МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра прикладної математики та моделювання складних систем

Допущено до захисту

Завідувач кафедри ПМ та МСС

_____ доц. Коплик І.В.
(підпис)

«__» _____ 20__ р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр» / «магістр»

спеціальність 113 «Прикладна математика»

освітньо-професійна програма «Прикладна математика»

тема роботи

«Моделювання криптографічної системи з різними шифрами на прикладі комбінації хешування та стеганографії»

Виконавець

студент факультету ЕлІТ

Наталуха Ілля Павлович

Науковий керівник

к.ф. – м.н.

Козлова Ірина Іванівна

Суми – 2020

Зміст

Вступ

Мета об’єднання в одну криптографічну модель стеганографії та хешування - це оптимізація даних користувача, та їх ретельніше приховання.

Стеганографія - наука, про спосіб таємницею передачі повідомлення, тобто вихідний текст залишається незмінним, а ховається сам лист або його вміст. (Наприклад лист, написаний невидимим чорнилом).

В криптографії, **шифрування** це трансформація відкритого тексту в кодований текст або шифротекст. Шифротекст призначений бути нечитабельним для неавторизованих читачів. Шифрування це основа криптографії: воно трансформує повідомлення з текстовими даними в шифротекст використовуючи алгоритм, що називається " шифром ". Шифруваннння з використанням сучасних шифрів виконується за допомогою специфічних алгоритмів і секрету, що має назву ключ. Так як алгоритми досить часто є в публічному доступі, то ключ має залишатись таємним, якщо шифрування має бути захищеним. Як працює шифрування. Відкритий текст (Cleartext) проходить процес шифрування(Encrypt) з використанням шифру (Cipher) й на виході маємо шифротекст(Ciphertext). В нашому випадку Cleartext є не символи ,а зображення, а саме: шифр кожного кольору пікселя в зображенні. Без знання секрету, зворотня операція дешифрування, є складною для виконання з математичної точки зору. Складність залежить від рівня захищеності обраного криптографічного алгоритму і розвивається з прогресом криптоаналізу.[1]

Хеш-функції – це функції, призначені для «стиснення» довільного повідомлення або набору даних, записаних, як правило, в двійковому алфавіті, в певну бітову комбінацію фіксованої довжини, яка називається згорткою. Хеш-функції мають різноманітні застосування при проведенні статистичних експериментів, при тестуванні логічних пристроїв, при побудові алгоритмів швидкого пошуку і перевірки цілісності записів в базах даних. Основною вимогою

до хеш-функцій є рівномірність розподілу їх значень при випадковому виборі значень аргументу.[8]

Криптографічного хеш-функцією називається всяка хеш-функція, яка є криптостійкою, тобто задовольняє ряду вимог специфічних для криптографічних додатків. У криптографії хеш-функції застосовуються для вирішення наступних завдань:

— побудови систем контролю цілісності даних при їх передачі або зберігання,

— аутентифікація джерела даних.

Мета об’єднання в одну криптографічну модель стеганографії та хешування це оптимізація даних користувача, та їх ретельне приховання. [2]

1.Розвиток стеганографії та хешування

Для захисту інформації стали використовуватися більш ефективні методи кодування і криптографії. Від криптографії стеганографія відрізняється тим, що з допомогою криптографії можна приховати зміст повідомлення, а, користуючись стеганографією, можна сховати саме існування повідомлення.

Комп'ютерні технології надали новий імпульс розвитку та вдосконалення стеганографії, з'явився новий напрямок в області захисту інформації - комп'ютерна стеганографія.

Комп'ютерна стеганографія – це приховування повідомлення або файлу в іншому повідомленні або файлі. Наприклад, стеганографи можуть сховати аудіо - або відеофайл в іншому інформаційному або навіть у великому графічному файлі. Боротьба з тероризмом і переслідування винних у скоєнні теракту 11 вересня 2001 року привернули особливу увагу до стеганографії. Деякі фахівці вважають, що терористична організація «Аль-Каїда» використовувала Інтернет для розробки плану нападів, а стеганографія допомогла зберегти в таємниці їх злочинні наміри.

Процес стеганографії можна розділити на кілька етапів. [3]

Вибір інформаційного файлу.

Першим етапом у процесі стеганографії є вибір файлу, який потрібно приховати. Його ще називають інформаційним файлом.

Вибір файлу-контейнера.

Другим етапом у процесі стеганографії є вибір файлу, який використовується для приховування інформації. Його ще називають файлом-контейнером. У більшості відомих програм стеганографії говориться, що для приховування інформації, обсяг пам'яті файлу-контейнера повинен десь у вісім разів перевищувати обсяг пам'яті інформаційного файлу. Отже, щоб заховати файл розміром 710КБ, знадобиться графіка об'ємом 5600КБ.[3]

Вибір стеганографічної програми.

Третім етапом у процесі стеганографії є вибір стеганографічної програми.

Один з кращих і найбільш поширених продуктів у цій області для платформи Windows95/NT - це S-Tools (має статус freeware). Програма дозволяє приховувати будь-які файли в gif і bmp, так і в аудіофайлі формату wav. При цьому S-Tools - це стеганографія і криптографія "в одному флаконі", тому що файл, що підлягає приховування, ще й шифрується за допомогою одного з криптографічних алгоритмів з симетричним ключем: DES (часів якого пройшли), потрійний DES або IDEA - два останніх на сьогодні цілком заслуговують довіри .

1.2. Практичне застосування стеганографії

Непомітна передача інформації

Найочевидніше, що перше приходить на розум. На відміну від криптографічних методів (які таємниці, але не потайливі), стеганографія може застосовуватися як метод непомітної передачі інформації. Це становить «класичне практичне застосування» стеганографії, тому дана мета — на першому місці. [4]

Приховане зберігання інформації

Дана мета стеганографії багато в чому схожа на попередню. Тільки в даному випадку стеганографія використовується не для передачі, а для зберігання будь-якої інформації, виявлення самого факту наявності якої (нехай хоч навіть у зашифрованому вигляді) користувачу небажано. Очевидно, що дана задача реалізована на носіях даних, але не в каналах зв'язку. Причому надмірність на багатьох носіях може бути неймовірно великий. Наприклад, загальний обсяг даних (з урахуванням кодів RLL), які можна записати на CD диск складають 1828 Мб даних. Це величезна надмірність, яку можна використовувати для приховування даних. [4]

Повсякденне зберігання інформації

Багато інформаційні ресурси дозволяють зберігати дані тільки визначеного виду. Наприклад портал YouTube дозволяє зберігати тільки відеоінформацію в форматах MOV, MPEG4, AVI, WMV, MPEG-PS, FLV, 3GPP, WebM. Однак можна використовувати стеганографію для зберігання даних в інших форматах. [5]

1.3 Технологія хешування

Технологічний розвиток вплинув на те, що на сьогоднішній день поширене хешування для ідентифікації файлів. Адже хеш є унікальним номером, дублювання якого виключається згідно алгоритму. Також, хешування використовується для підтвердження достовірності файлу.

