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

Горбатюк Тетяна Ігорівна,
група РФ-171

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Розробка додатку для шифрування даних зі збереженням формату

Спеціальність:
122 Комп’ютерні науки
Спеціалізація, освітня програма
Комп’ютерні науки та інформаційна безпека

Керівник:
Лебедєва Олена Юріївна,
к.т.н., доцент

Одеса – 2022

Національний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Спеціалізація, освітня програма
Комп'ютерні науки та інформаційна безпека

ЗАТВЕРДЖУЮ
Завідувач кафедри КБПЗ
_____
д.т.н.,проф. А.А.Кобозєва
____ _____202_р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Горбатюк Тетяні Ігорівні

1.Тема роботи: *Розробка додатку для шифрування даних зі збереженням формату,* керівник роботи Лебедєва Олена Юріївна*, к. т. н., доцент каф. КБПЗ,* затверджені наказом ректора від „____"_____ 20___ р. №_____ .

2.Зміст роботи*: аналіз предметної області, постановка задачі, аналіз сучасних методів шифрування зі збереженням формату, аналіз програм на ринку, розробка додатку для шифрування даних зі збереженням формату на основі покращеного методу псевдонімізації та маскування даних.*

3.Перелік ілюстративного матеріалу: *слайди презентації*.

4. Консультанти розділів роботи

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Дата видачі завдання "_____"_____20___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	*Аналіз джерел з теми випускної кваліфікаційної роботи*	*23-09-2022*	*виконано*
2	*Аналіз принципів роботи сучасних методів шифрування зі збереженням формату та методи псевдонімізації даних*	*05-10-2022*	*виконано*
3	*Аналіз сучасних програм по шифруванню зі збереженням формату та методів псевдонімізації даних*	*14-10-2022*	*виконано*
4	*Аналіз технологій необхідних для реалізації програмного продукту*	*19-10-2022*	*виконано*
5	*Розробка удосконаленого методу псевдонімізації та маскування даних*	*20-10-2022*	*виконано*
6	*Реалізація додатку для шифрування даних зі збереженням формату*	*02-11-2022*	*виконано*
7	*Дизайн та реалізація інтерфейсу програмного продукту*	*17-11-2022*	*виконано*
8	*Підготовка тексту роботи*	*25-11-2022*	*виконано*
9	*Підготовка презентації та доповіді*	*30-11-2022*	
9	*Попередній захист*	*08-12-2022*	*виконано*
10	*Нормоконтроль, рецензування*	*20-12-2022*	*виконано*

Здобувач вищої освіти_____ *Горбатюк Т. І.*

Керівник роботи_____ *Лебедєва О. Ю.*

АНОТАЦІЯ

Кваліфікаційна робота на тему "Розробка додатку для шифрування даних зі збереженням формату" на здобуття другого (магістерського) рівня вищої освіти за спеціальністю 122 Комп'ютерні науки, спеціалізація, освітня програма - Комп'ютерні науки та інформаційна безпека, містить 33 рисунки, 1 додаток, 55 літературних джерел за переліком посилань. Робота виконана на 103 сторінках загального тексту і 81 сторінці основного тексту.

Метою роботи є розробка додатку для шифрування персональних даних зі збереженням формату шляхом удосконалення методу псевдонімізації та маскування даних.

У результаті виконання кваліфікаційної роботи було розроблене програмне забезпечення для шифрування персональних даних зі збереженням формату та удосконаленими методами псевдонімізації полів даних: типу ім'я, прізвище та email адреса та імплементоване розумне застосування методу FPE в режимі FF3-1 для полів типу номер телефону та кредитної карти. Результати даної роботи можуть бути можуть бути використані для підвищення стійкості до кібератак підприємств та покращення захисту персональних даних, якими вони користуються і зберігають.

ПЕРСОНАЛЬНА ІНФОРМАЦІЯ, ШИФРУВАННЯ ЗІ ЗБЕРЕЖЕННЯМ ФОРМАТУ, ПСЕВДОНІМІЗАЦІЯ, МАСКУВАННЯ ДАНИХ.

ANNOTATION

Qualification work on the topic "Development of an application for data encryption with format preservation" for obtaining the second (master's) level of higher education in the specialty 122 Computer science, specialization, educational program - Computer science and information security, contains 33 drawings, 1 appendix, 55 literature sources in the list of references. The work is done on 103 pages of the general text and 81 pages of the main text.

The purpose of the work is to develop an application for encrypting personal data with format preservation by improving the method of pseudonymization and data masking.

As a result in this qualification work, the application for encrypting personal data with format preservation and improved methods of pseudonymization was developed for data fields: first name, last name, and email address, and the FPE method was implemented in FF3-1 mode for fields such as phone number and credit card cards. The results of this work can be used to increase the resistance to cyber attacks of emnterprises in order to improve the protection of personal data that they use and store.

PERSONAL INFORMATION, FORMAT PRESERVING ENCRYPTION, PSEUDONYMISATION, DATA MASKING.

ЗМІСТ

ВСТУП

За останні кілька років зростання кількості витоків персональних даних призвело до втрати цілісності даних для сотень мільйонів записів користувачів. Такі атаки спрямовані як проти великих компаній, так і проти малих підприємств. Частка кібератак з метою крадіжки персональних даних у підприємств забезпечена через халатність компаній щодо коректного шифрування та маскування персональних даних працівників та клієнтів.

Доступ до ідентифікаційної інформації без авторизації становить значний ризик як для окремих осіб, так і для компанії. Індивідуальна шкода може включати крадіжку особистих даних, шахрайство або шантаж. Водночас компанія може зазнати шкоди через порушення цілісності даних, наприклад втрати довіри громадськості, юридичної відповідальності або великих штрафів.

Витоки даних стають все більш поширеними, і заголовки видань часто наповнюються історіями про те, як компанії втрачають дані клієнтів. Але витоки даних — проблема не лише великих компаній — вони можуть трапитися з ким завгодно.

Під час внутрішніх тестів на проникнення в мережу консультанти з безпеки часто знаходять конфіденційну інформацію, яка незахищено зберігається на файлових серверах. Це включає чутливі дані про особу (повне ім’я, телефони, адреси, як фізичні так і електронні), номери кредитних карток, паспортів, сторонніх додатків для входу в платіжні системи, аналітику та інші веб-маркетингові та бізнес-портали. Це один із головних ризиків, виявлених під час незахищених практик зберігання інформації.

Оскільки кожне підприємство, незалежно від його розміру, підлягає під закони про ідентифікаційну інформацію, нормативні акти та інші повноваження, пов’язаних із захистом ідентифікаційної інформації, воно повинне дотримуватися чіткого підходу щодо безпеки та конфіденційності даних, щоб захистити ідентифікаційну інформацію, яку зберігає і використовує у своєму середовищі. [1]

Часто перепоною для компаній, що оперують персональними даними людей є комплексність в реалізації шифрування даних, які вони зберігають. Це пов’язане з тим, що часто вони використовують застарілі системи, чиї кодові бази та моделі даних не можна змінити або надто обтяжливі та ризиковані для оновлення. Наприклад, база даних, яка зберігає конфіденційні дані про охорону здоров’я, яка була введена в дію майже 20 років тому і продовжує працювати. Більшість організацій схильні до ризиків, пов’язаних з реструктуризацією такої ключової виробничої системи. [2]

У зв'язку з цим є актуальним завданням є втілити методи які б допомагали таким компаніям шифрувати та маскувати дані з мінімальними вкладами в реструктуризацію та збереження функціональності на льоту.

Метою даної роботи є розробка додатку для шифрування персональних даних зі збереженням формату шляхом удосконалення методу псевдонімізації та маскування даних.

Для досягнення поставленої мети необхідно розв’язати наступні задачі:

1. Проаналізувати принципи роботи сучасних методів шифрування зі збереженням формату та методи псевдонімізації даних.
2. Проаналізувати та визначити переваги та недоліки сучасних програм по шифруванню зі збереженням формату та методів псевдонімізації даних.
3. Удосконалити метод псевдонімізації та маскування даних.
4. Реалізувати додаток для шифрування даних зі збереженням формату на основі розробленого покращеного методу псевдонімізації та маскування даних.

Об’єктом дослідження є удосконалення методу псевдонімізації та маскування даних.

Предметом дослідження є методи шифрування зі збереженням формату та псевдонімізації даних.

Новизна кваліфікаційної роботи полягає в удосконаленні методу псевдонімізації та маскування даних без зміни формату даних.

Практична цінність роботи полягає в реалізації удосконаленого методу псевдонімізації та маскування даних без зміни формату даних у вигляді програмного додатку для шифрування даних.

Отримані результати можуть бути використані для підвищення стійкості до кібератак підприємств та покращення захисту персональних даних, якими вони користуються і зберігають.

Результати досліджень були подані у науковій статті «Розробка додатку для шифрування даних зі збереженням формату», що опублікована у журналі «######».

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1.Огляд ключових понять

Зі швидким технологічним розвитком, включаючи постійне зростання онлайн-покупок і недавній бум віддаленої роботи та навчання, безпека ідентифікаційної інформації стає все більш важливою для індивідів та компаній.

Без моніторингу конфіденційності чи захисту даних ідентифікаційної інформації люди та клієнти стають вразливими до зламів і порушень, які можуть призвести до серйозних наслідків, а саме викрадення особистих даних. З юридичної точки зору, розуміння загальної картини захисту конфіденційності в Інтернеті має вирішальне значення для безпеки окремої особи чи організації та дотримання законодавства.

Особиста інформація (англ. Personal Identifiable Information - PII) – це будь-які дані, які можуть бути використані для ідентифікації конкретної особи. Приклади включають повне ім’я особи, телефонні та номери соціального страхування, фізичні або імейл адреси, тощо[3].

Особиста інформація містить не лише очевидні посилання на особистість людини, а і інформаційні фрагменти, які в поєднанні з іншими наборами даних розкривають особу, тобто класифікується як ідентифікаційна інформація (рис.1.1).

Конфіденційна ідентифікаційна інформація включає будь-який набір даних, який містить повне ім’я, адресу або фінансову інформацію (рис.1.1). Неконфіденційна ідентифікаційна інформація – це будь-які загальні дані, доступні з загальнодоступних ресурсів (таких як профілі в соціальних мережах), наприклад поштовий індекс або дата народження.

Неконфіденційні дані знаходяться в сірій області. Хоча вони достатньо загальні, щоб застосовувати їх до широкого сегменту населення, але такі дані можна використовувати поряд з іншими наборами даних, щоб розкрити конкретну особистість людини. [3]

Конфіденційна ідентифікаційна інформація включає, але не обмежується, такими унікальними ідентифікаторами:

- ім'я – повне ім'я, дівоче прізвище, дівоче прізвище матері або псевдонім;
- інформація про адресу – вулиця, робоча адреса або адреса електронної пошти;
- персональний ідентифікаційний номер: номер соціального страхування (SSN), номер паспорта, номер водійського посвідчення, ідентифікаційний номер платника податків, номер фінансового рахунку, номер банківського рахунку або номер кредитної картки;
- фінансова інформація (номери кредитних/дебетових карток, банківських рахунків;
- медичні дані;
- IP-адреси. Деякі юрисдикції навіть класифікують IP-адреси як ідентифікаційну інформацію.

Неконфіденційна ідентифікаційна інформація – це будь-яка інформація, яка потенційно може бути пов’язана з особою. Приклади:

- раса;
- стать;
- місце народження;
- дата народження;
- релігійні переконання.

Як і будь-яка форма даних, не всі ідентифікаційні дані однакові. Слід оцінити ідентифікаційну інформацію шляхом визначення рівня її впливу на конфіденційність ідентифікаційної інформації.

Рівні впливу на конфіденційність ідентифікаційної інформації варіюються від низького, помірного або високого, щоб вказати на потенційну шкоду, яка може бути завдана особі чи організації, якщо дані будуть скомпрометовані.[1]

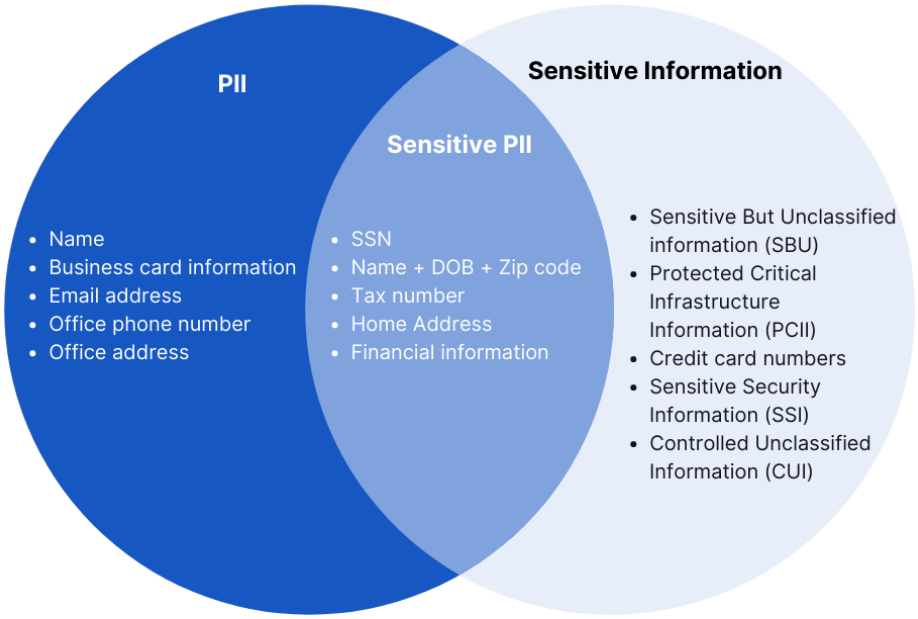


Рисунок 1.1 — Розділення особистої інформації на категорії

У більшості юрисдикцій ідентифікаційна інформація має бути захищена додатковими вимогами безпеки, і в багатьох галузях існують закони про конфіденційність даних.

З юридичної точки зору відповідальність за захист ідентифікаційної інформації може коливатися від відсутності відповідальності до виключної відповідальності організації. Як правило, відповідальність розподіляється з організацією, яка володіє ідентифікаційною інформацією, і окремим власником даних.

Постійно зростаюча кількість випадків витоку даних, пов’язаних із особистою (ідентифікаційною) інформацією, призвела до втрат акціонерів на мільярди доларів, штрафів на мільйони доларів і збільшення ризику крадіжки особистих даних осіб, чиї конфіденційні дані були розкриті. Порушення безпеки даних є небезпечним як для окремих осіб так і організацій:

- індивідуальна шкода: крадіжка особистих даних, шантаж;

– Організаційні збитки: втрата громадської довіри, юридична відповідальність, зниження вартості підприємства, закриття бізнесу або витрати на відновлення.

Щоб захистити конфіденційність ідентифікаційної інформації, організаціям необхідно використовувати оцінку ризиків кібербезпеки, управління ризиками третіх сторін, управління ризиками постачальників та управління інформаційними ризиками. [3]

Захист персональних даних давно регулюється в законодавстві різних країн.

Наприклад, в Україні роблему розкриття інформації про фізичних та юридичних осіб в законодавстві України регулюють декілька законів. До них належать Закон «Про інформацію», який регулює відносини, пов’язані з отриманням та поширенням інформації, Закон «Про захист персональних даних», який визначає захист та обробку персональних даних, та Закон «Про доступ до публічної інформації», який надає право на доступ до інформації, котрою володіють розпорядники.[4]

У Європейському Союзі є Директива 95/46/ЕС, що визначає персональні дані як інформацію, яка може ідентифікувати особу, наприклад ідентифікаційний номер або інші фактори, специфічні для фізичної, фізіологічної, психічної, економічної, культурної чи соціальної ідентичності.

У Сполучених Штатах існує Посібник Національного інституту стандартів і технологій (NIST) із захисту конфіденційності інформації, що дозволяє ідентифікувати особу, визначає ідентифікаційну інформацію як будь-яку інформацію про особу, яку зберігається, включно з будь-якою інформацією, яка може бути використана для розрізнення або її відстеження. Також може бути додатковою інформацією, такою як захищена інформація про здоров’я, освіту, фінанси та інформацію про роботу.

У Великобританії Закон про захист даних 2018 року є імплементацією Великобританією Загального регламенту захисту даних (GDPR). Він вимагає, щоб ідентифікаційна інформація використовувалася чесно, законно та прозоро, для конкретної,

чіткої мети, таким чином, який є адекватним, актуальним і обмеженим лише тим, що необхідно. Точні та, за необхідності, оновлені, зберігаються не довше, ніж це необхідно, і обробляються таким чином, щоб забезпечити відповідну безпеку, включаючи захист від незаконної або несанкціонованої обробки, доступу, втрати, знищення або пошкодження.

Канада має Закон про захист особистої інформації та електронних документів (PIPEDA) гарантує, що організації повинні отримати згоду особи на збір, використання або розкриття ідентифікаційної інформації.[3]

Хакерів часто приваблюють бізнеси, тому що вони витягують велику кількість ідентифікаційної інформації за один раз, а потім продають ці конфіденційні дані на чорному ринку. Незахищену персональну інформацію можна використовувати не тільки для крадіжки особистих даних, а для шахрайства та атак соціальної інженерії. [5]

В ідеалі, кожна організація шифрувала б конфіденційні дані, коли це можливо, розшифровуючи лише тоді, коли потрібен доступ до відкритих текстових даних. Проте застарілі програми та системи можуть очікувати, що дані будуть у певному форматі та не зможуть обробляти чи зберігати інші типи даних без істотних змін.

Це особливо критично для організацій, які використовують застарілі системи, чиї кодові бази та моделі даних не можна змінити або надто обтяжливі та ризиковані для оновлення. Наприклад, база даних, яка зберігає конфіденційні дані про охорону здоров’я, яка була введена в дію майже 20 років тому і продовжує працювати. Більшість організацій схильні до ризиків, пов’язаних з реструктуризацією такої ключової виробничої системи. [2]

Наприклад, якщо для стовпцю бази даних, у якому зберігаються номера кредитних карток (з максимальним обмеженням довжини 16 символів) застосувати популярний симетричний алгоритм (наприклад, AES-256-GCM), то отриманий зашифрований текст, безсумнівно, перевищить максимальну довжину, дозволену базою даних, що призведе до помилок в обробці даних.

Виникає питання, як зашифрувати дані без реструктуризації чи руйнування бази даних. Популярним варіантом є шифрування із збереженням формату (з англ. FPE - Format-preserving encryption).

Шифрування із збереженням формату (FPE) — це процес шифрування даних таким чином, щоб вихідні дані (зашифрований текст) залишалися в тому самому форматі, що й вхідні дані (відкритий текст). Значення «формату» різне. Зазвичай використовуються лише кінцеві набори символів; цифровий, буквений або буквено-цифровий». [2]

FPE дає змогу розгортати широко поширене шифрування без критичних наслідків для застарілих систем. Додаткам, яким не потрібен доступ до конфіденційних даних, надаються шифртексти FPE. Якщо програмі потрібен доступ до даних, у робочий процес можна вставити функцію дешифрування, щоб надати доступ до відкритого тексту.

Ось кілька прикладів як можна застосовувати FPE:

- для перевірки платіжних карток: у секторі роздрібної торгівлі та електронної комерції дані платіжної картки необхідно збирати та зберігати для здійснення платежів. Крім того, працівникам може знадобитися переглянути та перевірити останні чотири цифри інформації платіжної картки клієнта. FPE полегшує надання працівникам лише необхідної інформації, залишаючи інші дванадцять цифр захищеними;
- для застарілих баз даних: телекомунікаційна система може мати безліч застарілих систем, які потребують використання зашифрованих даних для безпеки. Однак організація може не мати можливості реструктуризувати свої бази даних для зберігання зашифрованих даних. З FPE структура баз даних може залишатися незмінною.

Хоча це лише два потенційних варіанти використання FPE, будь-які можливості шифрування конфіденційних даних у обмеженому середовищі зберігання можуть бути чудовим кандидатом для FPE. Окрім очевидної переваги безпеки шифрування

даних, які раніше були відкритими, FPE потребує значно менше зусиль і ресурсів для реалізації порівняно з іншими рішеннями, оскільки не потребує модифікації рівня зберігання. Якщо зашифрований номер кредитної картки все ще виглядає як номер кредитної картки, для бази даних не має значення, чи потрібно розшифрувати, перш ніж він стане придатним для використання.[2]

Є дві основні переваги використання шифрування зі збереженням формату. По-перше, шифрування можна застосовувати до застарілих систем, які вимагають даних у певному форматі, без необхідності вносити дорогі та трудомісткі зміни в моделі даних і код. Ось чому FPE зазвичай використовується для захисту наборів конфіденційних даних, таких як дані платіжних карток, реквізити банківських рахунків, номери соціального страхування та ідентифікаційна інформація, яка обробляється і зберігається в базах даних і програмах роздрібної торгівлі, охорони здоров'я та фінансів. Тому що, багато додатків для роздрібної торгівлі, охорони здоров'я та фінансів написані мовою COBOL — це все ще одна з найпоширеніших мов програмування, яку використовують великі підприємства. Щоб привести ці програми у відповідність до стандартів захисту даних, таких як Каліфорнійський закон про конфіденційність споживачів (CCPA) - США і Загальний регламент захисту даних (GDPR)- Європа, шифрування всього диска або на рівні файлової системи не є довгостроковим варіантом. Захист даних на рівні поля часто є єдиним реалістичним підходом до виконання нормативних вимог, але оскільки COBOL є строго типізованою мовою, для обробки зашифрованих даних вимагатиме ретельного аналізу та переписування коду. Алгоритм FPE шифрує значення даних, які зберігають вхідний формат і довжину даних, дозволяючи їм проходити через системи, які розпізнають лише певний формат, залишаючись у захищеному стані.

Друга головна перевага FPE перед звичайним шифруванням полягає в тому, що такі дані, як номери кредитних карток або номери соціального страхування, усе ще можуть використовуватися як унікальний ключ для ідентифікації рядка в базі даних. Дані, зашифровані за допомогою режиму CBC змінюють значення даних, коли

їх розшифровують і знову шифрують. Це означає, що багато обробок зашифрованих FPE даних можна виконувати з даними в захищеному стані. Лише в конкретних випадках використання — наприклад, передача номера соціального страхування агентству кредитних карток для перевірки кредитоспроможності — зашифрований текст потрібно буде перетворити назад у відкритий текст. Крім того, PII можна анонімізувати або псевдонімізувати за допомогою FPE, що робить аналіз даних, що містять, наприклад, медичну інформацію, набагато простішим. [6]

Наступним терміном для аналізу є псевдонімізація, що походить від слова псевдонім. Псевдонім у перекладі з грецької означає «фальшиве ім'я», і відомим прикладом є вигаданий персонаж Брюс Вейн, якого іноді називають Бетменом. Подібно до Брюса Вейна та Бетмена, псевдонімізація в IT-системах означає, що ви маскуєте зареєстрованих користувачів та їхні особисті дані.

У GDPR псевдонімізація визначається як «обробка персональних даних таким чином, що дані більше не можуть бути пов'язані з певним суб'єктом даних без використання додаткової інформації». Таким чином відбувається обмін особистими даними з неідентифікуючими даними, і для відтворення вихідних даних потрібна додаткова інформація. Крім того, додаткову інформацію слід зберігати окремо.

Псевдонімізація робить таку інформацію, як персональні ідентифікаційні номери та особисті дані менш доступною для неавторизованих користувачів у відповідності до вимог GDPR.[7]

Наприклад, за допомогою анонімізації дані очищуються щоб виявлення будь-якої інформації, яка може служити ідентифікатором суб'єкта даних було неможливим. Натомість, псевдонімізація не видаляє всю ідентифікаційну інформацію з даних, а лише зменшує зв'язок набору даних з оригінальною ідентичністю особи (наприклад, за допомогою схеми шифрування). [8]

Псевдонімізація може бути досягнута за допомогою різних методів, таких як маскування даних, шифрування або токенізація. Вона зазвичай використовується як

техніка для захисту особистих даних у застарілих виробничих системах від несанкціонованого доступу, де інші методи безпеки непридатні.

Однак, впроваджуючи нові IT-системи, організаціям потрібно думати про захист даних за проектом і за замовчуванням. А системи з архітектурою даних, що використовує псевдонімізацію, можуть бути дуже ефективними.

Іншим поширеним випадком використання псевдонімізації в виробничих системах, які обробляють персональні дані, є тимчасове зберігання початкових значень під час анонімізації особистих даних для механізму відновлення після відмови.

У цьому випадку псевдоніми можуть зберігатися протягом короткого періоду часу, достатнього для того, щоб компанія підтвердила успішне завершення анонімізації.[9]

Агентство Європейського Союзу з кібербезпеки (ENISA) опублікувало звіт «Методи псевдонімізації та найкращі практики», спровокований проблемами впровадження псевдонімізації на практиці. Посібник обговорює критерії вибору належних методів псевдонімізації, таких як захист даних, масштабованість і відновлення. Посібник також відображає конкретні випадки використання різних ідентифікаторів, таких як IP-адреса або адреса електронної пошти.

У звіті зроблено висновок, що не існує лише одного рішення чи одного способу запровадження псевдонімізації, який працює для всіх галузей чи всіх сценаріїв. Навпаки, він вимагає високого рівня компетентності для застосування надійного процесу псевдонімізації, можливо, зменшуючи загрозу дискримінації або атак повторної ідентифікації, зберігаючи при цьому ступінь корисності, необхідний для обробки псевдонімізованих даних».[9]

1.2 Огляд існуючих режимів шифрування зі збереженням формату

У 2016 році NIST випустив «Спеціальну публікацію (SP) 800-38G, Рекомендації щодо режимів роботи блокового шифру: методи шифрування зі збереженням фо-

рмату». NIST спочатку розглядав три режими роботи - FF1, FF2 і FF3 - з використанням алгоритму блокового шифрування Advanced Encryption Standard. [10]

FF2 так і не отримав схвалення. У 2017 році дослідники здійснили криптоаналітичну атаку на FF3 [11], зробивши його непридатним для FPE загального призначення, оскільки він не досяг запланованого 128-бітного рівня безпеки.

У відповідь на атаку NIST оновив FF3 до FF3-1 [12] на початку 2019 року. Оновлення спрямовано на потенційні вразливості, де кількість можливих вхідних даних, тобто розмір домену, є достатньо малим, наприклад, використовуються середні шість цифр номера кредитної картки або соціального страхування. У таких випадках не вистачає ентропії для створення безпечного виходу, який не піддається зворотній інженерії. У SP 800-38G розмір домену для FF1 і FF3 мав бути принаймні 100 і рекомендовано принаймні 1 000 000. Тепер розмір домену обов'язково повинен сягати бути 1 000 000. Організації, що розглядають імплементацію FPE, повинні забезпечити відповідність потенційних продуктів і послуг оновленим вимогам. [9]

Альтернативні підходи до захисту FPE із збереженням формату, такі як токенізація, використовується, якщо реалізація не є надто складною. Токенізація обмінюється конфіденційними даними з випадковими значеннями в тому самому форматі, але це не має власної внутрішньої цінності. Вихідні дані зберігаються в безпечному сховищі даних. Це не те саме, що шифрування. Зашифровані дані можна розшифрувати за допомогою відповідних ключів, тоді як токенізаційні маркери не можна реверсувати, оскільки між маркером і його вихідним значенням немає математичної залежності. Це означає, що існує більша гнучкість із широким набором токенів, у які можна конвертувати дані. [10]

FPE має три різні режими роботи: FF1, FF2 і FF3, які в цілому називаються FFX.

FFX використовує кілька раундів функції Feistel над відкритим текстом разом із ключем для створення зашифрованого тексту. Функція Feistel розділяє відкритий текст на дві частини, переставляє текст, щоб змінити його вигляд, а потім змінює лі-

ву половину тексту на праву і навпаки. Метод FF1 використовує 10 раундів функції Фейстеля, а FF3 використовує 8 раундів. [13]

Усі три методи роботи використовують блоковий шифр AES у своєму шифруванні. Другий режим, FF2, був створений, але так і не затверджений Національним інститутом стандартів і технологій (NIST). Натомість були затверджені FF1 і FF3. Версію FF1 прийнято вважати більш стійкою через криптоаналітичну атаку, виконану на метод FF3, яка показала, що він містить недоліки. Ця атака виявила, що запропонований 128-бітний рівень безпеки не досягнуто. У відповідь на початок 2019 року було створено покращений метод під назвою FF3-1, який вирішував ці проблеми.

FPE надзвичайно добре працює як з існуючими, так і з новими програмами. Якщо програмі потрібні дані певної довжини та формату, то FPE можна застосувати до даних для їх шифрування, не вимагаючи зміни програми. Це особливо добре працює з програмним забезпеченням, яке не може обробляти довгі рядки даних. Шифрування із збереженням формату є дійсним алгоритмом шифрування, який слід використовувати для відповідності стандартам NIST. Публікація NIST під назвою NIST 800-38G була створена для розгляду FPE. Ця публікація NIST зосереджується на трьох методах шифрування із збереженням формату, описуючи технічні деталі кожного режиму роботи. [13]

Ключова перевага схем шифрування із збереженням формату полягає в тому, що вони дозволяють безпечно шифрувати відкритий текст на невеликих просторах домену [14]. Однак це відбувається разом із введенням параметра під назвою tweak, який надається алгоритму FF3-1. До недавнього часу офіційні визначення безпеки в схемах FPE передбачали, що зловмисник має доступ лише до одного налаштування під час перевірки кодової книги на певний ключ. Однак після роботи Bellare це поняття було змінено, щоб припустити, що зловмисник може контролювати кілька налаштувань під час генерації кодової книги та намагаючись створити нетривіальний розрізнювач для FF3-1.

Найкращі практики під час використання налаштувань із перетворенням FF3-1:

- шифрування має бути максимально обмеженим через ACL(Access Lists - списки доступу);
- твіки не контролюються користувачем, а вибираються випадковим чином і розглядаються як конфіденційні (якщо не секретні);
- твіки використовують максимальну довжину, дозволену для байтового рядка, і мають високу ентропію (наскільки рівномірно змінюється в максимальному просторі домену налаштування та різниться зкожним шифруванням). Твіки повинні використовувати весь простір домену;
- слід подбати про те, щоб одне й те саме налаштування не використовувалося для шифрування двох однакових значень, які відповідають різним базовим об’єктам [14].

FF3-1 — це настроювана схема шифрування, створена для вирішення проблеми шифрування на невеликих доменах. Було введено налаштування для синтетичного збільшення доменного простору. Із запровадженням налаштувань зловмисник тепер повинен мати правильне шифрування відкритого тексту та правильне налаштування, щоб повернутися через кодову книгу з шифруваного до відкритого тексту. [15]

Однак це може бути ризиковано, якщо кількість налаштувань, що використовуються в налаштуваннях FF3-1, замала або вибрано в обмеженому домені, або коли їх створюють користувачі, а контроль доступу до шифрування є занадто складним. У першому випадку кодову книгу все ще можна легко створити, якщо налаштування відомі (оскільки значення параметрів не обов’язково є секретними [17]). В останньому випадку зловмисник, який отримує доступ до шифрувальних даних (тобто вибрав міцність відкритого тексту), може маніпулювати налаштуваннями, щоб фактично обійти це синтетичне розширення простору домену.

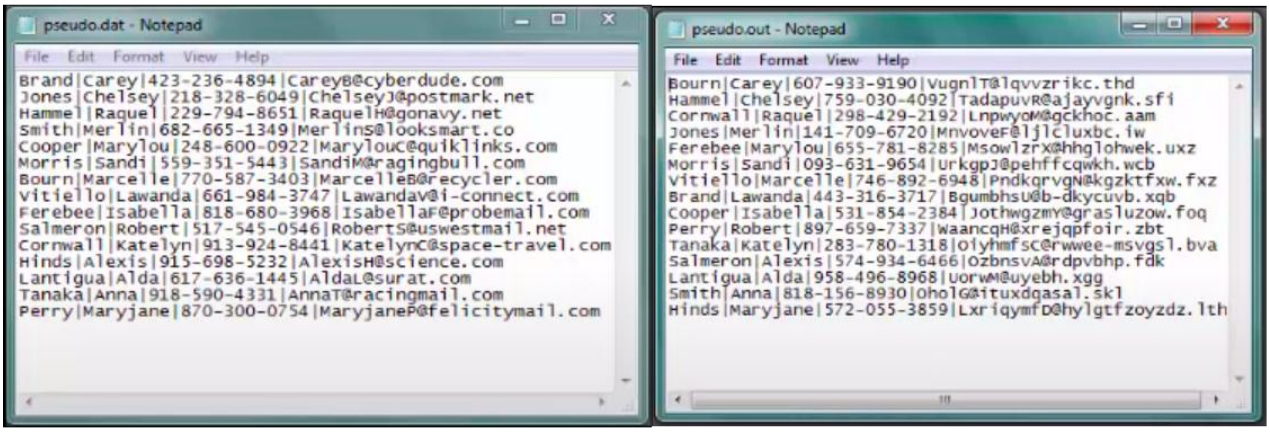
Ще одне зауваження щодо FF3-1 полягає в тому, що шифрування того самого відкритого тексту з тим самим ключем і налаштуванням є детермінованим, тому два відкриті тексти, які випадково збігаються, призведуть до двох зашифрованих текстів, які збігаються. Таким чином, жодні відповідні відкриті тексти для різних об’єктів не повинні зберігатися з однаковими налаштуваннями. Наприклад, якщо зберігаються зашифровані тексти, що відповідають останнім чотирьом цифрам телефонних номерів, то можуть виникнути колізії відкритого тексту, і ці значення колізій не слід шифрувати за допомогою того самого налаштування [16].

Як третє спостереження, FF3-1 ділить налаштування навпіл і використовує половину налаштування в кожній функції раунду. Тому в ідеалі випадково вибрані налаштування повинні мати високу ентропію в кожній половині налаштування. У гіршому випадку створення багатьох шифрувань із налаштуваннями з однаковою правою чи лівою половиною призведе до більш упередженого шифрування (де відмінність між відповідними шифруваннями покладається виключно на випадковість оператора абелевого підсумовування). На практиці використання випадково вибраних налаштувань буде достатнім - однак важливо уникати використання алгоритму для генерації налаштувань, який дає схожість між двома половинами налаштувань [17].

1.3 Аналіз продуктів з імплементованим методами псевдонімізації та шифруванням зі збереженням формату

Багато постачальників пропонують свої послуги FPE, зокрема IRI TheCo-?ortCo, Comforte, HashiCorp, Futurex і Xmart Solutions. Деякі вендори хмарних послуг (CSP) пропонують варіанти використання FPE на своїй платформі, але не настільки розширено, як звичайні постачальники. З трьох найбільших CSP, Microsoft Azure, Amazon Web Services (AWS) і Google Cloud Platform (GCP), лише GCP пропонує користувачам можливість працювати з інструментом шифрування із збереженням формату.

Першим вендором є компанія IRI TheCoSortCo з своїм продуктом IRI FieldShield. Згідно цієї відеодемонстрації [18] можна побачити демонстрацію FPE шифрування і окремо виділити деякі недоліки. Наприклад, якщо порівняти вхідний (рис. 1.2 - а) і вихідний (рис.1.2 - б) (зашифрований) датасети, то імена юзерів просто перемішані між собою в межах документу, що є не зовсім надійною практикою для маскування даних. Якщо звернути увагу на прізвища, то дані не відрізняються між вхідними та вихіними, що вказує на факт, що ніякого маскування чи шифрування застосовуваного до цього поля не було. Окремого акценту також заслуговує поле email адреси, котре відразу викликає підозру у тому, що дані зашифровані завдяки шифруванню частини домену в адресі.



а

б

Рисунок 1.2 — Скріншот вхідного та вихідного набору даних показаного у відеопрезентації продукту IRI FieldShield (Format Preserving Encryption (FPE) and Pseudonymization Demonstration)

Також на іншому відеоогляді [19] можна побачити реалізовану частину псевдонімізації даних типу маскування за допомогою знаків * (рис.1.3).

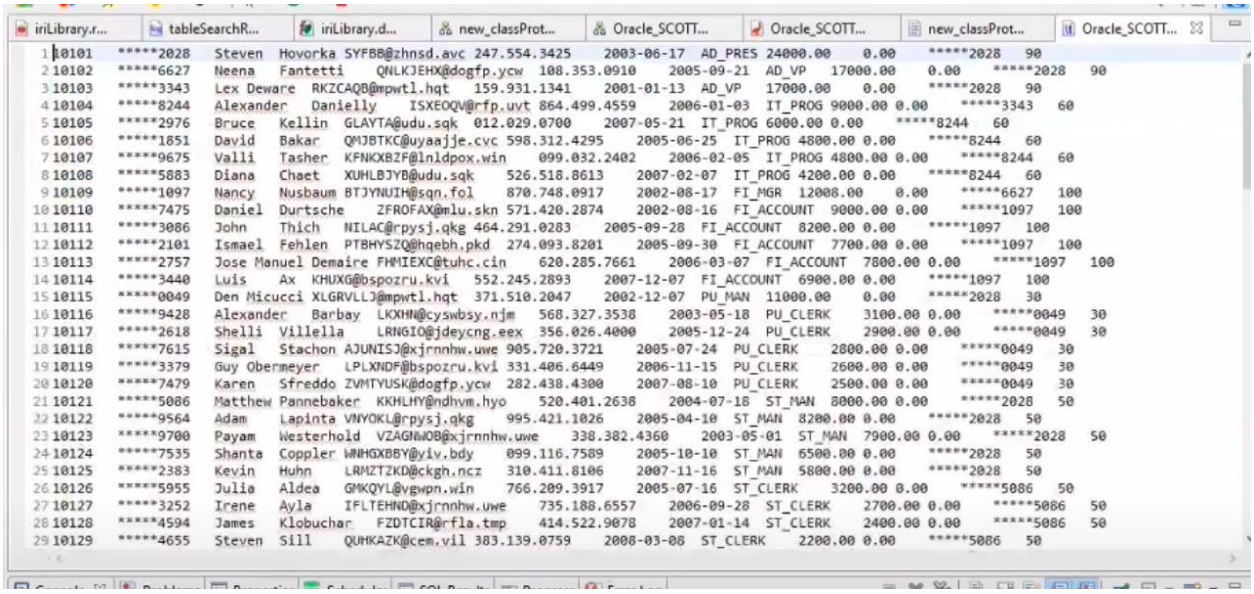


Рисунок 1.3 — Скріншот вихідного набору даних показаного у відеопрезентації продукту IRI FieldShield (Consistent RDB Masking)

Щодо цінової категорії продукту IRI FieldShield компанії IRI TheCoSortCo, то початкова ціна $4000 безстрокового використання або від 4 до 40 тисяч доларів США за хост залежно від конфігурації [20].

Другим постачальником є HashiCorp з продуктом Vault, який пропонує лише алгоритм FPE в режимі FF3-1 (рис. 1.4). Відмінність від попереднього у тому, що клієнт HashiCorp повинен володіти навичками програмування для того, щоб використовувати шифрування у Vault. Сам розділ конфігурації і окремих налаштувань алгоритму під поля даних призначений лише для розробників, тобто людини, яка повинна мати навички програмування, щоб уміти користуватись продуктом, і це далеко не кожен пересічний користувач. Клієнту потрібно самому підключати кастомні алфавіти для шифрування зі збереженням формату та налаштовувати поля згідно бажаної конфігурації [21].

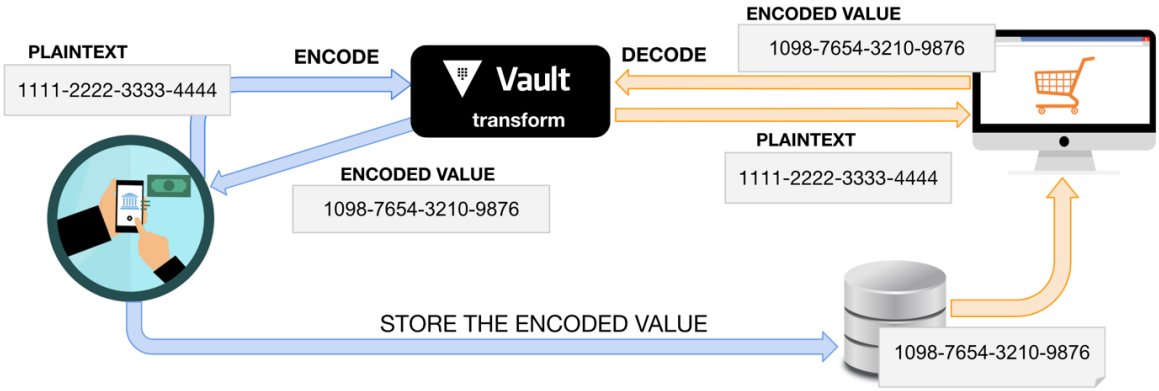


Рисунок 1.4 — Схема роботи Vault Enterprise 1.4 with Advanced Data Protection module

Також, у продукті реалізоване маскування даних через приховування символами * або # (рис.1.5):

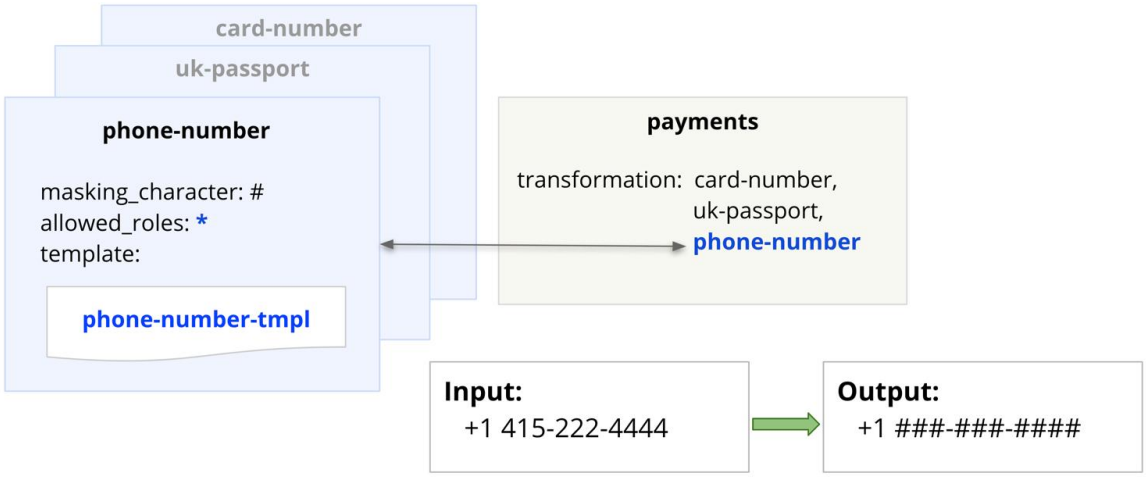


Рисунок 1.5 — Схема маскування даних від HashiCorpVault

Цінова категорія продукту Vault від HashiCorp представлена на рис. 1.6.

Тариф Standard обійдеться у $1.578/hr, що у місяць буде коливатись від $400 до близько $1100 [22].

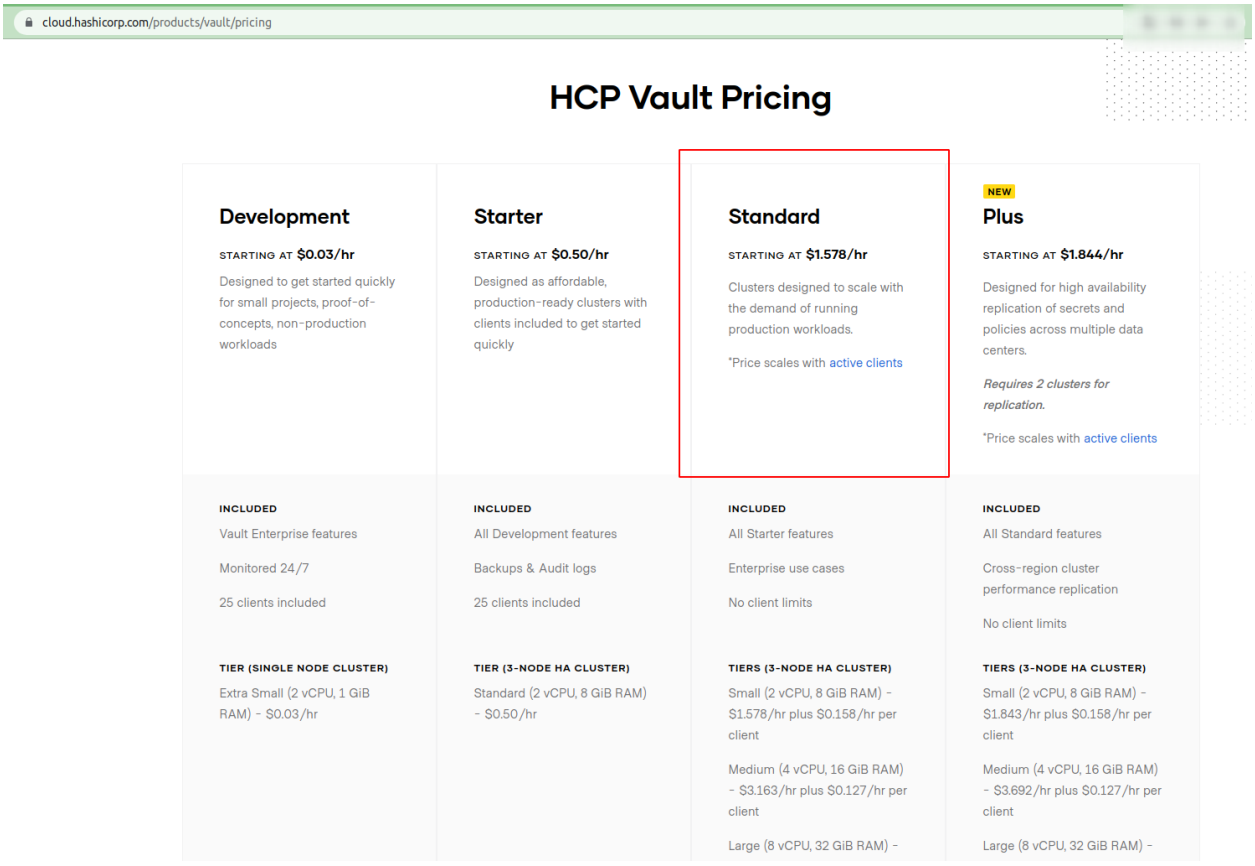


Рисунок 1.6 — Цінова категорія на HashiCorp Vault

Наступним вендором є компанія Comforte з продуктом SecurDPS. Comforte застосовують парадигму FPE, але у відкритих джерелах не вказаний режим який використовується для шифрування даних. У презентації продукту [23] наявне зображення прикладу шифрування email адреси без домену, та шифрування прізвища за допомогою FPE. Процес маскування даних реалізоване через маскувальні символи “X” або “0” (рис. 1.7).

Рисунок 1.7 — Приклад FPE та маскування даних від Comforte у їх продукті SecurDPS

Ціна не розповсюджена у відкритому доступі, але компанія пропонує 14 днів безкоштовного користування для початку.

Ще одним вендором для порівняння є компанія Futurex яка вийшла на ринок з продуктом VirtuCrypt [24]. У текстовому описі продукту наявний коментар лише про те, що Futurex використовує шифрування зі збереженням формату (FPE) за стандартом NIST (але режим невідомий) та приклад застосування FPE до зберігання номерів кредитних карток.

Ціна з урахуванням регіону та швидкості доступу до хмарного середовища коливається від $250 до $2000.

Остання компанія у порівняннях є компанія Xmart Solutions та їх продукт Voltage Security (Hyper FPE) [25]. Hyper FPE використовує режим FF1 для шифрування зі збереженням формату та застосовується до всіх типів даних. Як видно на

скріншоті (рис. 1.8) з документації важливим акцентом цього продукту є маскування полів імені та прізвища, але це явно відображає факт зашифрованих даних. Також, відсутній приклад шифрування поля email адреси.

Рисунок 1.8 — Приклад FPE шифрування даних від Xmart Solutions у їх продукті Voltage Security (Hyper FPE)

Ціна на продукт Voltage Security (Hyper FPE) не вказана у відкритих джерелах.

Отже, провівши аналіз продуктів від найбільш відомих вендорів на ринку постачальників механізму FPE та маскування даних можна дійти висновку щодо не завжди доречного або коректного шифрування чи маскування полів email адрес, імен та прізвищ юзерів, а також не враховані коди номерів телефонів. Навіть неозброєним оком прості юзери можуть помітити факт шифрування та приховування даних. Найбільш використовуваним режимом шифрування FPE є FF3-1, який станом на грудень 2022 року є стандартом NIST для шифрування даних в межах невеликих доменів і з підтвердженим рівнем безпеки.

2. УДОСКОНАЛЕННЯ МЕТОДУ ПСЕВДОНІМІЗАЦІЇ ТА МАСКУВАННЯ ДАНИХ

2.1 Теорія про поняття шифрування даних, шифрування зі збереженням формату та маскування даних

Шифрування даних — це спосіб захисту даних шляхом їх кодування таким чином, що їх може розшифрувати або отримати до них доступ лише особа, яка має правильний ключ шифрування.

Дані, які потрібно зашифрувати, називають відкритим текстом або вхідним текстом. Відкритий текст шифрують за допомогою різноманітних алгоритмів шифрування, які в основному є математичними обчисленнями, що виконуються над необробленою інформацією. Існує кілька алгоритмів шифрування, кожен з яких відрізняється послідовними кроками та рівнем безпеки.

Окрім алгоритмів, потрібен також ключ шифрування. За допомогою згаданого ключа та відповідного алгоритму шифрування відкритий текст перетворюється на зашифрований фрагмент даних, також відомий як зашифрований текст. Замість того, щоб надсилати відкритий текст напряму одержувачу, зашифрований текст надсилається через незахищені канали зв’язку.

Коли зашифрований текст досягає призначеного одержувача, він може використати ключ дешифрування, щоб перетворити зашифрований текст назад у читабельний формат, тобто відкритий текст. Цей ключ розшифрування має постійно зберігатися в таємниці та може бути не схожим на ключ, який використовується для шифрування повідомлення.

Існують два види шифрування інформації: симетричний та асиметричний.

Симетричний, також називається криптографією із закритим ключем або алгоритмом секретного ключа. Цей метод вимагає, щоб відправник і одержувач мали доступ до одного ключа. Отже, одержувач повинен мати ключ до того, як повідомлення буде розшифровано. Цей метод найкраще працює для закритих систем, які мають

менший ризик вторгнення третьої сторони. Процес симетричного шифрування відбувається швидко. Однак, у цьому випадку обидві сторони мають переконатися, що ключ надійно зберігається та, наприклад, доступний лише програмному забезпеченню, яке має його використовувати. [26]

Симетричне шифрування (рис.2.1), яке використовує один ключ, краще для даних, що перебувають у спокої. Наприклад, дані, що містяться в базах даних, повинні бути зашифровані, щоб запобігти їх злому або крадіжці. Оскільки ці дані мають бути захищеними лише до тих пір, поки їх не знадобиться отримати в майбутньому, для них не потрібні два ключі, просто один, який надається за допомогою симетричного шифрування. [27]

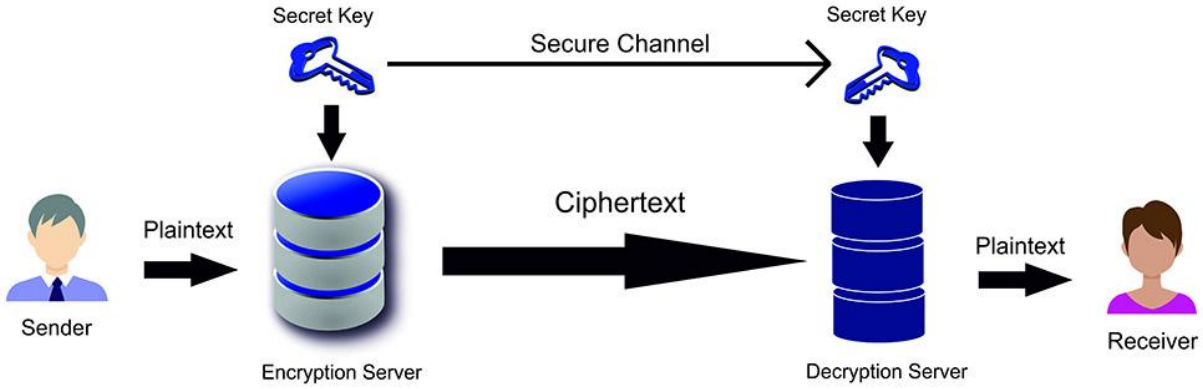


Рисунок 2.1 — Схема симетричного шифрування

Приклади популярних алгоритмів із симетричним ключем включають Twofish, Serpent, AES (Rijndael), Camellia, Salsa20, ChaCha20, Blowfish, CAST5, Kuznyechik, RC4, DES, 3DES, Skipjack, Safer та IDEA. [28]

Асиметричний метод шифрування (рис.2.2), також званий криптографією з відкритим ключем, використовує два ключі для процесу шифрування, відкритий і закритий, які математично пов’язані між собою. Користувач використовує один ключ для шифрування, а інший – для дешифрування.[28]

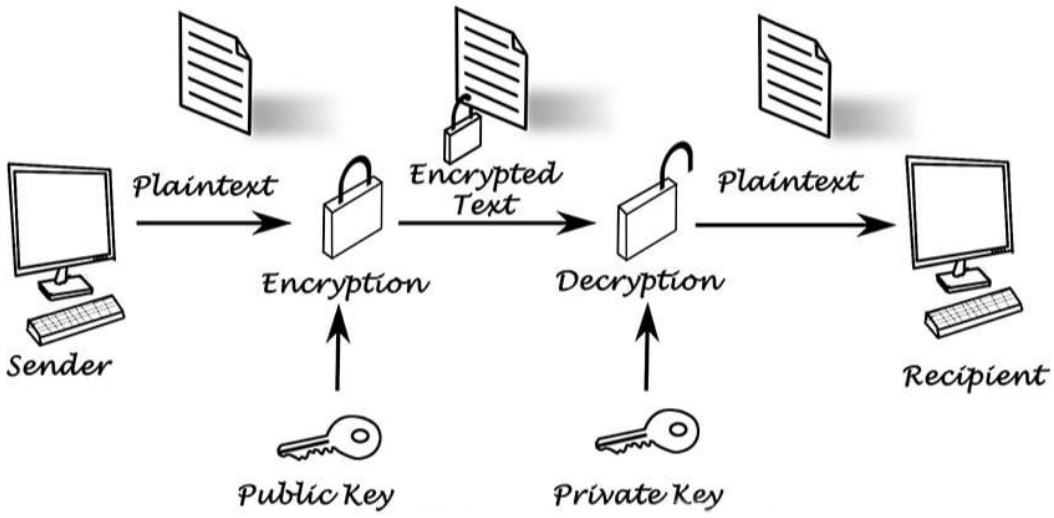


Рисунок 2.2 — Схема асиметричного шифрування

Асиметричне шифрування, слід використовувати для даних, які передаються іншим особам, наприклад, електронною поштою. Якщо для даних в електронних листах використовувалося лише симетричне шифрування, зловмисник може викрасти або скомпрометувати матеріал, отримавши ключ, який використовується для шифрування та дешифрування. Оскільки їхній відкритий ключ використовувався для шифрування даних, відправник і одержувач гарантують, що лише одержувач може розшифрувати дані за допомогою асиметричного шифрування.

Прикладами асиметричного шифрування є алгоритми: Rivest Shamir Adleman (RSA), the Digital Signature Standard (DSS), який включає в себе the Digital Signature Algorithm (DSA), Elliptical Curve Cryptography (ECC), the Diffie-Hellman exchange method. [28]

Шифрування зі збереженням формату (англ. FPE) зберігає формати вхідних доменів після шифрування. Алгоритми FPE мають додаткові параметри tweak t і домен d. Tweak t подібний до криптографічної солі, тобто це додаткова випадковість. Домен d визначає домен, якому належить відкритий текст.

Наприклад, можуть бути домени формату номера телефону або номер кредитної картки, тощо. Основнимий алгоритм схеми шифрування FPE визначений нижче:

1. KeyGen(σ) Генерує ключ k, tweak t на основі а параметр безпеки σ.

2. Шифрування (p, k, t, d): дано відкритий текст p, ключ k і tweak t. алгоритм створює зашифрований текст c таким чином, що відкритий текст p, і зашифрований текст c знаходяться в одному домені d. Як показано на рис. 2.3, він внутрішньо використовує ранг і методи зниження рангу разом із блоком збереження довжини шифр enc' .

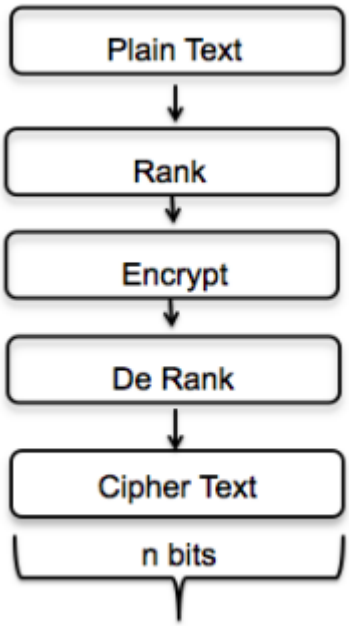


Рисунок 2.3 — Процес шифрування FPE

3. Дешифрування (c, k, t, d): дано зашифрований текст c, ключ k і a tweak t. Алгоритм повертає відповідний вхідний текст. І c, p знаходяться в одному домені. [29]

Стандартизований FPE загального призначення повинен відповідати ряду вимог:

– зашифрований текст має ту саму довжину та формат, що й відкритий текст.

– він повинен бути адаптованим для роботи з різними символами та числами. Приклади включають десяткові цифри, малі літери, і повний набір символів стандартної клавіатури.

– він повинен працювати зі змінною довжиною відкритого тексту.

– рівень безпеки має бути порівнянним із тим, що досягається за допомогою AES.

– захист має бути сильним навіть для дуже невеликої довжини відкритого тексту.

Задовольнити першу вимогу зовсім не просто. Як показано на рисунку 2.4 пряме шифрування за допомогою AES дає 128-бітний двійковий блок що не схожий на необхідний формат. Також стандартний симетричний блок шифр нелегко адаптувати для створення FPE.

	Credit card	Tax ID	Bank account number
Plaintext	8123 4512 3456 6780	219-09-9999	800N2982 K-22
FPE	8123 4521 7292 6780	078-05-1120	709G9242H-35
AES (hex)	af411326466add24 c86abd8aa525db7a	7b9af4f3f218ab25 07c7376869313afa	9720ec7f793096ff d37141242e1c51bd

Рисунок 2.4 — Порівняння шифрувань значень методами FPE та AES

Наприклад, якщо потрібен алгоритм, який може шифрувати рядки десяткових цифр максимальною довжиною 32 цифри, то вхід до алгоритму може зберігатися в 16 байтах (128 біт), кодуючи кожну цифру як чотири біт і використовуючи відповідне двійкове значення для кожної цифри (наприклад, 6 закодовано як 0101). Далі використовується AES для шифрування 128-бітного блоку, як показано нижче:

1. Введення відкритого тексту *X*, який представлений рядком із 4-бітових десяткових цифр *X[1] ... X[16]*. Якщо відкритий текст містить менше 16 цифр, він доповнюється ліворуч нулями.

2. Розглядаючи *X* як 128-бітний двійковий рядок і використовуючи ключ *K*, формується зашифрований текст $Y = AES_K(X)$.

3. Розглядається Y як рядок довжиною 16 з 4-бітових елементів.

4. Деякі входи в Y можуть мати значення більше 9 (наприклад, 1100), тоді, щоб генерувати шифртекст Z у потрібному форматі, необхідно обчислити:

$$Z[i] = Y[i] mod 10, 1 \leq i \leq 16$$

На рисунку 2.5 показано структуру Фейстеля, яка використовується у всіх алгоритмах NIST, де шифрування показано ліворуч, а розшифрування – праворуч. Вхідними даними для алгоритму шифрування є текстовий рядок символів $n = u + v$. Якщо n парне, то $u = v$; інакше u і v відрізняються на 1. Дві частини рядка проходять через парну кількість циклів обробки для створення блоку зашифрованого тексту з n символів того самого формату, що й відкритий текст. Кожен раунд і має входи A_i та B_i, отримані з попереднього раунду (або відкритий текст для раунду 0).

Усі раунди мають однакову структуру. У раундах з парними номерами, заміна виконується в лівій частині (довжина u) даних, A_i. Це зроблено шляхом застосування округлюючої функції F_K до правої частини (довжина v) даних, B_i, а потім виконуємо модульне додавання виведення F_K з A_i. Функція додавання описана далі. На непарних раундах, заміна виконується в правій частині даних. F_K є односторонньою функцією який перетворює вхідні дані у двійковий рядок, виконує перетворення рядка, а потім перетворює результат назад у рядок символів з відповідним форматом і довжиною. Функція має параметри секретного ключа K, довжина відкритого тексту n, параметр (налаштування) tweak T і кругле число i.

Процес дешифрування по суті такий самий, як процес шифрування. Відмінності: (рис. 2.5 - a) функція додавання замінюється на віднімання функція, яка є її оберненою, і (рис. 2.5 - b) порядок круглих індексів є зворотним.

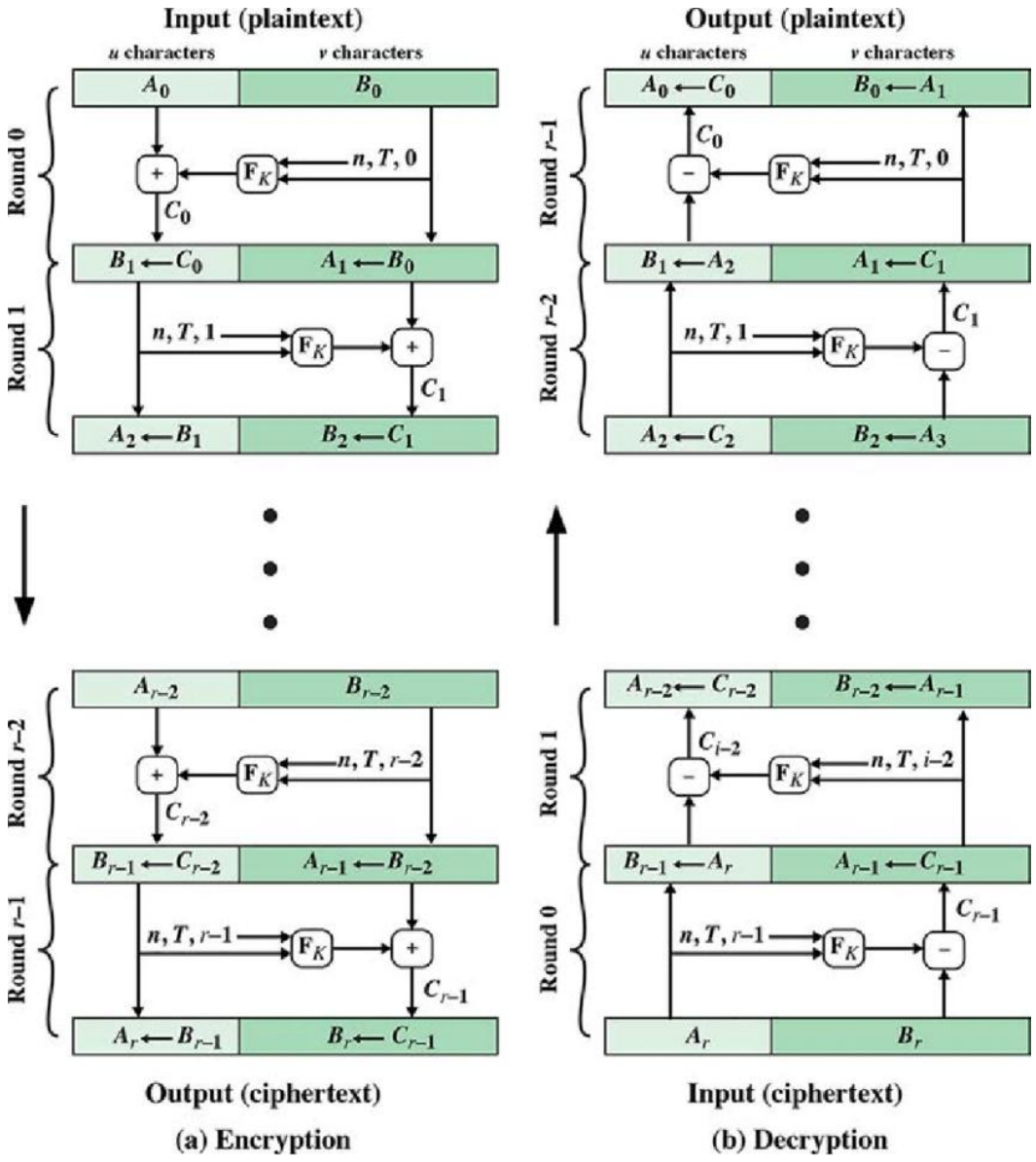


Рисунок 2.5 — Структура Фейстеля

Щоб продемонструвати, що дешифрування дає правильний результат, рис. 2.5 - b показує процес шифрування, що йде вниз зліва, і дешифрування процес, що йде вгору праворуч. Діаграма показує, що в кожному раунді проміжне значення процесу дешифрування дорівнює відповідному значенню процесу шифрування. Ми можемо переглянути схему, щоб підтвердити це, починаючи знизу. Зашифрований текст створюється в кінці раунду r - 1 як рядок for $A_r \| B_r$ де A_r і B_r мають довжини рядків

u і *v* відповідно. Раунд шифрування r - 1 може бути описаний наступними рівняннями:

$$A_r = B_r - 1$$
$$B_r = A_r - 1 + F_K[B_r - 1]$$

Оскільки функція визначена як функцію віднімання тобто, вона обернена до функції додавання, то ці рівняння можна переписати наступним чином:

$$B_r - 1 = A_r$$
$$A_r - 1 = B_r - F_K[B_r - 1]$$

Можна побачити, що останні два рівняння описують дію раунду 0 функції дешифрування, щоб результат раунду 0 дешифрування дорівнював входу раунду r - 1 шифрування. Аналогічно, процес проходить увесь шлях через r ітерацій. [30]

Маскування даних – це спосіб створити фальшиву, але реалістичну версію організаційних даних. Мета полягає в тому, щоб захистити персональні та конфіденційні дані, одночасно забезпечуючи функціональну альтернативу, коли реальні дані не потрібні, наприклад, під час навчання користувачів, демонстрацій продажів або тестування програмного забезпечення.

Процеси маскування даних змінюють значення даних залишаючи їх вхідний самий формат (рис. 2.6). Мета полягає в тому, щоб створити версію, яка не піддається розшифровці або зворотній інженерії. Є кілька способів змінити дані, включаючи перетасування символів, заміну слів або символів і шифрування.

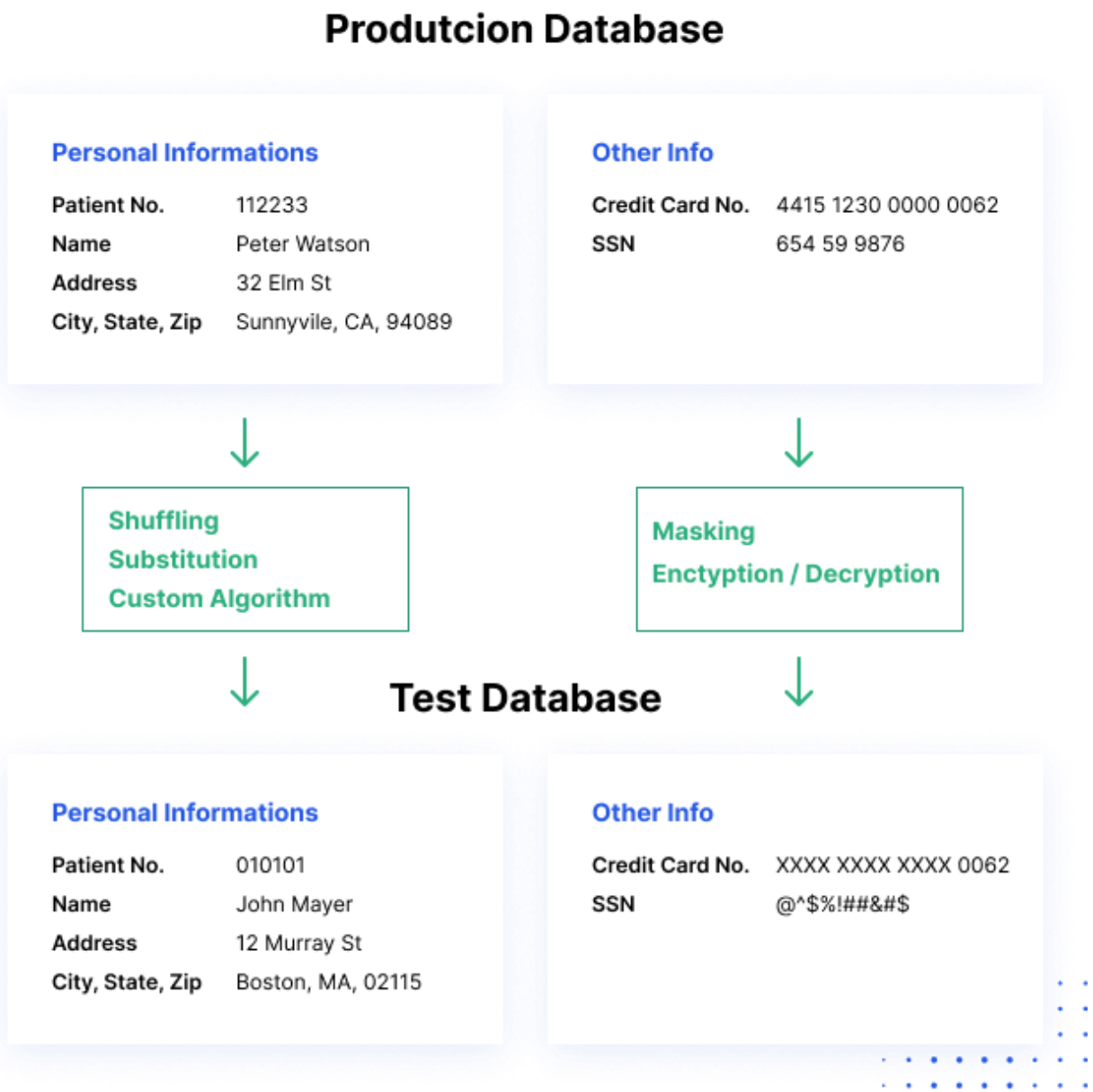


Рисунок 2.6 — Процеси маскування даних при переході з бази продакшин сервера в тестову

Існує кілька типів типів маскування даних, які зазвичай використовуються для захисту конфіденційних даних.

Наприклад, статичне маскування даних. Статичні процеси маскування даних можуть допомогти створити очищену копію бази даних. У процесі змінюються всі конфіденційні дані, доки копію бази даних не можна буде безпечно поділитися. Як правило, процес передбачає створення резервної копії бази даних у робочому стані, завантаження її в окреме середовище, видалення будь-яких непотрібних даних і мас-

кування даних. Потім замасковану копію можна перемістити в цільове розташування.

Детерміноване маскування даних включає зіставлення двох наборів даних, які мають однаковий тип даних, таким чином, що одне значення завжди замінюється іншим значенням. Наприклад, ім'я «Джон Сміт» завжди замінюється на «Джим Джеймсон», скрізь, де воно з'являється в базі даних. Цей метод зручний для багатьох сценаріїв, але за своєю суттю менш безпечний.

Маскування даних на льоту під час їх передачі з виробничих систем до систем тестування або розробки перед тим, як дані будуть збережені на диску. Організації, які часто розгортають програмне забезпечення, не можуть створити резервну копію вихідної бази даних і застосувати маскування — їм потрібен спосіб безперервної потокової передачі даних із продакшина до кількох тестових середовищ. Кожна підмножина замаскованих даних зберігається в середовищі розробки/тестування для використання невиробничою системою. Важливо застосувати маскування на льоту до будь-якого каналу з виробничої системи до середовища розробки на самому початку проекту розробки, щоб запобігти проблемам відповідності та безпеки.

Динамічне маскування даних подібне до маскування на льоту, але дані ніколи не зберігаються у вторинному сховищі даних у середовищі розробки/тестування. