypt(patch_im, patch_keys):
    a = []
    keys = []
    img = Image.open(patch_im)
    pix = img.load()
    f = open(patch_keys, 'r')
    y = str([line.strip() for line in f])

    for i in range(len(findall(r'\((\d+)\,', y))):
        keys.append((int(findall(r'\((\d+)\,', y)[i]),
int(findall(r'\,\s(\d+)\)', y)[i])))
    for key in keys:
        a.append(pix[tuple(key)][0])
    return ''.join([chr(elem) for elem in a])
```

Файл №3 data.py

```
password0 =
'5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d
'
password1 =
'0b14d501a594442a01c6859541bcb3e8164d183d32937b851835442f69
'
file_password = ''
api_id =
api_hash = ""
```

Файл №4 interface.py

```
from tkinter import *
from tkinter import messagebox
from tkinter import filedialog
```

```
import telegram
import crypto


mode = -1
image_expansion = ['png', 'jpg'] # в какие расширения можно
запихнуть сообщения


def login():
    def accept_login(password):
        global mode
        mode = crypto.vereficate password(password)
        if mode == 0 or mode == 1:
            login_page.destroy()
        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()


def main():
    global mode

    def set_listbox():
        listbox.delete(0, END)
        for file in telegram.search_message(mode):
            listbox.insert(END, file['file_name'])
```

```
    def delete_message():
        need_file = listbox.get(listbox.curselection())
        for file in telegram.search_message(mode):
            if need_file == file['file_name']:
                telegram.delete_message(file['message_id'])
                break

    def download_file():
        def image_param(file_name):
            def work(decrypt, message):
                if decrypt:
                    crypto.decrypt('./TEMP/', file name)
                    print(1)
                if message:
                    messagebox.showinfo(title='MESSAGE',
message=crypto.stega_decrypt(f'./TEMP/{file_name}',
filedialog.askopenfilename(title='patch to keys')))
                path = filedialog.askdirectory() + "/"
+file name
                crypto.replace(f'./TEMP/{need_file}', path)
                question.destroy()



            question = Tk()

            Decrypt = BooleanVar(question)
            Message = BooleanVar(question)

            Checkbutton(question, text='Message in pic',
variable=Message, onvalue=True, offvalue=False).grid(row=0,
column=1)
            Checkbutton(question, text='Decrypt',
variable=Decrypt, onvalue=True, offvalue=False).grid(row=0,
column=0)
            Button(question, text='DOWNLOAD', width=15,
command=lambda: work(Decrypt.get(),
Message.get())).grid(row=3, column=0, columnspan=2)

            question.mainloop()


        need_file = listbox.get(listbox.curselection())
```

```
        for file in telegram.search_message(mode):
            if need_file == file['file_name']:
                print(file)
                if need_file.split('.')[-1] == 'png':
                    telegram.download_file(file['file_id'],
'./TEMP', need_file, deccrypt=False)
                    image_param(need_file)

                else:
                    telegram.download_file(file['file_id'],
filedialog.askdirectory(), need_file)
                    crypto.remove(f'./TEMP/{need_file}')
                break


    def send_file(): # отправка файла
        def image_param(patch_to_file, mode): # меню если
изображения
            def work(encrypt, message): # варианты с
обработкой фотки
                if message == '' and encrypt:
                    crypto.crypt(path to file, mode)
                elif message == '' and encrypt == False:
                    telegram.send_file(patch_to_file, mode)
                elif message != '' and encrypt:
                    crypto.stega_encrypt(patch_to_file,
message,

filedialog.askdirectory())

crypto.crypt(f'./TEMP/{patch_to_file.split("/")[-
1].replace("jpg", "png")}', mode)
                elif message != '' and encrypt == False:
                    crypto.stega_encrypt(patch_to_file,
message,

filedialog.askdirectory())

telegram.send_file(f'./TEMP/{patch_to_file.split("/")[-
1].replace("jpg", "png")}', mode)


                question.destroy()
                set_listbox()
```

```
            question = Tk()

            question.title(patch_to_file.split('/')[-1])

            Encrypt = BooleanVar(question)
            Message = StringVar(question)

            Checkbutton(question, text='Encrypt',
variable=Encrypt, onvalue=True, offvalue=False).grid(row=0,
column=0)

            Label(question, text='Message:').grid(row=1,
column=0)
            Entry(question,
textvariable=Message).grid(row=2, column=0, columnspan=2)

            Button(question, text='SEND', width=15,
command=lambda: work(Encrypt.get(),
Message.get())).grid(row=3, column=0, columnspan=2)

            question.mainloop()

        path_to_file = filedialog.askopenfilename()
        if path_to_file.split('.')[-1] in image_expansion:
            image_param(path_to_file, mode)
        else:
            crypto.crypt(path_to_file, mode)

    root = Tk()

    root.title('Vault')

    listbox = Listbox(root, selectmode=SINGLE, width=35)
    listbox.grid(row=0, column=0, columnspan=2)

    Button(root, text='REFRESH', width=30, command=lambda:
set_listbox()).grid(row=1, column=0)
    Button(root, text='UPLOAD', width=30, command=lambda:
send_file()).grid(row=2, column=0)
    Button(root, text='DOWNLOAD', width=30, command=lambda:
download_file()).grid(row=3, column=0)
    Button(root, text='DELETE', width=30, command=lambda:
[delete_message(), set_listbox()]).grid(row=4, column=0)
    Button(root, text='EXIT', width=30, command=lambda:
root.destroy()).grid(row=5, column=0)
```

```
    set_listbox()
    root.mainloop()
```

Файл №5 telegram.py

```
from pyrogram import Client
import crypto
import data


def send_file(file, mode):
    with Client("my_account", data.api_id, data.api_hash)
as app:
        app.send_document("me", file, caption=f'#{mode}')


def search_message(mode):
    files = []
    with Client("my_account", data.api_id, data.api_hash)
as app:
        for message in app.search_messages("me",
query=f"#{mode}"):
            files.append({'caption':
message.caption.split(' ', 1)[-1], 'message_id':
message.message_id,
                          'file_id':
message.document.file_id,
                          'file_name':
message.document.file_name})
        print(files)
        return files


def download_file(file_id, file_patch, file_name,
deccrypt=True):
    with Client("my_account", data.api_id, data.api_hash)
as app:
        app.download_media(file_id,
file_name=f'./TEMP/{file_name}')
        if deccrypt:
            crypto.decrypt(file_patch, file_name)
```

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