Якщо в хешованом файлі відбувається зміна, його хеш автоматично змінюється. Кожен наступний хеш зав'язаний до попереднього хешу, чим забезпечується зв'язність всіх блоків. Тому, процес хешування так необхідний в блокчейні, забезпечуючи унікальність і неповторність кожного елемента системи. Крім того, для повної безпеки, всі транзакції блокчейна функціонують тільки з цифровими підписами. Ці підписи забезпечують перевірку того, що транзакцію відправив саме відправник.

Як працює хешування в блокчейні:

Згідно з алгоритмом, перший хеш обчислюється для першого блоку або блоку Genesis, всередині якого знаходяться транзакції від запуску мережі до моменту знаходження хеш. Операції записуються послідовно в кожному блоці. Хеш і транзакції попереднього блоку використовуються як вхідні дані для визначення хешу нового блоку. Так формується ланцюжок, в якій блоки розташовуються послідовно, згідно з номером хеш-коду.

Цей механізм гарантує автентичність кожної транзакції. У разі зміни історія окремої транзакції, відбувається зміна хеш блоку, а в результаті нове хешування всіх наступних транзакцій і блоків. Тому, хеши використовуються для відображення поточного стану мережі.

Пошуком хеш займаються повні вузли, валідатори або майнери, в залежності від алгоритму. Деякі алгоритми передбачають хешування транзакцій, крім хешування блоків валідаторами. Існують так само алгоритми, які функціонують тільки за допомогою потужного обчислювального обладнання.

Як ми бачимо хешування найкращий спосіб передачі інформації користувача, також хешування доводиться головною і невід'ємною частиною роботи блокчейн мереж. Які нещодавно були настільки популярні.

На сьогодні існує величезна кількість різних алгоритмів хешування, які відрізняються тільки способом обробки даних.

2.Основні визначення стеганографії

Деякі помилково вважають, що стеганографія є замінником криптографії. Це не заміна, а додаток, який захищає інформацію. Інформація, прихована за допомогою стенографії має менше шансів на виявлення факту передачі змісту повідомлення. А шифрування повідомлення забезпечує додатковий захист.

Стеганографія надає неоціненну послугу в тому випадку, коли не тільки треба передати засекречену інформацію, але і зробити це так, щоб про цю передачу ніхто не знав.

Існує два способи кодування повідомлень – шифрування і стенографія. Як же можна заховати передану інформацію? Існує тільки одна відповідь: тільки в інформації ще більшого розміру. Принцип роботи стенографії полягає в тому, щоб розкидати секретний текст в основному масиві повідомлення, яке може бути навіть відмінно за змістом. При цьому витягти його буде можливо, тільки знаючи принцип, за яким була проведена розбивка і розсіювання.

Існують два головних принципи, на яких базується комп'ютерна стенографія:

-при оцифрування звуку або зображення файли можуть бути легко змінені, але при цьому їх функціональність зберігається;

-людина не здатний відрізнити найдрібніші зміни в зображення або звук.

Наприклад, хто-то вам відправив закодоване повідомлення. Ви відкриваєте свої ноутбуки, планшети, електронні книги і бачите текст листа з певним зображенням. Але якщо пропустити його через фільтр, який виділить кожен біт в коді яскравості і перетворить його в те саме таємне послання.

Отже, стенографічна система (або, як її ще називають, стегосистема) це об'єднання методів і засобів, за допомогою яких здійснюється створення таємного каналу для пересилання інформації. При її створенні не варто забувати про те, що:

-противник прекрасно обізнаний про існування стеганографічної системи. Єдине, що залишається для нього таємницею – це ключ до розшифровки, який дає

можливість встановити факт знаходження шифровки в тексті, а також розкодувати її.

-потрібно створити всі умови для того, щоб противник не мав можливості у розпізнаванні і розшифровці таємних послань.

Для створення стегосистеми можливо використовувати будь-який вид інформації (відео або фотозображення, текст і так далі). Комп'ютерна стеганографії розрізняє два тифу файлів: файл-контейнер, який служить для приховування безпосередньо файла-повідомлення. Іншими словами лист-секрет монтується в лист-контейнер.

2.1 Методи та способи шифрування за допомогою стеганографії:

На сьогодні розроблені і випробувані різні способи і методи стеганографії, ми відзначимо наступні:

1. LSB-стеганографія (повідомлення ховається в молодших бітах (можливе використання одного або кількох молодших біт) контейнера. Чим менше біт задіяно, тим менше артефактів отримує оригінальний контейнер після впровадження.

2. Метод, оснований на приховуванні даних в коефіцієнтах дискретного косинусного перетворення (далі-ДКП) — різновид попереднього методу, яка активно використовується, наприклад, при впровадженні повідомлення в контейнер формату JPEG. При інших рівних, такий контейнер має дещо меншу ємність ніж у попередньому методі, в тому числі за рахунок того, що коефіцієнти «0» і «1» залишаються незмінними — впровадження повідомлення у них неможливо.

3. Метод приховування інформації за допомогою молодших біт палітр — цей метод по суті є варіантом загального методу LSB, але інформація вбудовується не в найменш значущі біти контейнера, а в найменш значущі біти палітри, очевидний недолік такого методу — низька ємність контейнера.

4. Метод приховування інформації у службових полях формату — досить простий метод, заснований на використанні службових полів заголовка контейнера

для зберігання повідомлення. Очевидні мінуси — низька ємність контейнера і можливість виявлення впроваджених даних за допомогою звичайних програм для перегляду зображення (які іноді дозволяють бачити вміст службових полів).

5. Метод вбудовування повідомлення — полягає в тому, що повідомлення вбудовується в контейнер, потім за допомогою схеми, відомої обом сторонам, витягується. Можна вбудувати кілька повідомлень в один контейнер, за умови, що способи їх впровадження ортогональні.

6. Дуже багато формати можуть зберігати деякі метадані. Плюсом цього методу є те, що він не порушує формат файлу, тому робота з цими метаданими зазвичай добре документована, і бібліотеки вже готові, що дозволяє вам швидко написати програму для зберігання ваших даних у цих файлах. Майже всі медіаформати мають підтримку метаданих. Однак дані не завжди можуть зберігатися там, щоб їх не можна було побачити. Отже, де ви можете спробувати зберегти секретні дані:

7. Широкосмугові методи, які поділяються на:

1) метод псевдовипадковою послідовності; використовується секретний сигнал, який моделюється псевдовипадкових сигналом.

2) метод стрибаючих частот: частота несучого сигналу змінюється за певним псевдовипадковим законом.

8. Метод накладання — по суті не є справжньою стеганографією, заснований на тому, що деякі формати містять в заголовку розмір даних, або ж обробник цих форматів буде читати файл до маркера кінця даних. Прикладом такого методу є добре відомий метод «rar-jpeg», який заснований на конкатенації графічного файлу в форматі JREG і RAR-архіву. Для перегляду JPEG буде зчитувати інформацію до межі, зазначеної в заголовку файлу, а RAR-архіватор відкине все, що знаходиться до сигнатури «RAR!», яка позначає початок архіву. Таким чином, якщо такий файл відкрити у вікні перегляду графічних файлів — ми побачимо картинку, а якщо в RAR-архіваторі — вміст RAR-архіву. Очевидні мінуси такого підходу полягають у

тому, що оверлей, доданий до контейнера, легко виділяємо при візуальному дослідженні такого файлу.

9. Тільки нещодавно з'явилося повідомлення: "ми приховуємо текст у форматі MP3, де описана реалізація зберігання інформації PHP в тезі ID3v1. Але справа в тому, що тег ID3v1 дуже обмежений і багато інформації там не зберігається. Крім того, всі дані добре видно в будь звичайної медіа-програмі. Зовсім інша справа - тег ID3v2.4, який дозволяє зберігати дані набагато більших розмірів, а також дозволяє зберігати нестандартні дані. Наприклад, деякі програми зберігають налаштування гучності і нормалізації для кожного файлу. Медіаплеєри зазвичай не відображають невідомі параметри.

Варто згадати і маловідомий тег Lyrics3, який був створений для збереження у текстовому файлі пісні, а також розширення тегів ID3v1 (наприклад, дозволяє записати більш довге ім'я пісні), але висновок тегів ID3v2 і не дав тегу Lyrics3 широкого поширення. Але дивно, що велика кількість файлів MP3, які тепер можна знайти в Інтернеті, містять цей тег (хоча виключення з назви пісні там не зберігається).

Ще один спосіб зберігати конфіденційні дані у файлах MP3. Це реалізовано в програмі MP3Stego. На жаль, автору цієї програми не був створений проект 2006 року. Ідея полягає в тому, що дані спочатку шифруються, а потім лежать в основі процесу кодування файлу MP3 з WAV) до кінцевого результату. У підсумку виходить звичайний MP3-файл, без яких-небудь помітних спотворень, але він зберігає закодовані дані.

10. Мало хто знає, що файли AVI також підтримують метадані, і багато речей можуть бути збережені. Як і у форматах MP3 і JPEG, ви можете створити деякі з ваших ключів, які будуть просто ігнорувати роботу додатків з метаданими.[1]

3. Основні поняття хешування

В даний час практично жодне додаток криптографії не обходиться без використання хешування.

Хеш-функцією називається всяка функція h:X -> Y, легко вироховувана і така, що для будь-якого повідомлення M значення h(M) = H (згортка) має фіксовану бітову довжину. X — множина всіх повідомлень, Y — множина двійкових векторів фіксованої довжини. Як правило хеш-функції будують на основі так званих однокрокових стискаючих функцій y = f(x1, x2) двох змінних, де x1, x2 і y — двійкові вектори довжини m, n і n відповідно, причому n — довжина згортки, а m — довжина блоку повідомлення. Для отримання значення h(M) повідомлення спочатку розбивається на блоки довжини m (при цьому, якщо довжина повідомлення не кратна m то останній блок якимось спеціальним чином доповнюється до повного), а потім до отриманих блокам M1, M2,.., MN застосовують таку послідовну процедуру обчислення згортки:

Ho = v,

Hi = f(Mi,Hi-1), i = 1,.., N,

h(M) = HN

Тут v — деяка константа, часто її називають ініціалізованим вектором.

Вона вибирається з різних міркувань і може являти собою секретну константу або набір випадкових даних (вибірку дати і часу, наприклад). При такому підході властивості хеш-функції повністю визначаються властивостями однокроковою стискаючої функції. [6]

3.1 **Види хеш-функцій**

Виділяють два важливих види криптографічних хеш-функцій — ключові і безключеві. Ключові хеш-функції називають кодами аутентифікації повідомлень. Вони дають можливість без додаткових коштів гарантувати правильність джерела даних, так і цілісність даних в системах з довіряють одне одному користувачами.

Безключеві хеш-функції називаються кодами виявлення помилок. Вони дають можливість за допомогою додаткових засобів (шифрування, наприклад) гарантувати цілісність даних. Ці хеш-функції можуть застосовуватися в системах як з довіряють, так і не довіряють одне одному користувачами. [7]

Про статистичні властивості та вимоги

Основною вимогою до хеш-функцій є рівномірність розподілу їх значень при випадковому виборі значень аргументу. Для криптографічних хеш-функцій також важливо, щоб при найменшій зміні аргументу значення функції сильно змінювалося. Це називається лавинним ефектом.

До ключових функцій хешування пред'являються наступні вимоги:

— неможливість фабрикації,

— неможливість модифікації.

Перша вимога означає високу складність підбору повідомлення з правильним значенням згортки. Друге — високу складність підбору для заданого повідомлення з відомим значенням згортки іншого повідомлення з правильним значенням згортки.

До безключовим функцій пред'являють вимоги:

— односпрямованість,

— стійкість до колізій,

— стійкість до знаходження другого прообразу.

Під односпрямованістю розуміють високу складність знаходження повідомлення по заданому значенню згортки. Слід зауважити, що на даний момент немає використовуваних хеш-функцій з доведеною односпрямованість. [8]

Під стійкістю до колізій розуміють складність знаходження пари повідомлень з однаковими значеннями згортки. Зазвичай саме знаходження способу побудови колізій криптоаналітиками служить першим сигналом старіння алгоритму і необхідність швидкої заміни.

Під стійкістю до знаходження другого прообразу розуміють складність знаходження другого повідомлення з тим же значенням згортки для заданого повідомлення з відомим значенням згортки.

Популярні хеш-алгоритми

Алгоритми CRC16/32 — контрольна сума (не криптографічне перетворення).

Алгоритми MD2/4/5/6. Є творінням Рона Райвеста, одного з авторів алгоритму RSA.

Алгоритм MD5 мав колись велику популярність, але перші передумови злому з'явилися ще в кінці дев'яностих, і зараз його популярність стрімко падає.

Алгоритм MD6 — дуже цікавий з конструктивної точки зору алгоритм. Він висувався на конкурс SHA-3, але, на жаль, автори не встигли довести його до кондиції, і в списку кандидатів, які пройшли у другий раунд цей алгоритм відсутня.

Алгоритми лінійки SHA Широко поширені зараз алгоритми. Йде активний перехід від SHA-1 до стандартів версії SHA-2. SHA-2 — збірна назва алгоритмів SHA224, SHA256, SHA384 і SHA512. SHA224 і SHA384 є по суті аналогами SHA256 і SHA512 відповідно, тільки після розрахунку згортки частина інформації в ній відкидається. Використовувати їх слід лише для забезпечення сумісності з устаткуванням старих моделей. [6]

4. Практична частина

Хешування для подальшого кодування в зображення

Створюємо масив простих чисел для подальшого хешування

```
import random
arr = input()
arr = list(arr.upper())
#----------масив простих чисел------
n = 500
lst=[2]
for i in range(3, n+1, 2):
  if (i > 10) and (i%10==5):
    continue
  for j in lst:
    if j*j-1 > i:
      lst.append(i)
      break
    if (i % j == 0):
      break
  else:
    lst.append(i)
#print (lst)
```

Сам процес хешування, де за основу береться p і q, з випадковими числами

```
h0 = h1 = 100
p = random.choice(lst)
q = random.choice(lst)
r = p*q

if p == q:
```

```
    q = random.choice(lst)

r = p*q

alphabet = 'QWERTYUIOPASDFGHJKLZXCVBNMАБВГҐДЕЄЖЗДИІЙКЛМНОПРСТУФХЦЧШЩЬЮЯ '
alphabet = list(alphabet)
```

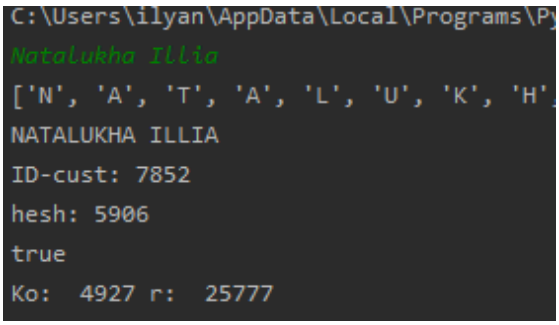
Принцип розрахунку такий же як і для цифрового підпису, але сам результат ми будемо використовувати для прив’язування не до підпису, а до фото користувача, назвемо це ID-користувача.

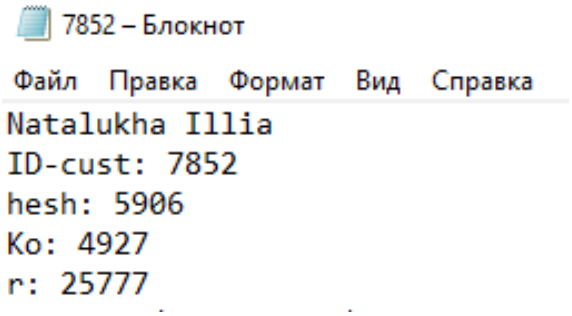
```
#----------ID--------
Kc = random.choice(lst)
delt = (p-1)*(q-1)
if delt%Kc == 0:
    Kc = random.choice(lst)
y = 1
Ko = abs((delt*y +1))%Kc
while Ko != 0:
    y += 1
    Ko = abs((delt * y+1)) % Kc
Ko = int(abs((delt * y+1))/ Kc)

for i in range(len(arr)):
    o = ord(arr[i])
    for j in range(len(alphabet)):
        l = ord(alphabet[j])
        if o == l:
            h0 = ((h0 + j + 1)**2) % r #Вироховування хеш-функції
            h1 = ((h1 + j + 1)**2) % r
            break
print(arr)
```

```
S = (h0**Kc)%r
print('check = (ID-cust^Ko)/r: %d' %(s1))
print('hesh: %d' %(h1))
if s1 == h1:
    print('true')
else: print('false')
print('Ko: ', Ko, 'r: ', r)
with open(f'{S}.txt',"w") as f:
    f.write(sss)
    f.write('\n')
    f.write("ID-cust: ")
    f.write(str(S))
    f.write('\n')
    f.write("hesh: ")
    f.write(str(h1))
    f.write("\n")
    f.write("Ko: ")
    f.write(str(Ko))
    f.write("\n")
    f.write("r: ")
    f.write(str(r))
    f.write("\n")
    f.write("check = (ID-cust^Ko)/r: ")
    f.write(str(s1))
    f.write("\n")
```

Результат:

Ми зберегли всю вихідну інформацію в файл, для кожного нового користувача генерується новий.





Залишилась тільки перевірка данних, це ми зробимо після того як зашифруемо та розшифруемо наш ID в фото.

4.1 Застосування методу компьютерної стеганографії

Отже, у нас є зображення. В зображенні є пікселі. Пікселі утворені з основних кольорів — червоного, зеленого і синього. Кожен з кольорів закодований числом від 0 до 255.

Color Chart	R	G	B	Color Name
	0	0	0	Black
	255	255	255	White
	224	224	224	Light Gray
	128	128	128	Gray
	64	64	64	Dark Gray
	255	0	0	Red
	255	96	208	Pink
	160	32	255	Purple
	80	208	255	Light Blue
	0	32	255	Blue
	96	255	128	Yellow-Green
	0	192	0	Green
	255	224	32	Yellow
	255	160	16	Orange
	160	128	96	Brown
	255	208	160	Pale Pink

Існують ASCII символи, які закодовані також.

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	*[NULL]*
1	1	1	1	*[START OF HEADING]*
2	2	10	2	*[START OF TEXT]*
3	3	11	3	*[END OF TEXT]*
4	4	100	4	*[END OF TRANSMISSION]*
5	5	101	5	*[ENQUIRY]*
6	6	110	6	*[ACKNOWLEDGE]*
7	7	111	7	*[BELL]*
8	8	1000	10	*[BACKSPACE]*
9	9	1001	11	*[HORIZONTAL TAB]*
10	A	1010	12	*[LINE FEED]*
11	B	1011	13	*[VERTICAL TAB]*
12	C	1100	14	*[FORM FEED]*
13	D	1101	15	*[CARRIAGE RETURN]*
14	E	1110	16	*[SHIFT OUT]*
15	F	1111	17	*[SHIFT IN]*
16	10	10000	20	*[DATA LINK ESCAPE]*
17	11	10001	21	*[DEVICE CONTROL 1]*
18	12	10010	22	*[DEVICE CONTROL 2]*
19	13	10011	23	*[DEVICE CONTROL 3]*
20	14	10100	24	*[DEVICE CONTROL 4]*
21	15	10101	25	*[NEGATIVE ACKNOWLEDGE]*
22	16	10110	26	*[SYNCHRONOUS IDLE]*
23	17	10111	27	*[ENG OF TRANS. BLOCK]*
24	18	11000	30	*[CANCEL]*
25	19	11001	31	*[END OF MEDIUM]*
26	1A	11010	32	*[SUBSTITUTE]*
27	1B	11011	33	*[ESCAPE]*
28	1C	11100	34	*[FILE SEPARATOR]*
29	1D	11101	35	*[GROUP SEPARATOR]*
30	1E	11110	36	*[RECORD SEPARATOR]*
31	1F	11111	37	*[UNIT SEPARATOR]*
32	20	100000	40	*[SPACE]*
33	21	100001	41	**!**
34	22	100010	42	**"**
35	23	100011	43	**#**
36	24	100100	44	**$**
37	25	100101	45	**%**
38	26	100110	46	**&**
39	27	100111	47	**'**
40	28	101000	50	**(**
41	29	101001	51	**)**
42	2A	101010	52	*****
43	2B	101011	53	**+**
44	2C	101100	54	**,**
45	2D	101101	55	**-**
46	2E	101110	56	**.**
47	2F	101111	57	**/**
48	30	110000	60	**0**
49	31	110001	61	**1**
50	32	110010	62	**2**
51	33	110011	63	**3**
52	34	110100	64	**4**
53	35	110101	65	**5**
54	36	110110	66	**6**
55	37	110111	67	**7**
56	38	111000	70	**8**
57	39	111001	71	**9**
58	3A	111010	72	**:**
59	3B	111011	73	**;**
60	3C	111100	74	**<**
61	3D	111101	75	**=**
62	3E	111110	76	**>**
63	3F	111111	77	**?**
64	40	1000000	100	**@**
65	41	1000001	101	**A**
66	42	1000010	102	**B**
67	43	1000011	103	**C**
68	44	1000100	104	**D**
69	45	1000101	105	**E**
70	46	1000110	106	**F**
71	47	1000111	107	**G**
72	48	1001000	110	**H**
73	49	1001001	111	**I**
74	4A	1001010	112	**J**
75	4B	1001011	113	**K**
76	4C	1001100	114	**L**
77	4D	1001101	115	**M**
78	4E	1001110	116	**N**
79	4F	1001111	117	**O**
80	50	1010000	120	**P**
81	51	1010001	121	**Q**
82	52	1010010	122	**R**
83	53	1010011	123	**S**
84	54	1010100	124	**T**
85	55	1010101	125	**U**
86	56	1010110	126	**V**
87	57	1010111	127	**W**
88	58	1011000	130	**X**
89	59	1011001	131	**Y**
90	5A	1011010	132	**Z**
91	5B	1011011	133	**[**
92	5C	1011100	134	****
93	5D	1011101	135	**]**
94	5E	1011110	136	**^**
95	5F	1011111	137	**_**
96	60	1100000	140	**`**
97	61	1100001	141	**a**
98	62	1100010	142	**b**
99	63	1100011	143	**c**
100	64	1100100	144	**d**
101	65	1100101	145	**e**
102	66	1100110	146	**f**
103	67	1100111	147	**g**
104	68	1101000	150	**h**
105	69	1101001	151	**i**
106	6A	1101010	152	**j**
107	6B	1101011	153	**k**
108	6C	1101100	154	**l**
109	6D	1101101	155	**m**
110	6E	1101110	156	**n**
111	6F	1101111	157	**o**
112	70	1110000	160	**p**
113	71	1110001	161	**q**
114	72	1110010	162	**r**
115	73	1110011	163	**s**
116	74	1110100	164	**t**
117	75	1110101	165	**u**
118	76	1110110	166	**v**
119	77	1110111	167	**w**
120	78	1111000	170	**x**
121	79	1111001	171	**y**
122	7A	1111010	172	**z**
123	7B	1111011	173	**{**
124	7C	1111100	174	**|**
125	7D	1111101	175	**}**
126	7E	1111110	176	**~**
127	7F	1111111	177	*[DEL]*

Давайте спробуємо зашифрувати в фото наш ID.

Фото:

Підключимо необхідні бібліотеки.

```
import numpy as np
import scipy.misc
from PIL import Image, ImageDraw
import imageio
```

Потім оголосимо функцію і помістимо в неї всі об'єкти, які нам знадобляться.

```
img = Image.open('new.bmp')
draw = ImageDraw.Draw(img)
data = np.asarray(img, dtype='int32')
width = img.size[0]
height = img.size[1]
pix = img.load()
f = open('keys.txt','w')
```

Найголовніше завдання — придумати спосіб, за допомогою якого стане можливим шифрувати повідомлення. Був запропонований такий спосіб:
Беремо символ, переводимо його в число ASCII, запам’ятовуємо одну єдину цифру, це довжина нашого ID. Також так, як в результаті хешування ID користувачів має різну довжину, ми маємо запам’ятати її і позначимо k.

Принцип кодування є початковою координатою першого шифрованого пікселя, та відступом до іншого шифрованого пікселя.
Кожній цифрі відповідає свій код ASCII, в нашому випадку цей код дублює зелений діапазон в палітрі Red Green Blue (червоний, зелений, синій)

```
red = []
green = []
blue = []
for x in range(len(data)):
    for y in range(len(data)): #створення палітри
        red.append(data[x, y][0])
        green.append(data[x, y][1])
        blue.append(data[x, y][2])
word = S          #наш айді
enc = [ord(i) for i in str(word)]
```

```
print(enc)
dec = [chr(i) for i in enc]
print(dec)
key = width / 10
key = int(key)
l = key * width #додаткова умова захисту, залежить від ширини фото
if (l >= 10 and l < 100):
    l = l / 10
if (l >= 100 and l < 1000):
    l = l / 100
if (l >= 1000 and l < 10000):
    l = l / 1000
if (l >= 10000 and l < 100000):
    l = l / 10000
if (l >= 100000 and l < 1000000):
    l = l / 100000
if (l >= 1000000 and l < 100000000):
    l = l / 1000000
l = int(l)
count = 0
j = 0
k = 0
x = height-key
y = 0
for i in range(len(enc)): #спосіб шифрування за допомогою ключа
    y = y + key
    x = x - 1
    if x <= 1: x = height-key+1
    if y >= width: y = 0
    data[x, y][1] = enc[count] + key
    count += 1
```

```
    print(data[x, y][0], data[x, y][1], data[x, y][2], '\n')
```

Зберігаємо ключі і зображення.

```
#scipy.misc.imsave('resultat2.bmp', data)
imageio.imwrite('result2.bmp', data)
print(len(enc))              #довжина айди
```

Результат:

```
[55, 56, 53, 50]
['7', '8', '5', '2']
7852
```

Можно побачити довжину та ширину картинки, та 2 масиви, перший це наш ID в коді ASCII, та другий масив наші вхідні дані, тобто ID без змін.

Також в результаті ми отримали теж сам зображення з зміненими пікселями, які помітити дуже важко без додаткового наближення.

Збільшене зображення

Початок шифрування починається знизу, з лівої сторони для того, щоб ретельніше приховати шифровані пікселі, тому що зверху завжди фон світліше, будь то небо, стіна, або хромакей

4.2 **Розшифрування**

Так як і планувалося ми отримали вхідні данні, перше – це фото з шифром, друге – це один єдиний ключ 4 – це довжина шифру.

```
import numpy as np
import scipy.misc
import imageio
from PIL import Image, ImageDraw

img = Image.open('result2.bmp')
draw = ImageDraw.Draw(img)
data = np.asarray(img, dtype='int32')
width = img.size[0]
```

```
height = img.size[1]
pix = img.load()
f = open('keys.txt','w')
print(width, height)
red = []
green = []
blue = []
for x in range(len(data)):
    for y in range(len(data)):
        red.append(data[x, y][0])
        green.append(data[x, y][1])
        blue.append(data[x, y][2])
print('Введите ключ - ')
k = int(input()) #ключ, залежить від довжини айді
key = width / 10
key = int(key)
l = key * width #додаткова умова захисту, залежить від ширини фото
if (l >= 10 and l < 100):
    l = l / 10
if (l >= 100 and l < 1000):
    l = l / 100
if (l >= 1000 and l < 10000):
    l = l / 1000
if (l >= 10000 and l < 100000):
    l = l / 10000
if (l >= 100000 and l < 1000000):
    l = l / 100000
if (l >= 1000000 and l < 100000000):
    l = l / 1000000
l = int(l)
text = []
count = 0
```

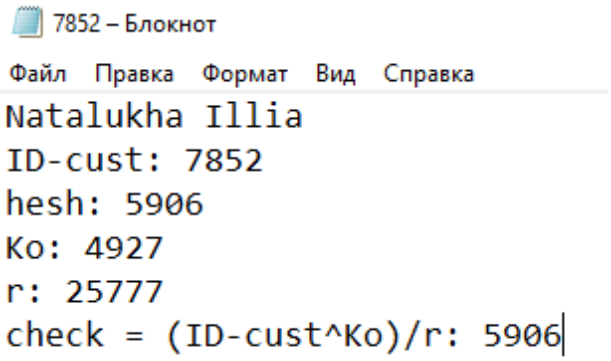
```
j = 0
x = height-key
y = 0
for i in range(k): #алгоритм розшиврування з використанням ключів
    y = y + key
    x = x - 1
    if x <= 1: x = height - key+1
    if y >= width: y = 0
     text.append(data[x, y][1])
    count += 1
    print(count, data[x, y][0], data[x, y][1], data[x, y][2], '\n')
for i in range(k):
    text[i] = text[i] - key

text = list(text)
print(text)
dec = [chr(i) for i in text] #масив в якому зберігається розшифровані данні
dec =''.join(dec)
print(dec)
print(''.join(dec))
#scipy.misc.imsave('resultat2.bmp', data)
imageio.imwrite('result22.bmp', data)
os.startfile(f'{dec}.txt')
```

Результат:

```
1280 1069
Введите ключи -
20
4
1 5 55 5

2 6 56 6

3 6 53 6

4 6 50 6

[55, 56, 53, 50]
['7', '8', '5', '2']
```

На виході бачимо наш ID, за допомогою ID у назві потрібного файлу программа автоматично відкриває потрібний файл на компьютері.

os.startfile(f'{dec}.txt') – це строка яка відповідальна за цей процесс

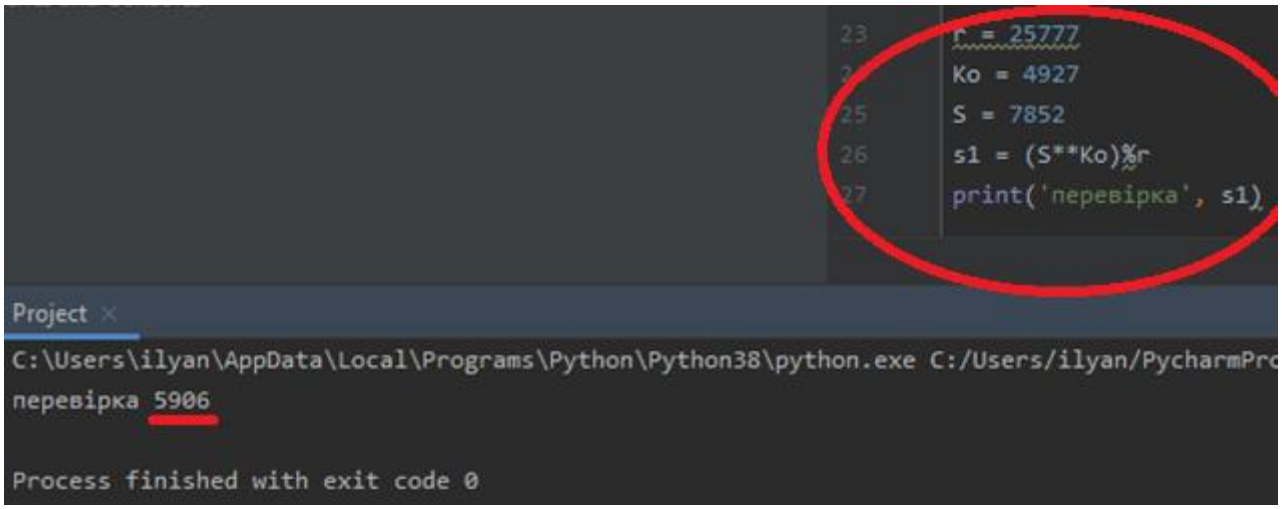


Перевірка виконується автоматично ще в програмі для хешування, але можна виконати її вручну для наглядної демонстрації якості.

Використаємо формулу для перевірки.

$$S_1 = (S^{Ko})/r$$

Де S - це наш ID.

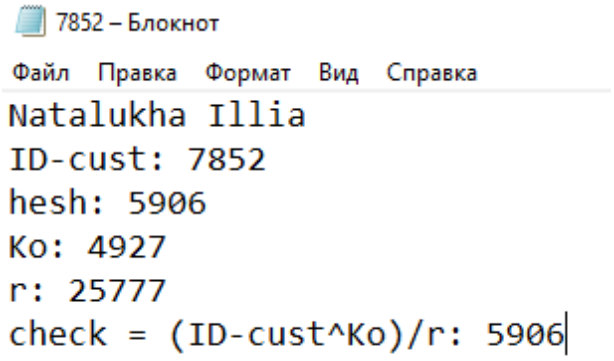


Порівняємо з прив’язаним до нашого айді хешем і робимо висновок, що все зходится, перевірка задовільна

5. Висновки

Отже поєднання хешування та стеганографії дало змогу змоделювати криптографічну систему, яка оптимізує дані які потрібні для індифікації особи до звичайного фото на паспорт.

А після розшифровки фото, отриманий ID напряму зв’язаний з хешем та файлом з даними за ним.

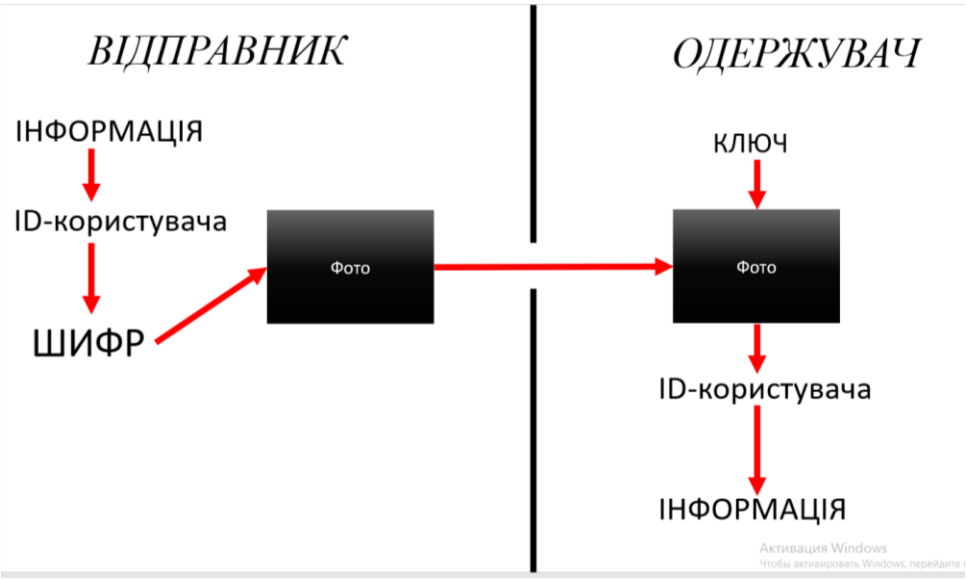


Звичайно є ймовірність того, що користувач допустить втрату ключей або витік даних, але зіткнувшись з звичайною фотографією, зловмисник навіть і не здогадається, що це фото є носієм якоїсь інформації або в гіршому випадку не зможе відтворити розшифрування без потрібного алгоритму який потрібен у випадку із дешифруванням як для стеганографії, так і для більш складнішого алгоритму хешування. Також користувачу непотрібен доступ до носія з даними, це означає, що для правильного користування програмою йому треба мати при собі лише ключ і фото.

Тобто окрім оптимізації даних, за допомогою даної моделі вдалося ще більш захистити користувача в два етапи, та зручно її використовувати.

Використання даної моделі можливе як через інтернет, тобто дистанційна перевірка юридичною особою, яка має доступ до бази даних, або ж це може бути частиною електронного паспорту, який може зберігатися прямо в мобільному телефоні для перевірки особи напряму.

Для зручності розуміння усього алгоритму, я написав схему.



З схеми видно, що ключ потрібен тільки для розшифрування, ще можна помітити, що відправнику для використання моделі непотрібно запам’ятовувати ID, що забезпечує безпеку при передачі даних. Звичайно він з легкістю може продивитися її сам якщо йому це буде потрібно і він пройде індифікацію.

Для **дешифровки**, а саме для незаконного розшифрування в цю схему потрібно добавити алгоритм стеганографії та алгоритм хешування, а також доступ до носія з інформацією, що ускладнює схему на ще 3 складні етапи, навіть якщо зловмисник має ID-користувача і ключ.

Джерела

1. В.П. Гребєніков. Стеганографія. Історія тайнопису – 2019
2. Брюс Шнайєр, Нільс Фергюсон. Практична кріптографія - 2016
3. Г. Ф. Конахович, А. Ю. Пузиренко. Компьютерная стеганографія: "МК-Пресс" - Київ, 2006
4. Брюс Шнайєр, Нільс Фергюсон. Практична кріптографія - 2016
5. Т. Рафгарден. Досконалий алгоритм. Основи – 2019
6. Геннадій Халімов. Універсальне хешування – 2014
7. https://developer.mozilla.org/uk/docs/Glossary/Encryption
8. https://studopedia.su/14_70526_hesh-funktsii.html

Додатки

Додаток А

Хешування, та шифрування в фото.

```
     import random
#my_file = open("some.txt", "w")
arr = input()
sss = arr
#my_file.write(arr)
#my_file.write('\n')
arr = list(arr.upper())

#----------массив простых чисел------
n = 500
lst=[2]

for i in range(3, n+1, 2):
  if (i > 10) and (i%10==5):
    continue
  for j in lst:
    if j*j-1 > i:
      lst.append(i)
      break
    if (i % j == 0):
      break
  else:
    lst.append(i)
#print (lst)

#-------------------хеширование----------
h0 = h1 = 100
p = 379
```

```
q = 83
r = p*q
print(p, q, r)
if p == q:
    q = random.choice(lst)
r = p*q
alphabet = '.:QWERTYUIOPASDFGHJKLZXCVBNMАБВГҐДЕЄЖЗДИІЙКЛМНОПРСТУФХЦЧШЩЬЮЯ '
alphabet = list(alphabet)

#------------цифровая подпись-------
Kc = random.choice(lst)
delt = (p-1)*(q-1)
if delt%Kc == 0:
    Kc = random.choice(lst)
y = 1
Ko = abs((delt*y +1))%Kc
while Ko != 0:
    y += 1
    Ko = abs((delt * y+1)) % Kc
Ko = int(abs((delt * y+1))/ Kc)
c = 0
for i in range(len(arr)):
    o = ord(arr[i])
    for j in range(len(alphabet)):
        l = ord(alphabet[j])
        c = c + 1
        if o == l:
            h0 = ((h0 + j + 1)**2)%r #вычисление хеш функции сообщения
            h1 = ((h1 + j + 1)**2)%r
            break
print(arr)
print(''.join(arr))
```

```
S = (h0**Kc)%r
print('ID-cust: %d' %(S))
#my_file = open("some.txt", "w")
#--------проверка подлинности-----
s1 = (S**Ko)%r
print('hesh: %d' %(h1))
if s1 == h1:
    print('true')
else: print('false')
print('Ko: ', Ko, 'r: ', r)
print('check = (ID-cust^Ko)/r: %d' %(s1))
with open(f'{S}.txt',"w") as f:
    f.write(sss)
    f.write('\n')
    f.write("ID-cust: ")
    f.write(str(S))
    f.write('\n')
    f.write("hesh: ")
    f.write(str(h1))
    f.write("\n")
    f.write("Ko: ")
    f.write(str(Ko))
    f.write("\n")
    f.write("r: ")
    f.write(str(r))
    f.write("\n")
    f.write("check = (ID-cust^Ko)/r: ")
    f.write(str(s1))

import numpy as np
import scipy.misc
import imageio
```

```
from random import randint
from PIL import Image, ImageDraw

img = Image.open('new.bmp')
draw = ImageDraw.Draw(img)
data = np.asarray(img, dtype='int32')
width = img.size[0]
height = img.size[1]
pix = img.load()
f = open('keys.txt','w')
#print(pix)
#print(width, height)
print(width, height)

red = []
green = []
blue = []

for x in range(len(data)):
    for y in range(len(data)):
       red.append(data[x, y][0])
       green.append(data[x, y][1])
       blue.append(data[x, y][2])
word = S
#word = list(word)
enc = [ord(i) for i in str(word)]
print(enc)
dec = [chr(i) for i in enc]
print(dec)
key = width/10
key = int(key)
if key >= width:
```

```
    print('pls, repeat')
    key = randint(1,(width-1))
    print(key)
count = 0
j = 0
k = 0
x = height-key
y = 0
l = key * width #додаткова умова захисту, залежить від ширини фото
if (l >= 10 and l < 100):
    l = l / 10
if (l >= 100 and l < 1000):
    l = l / 100
if (l >= 1000 and l < 10000):
    l = l / 1000
if (l >= 10000 and l < 100000):
    l = l / 10000
if (l >= 100000 and l < 1000000):
    l = l / 100000
if (l >= 1000000 and l < 100000000):
    l = l / 1000000
l = int(l)
for i in range(len(enc)):
    y = y + key
    x = x - 1
    if x <= 1: x = height-key+1
    if y >= width: y = 0
    data[x, y][1] = enc[count]+l
    count += 1
    print(data[x, y][0], data[x, y][1], data[x, y][2], '\n')
#scipy.misc.imsave('resultat2.bmp', data)
```

```
imageio.imwrite('result2.bmp', data)
print(len(enc))
```

Додаток Б

Розшифрування.

```
import numpy as np
import scipy.misc
import imageio
from PIL import Image, ImageDraw
import os

img = Image.open('result2.bmp')
draw = ImageDraw.Draw(img)
data = np.asarray(img, dtype='int32')
width = img.size[0]
height = img.size[1]
pix = img.load()

print(width, height)
red = []
green = []
blue = []
for x in range(len(data)):
    for y in range(len(data)):
       red.append(data[x, y][0])
       green.append(data[x, y][1])
       blue.append(data[x, y][2])
k = int(input()) #ключ, залежить від довжини айді
key = width/10
key = int(key)
l = key * width #додаткова умова захисту, залежить від ширини фото
text = []
count = 0
```

```
j = 0
x = height-key
y = 0
l = key * width
if (l >= 10 and l < 100):
    l = l / 10
if (l >= 100 and l < 1000):
    l = l / 100
if (l >= 1000 and l < 10000):
    l = l / 1000
if (l >= 10000 and l < 100000):
    l = l / 10000
if (l >= 100000 and l < 1000000):
    l = l / 100000
if (l >= 1000000 and l < 100000000):
    l = l / 1000000
l = int(l)
for i in range(k): #алгоритм розшиврування з використанням ключів
    y = y + key
    x = x - 1
    if x <= 1: x = height - key+1
    if y >= width: y = 0
    text.append(data[x, y][1])
    count += 1
    print(count, data[x, y][0], data[x, y][1], data[x, y][2], '\n')
for i in range(k):
    text[i] = text[i]-l+1
text = list(text)
print(text)
dec = [chr(i) for i in text] #масив в якому зберігається розшифровані данні
dec =''.join(dec)
print(''.join(dec))
#scipy.misc.imsave('resultat2.bmp', data)
imageio.imwrite('result22.bmp', data)
os.startfile(f'{dec}.txt')
```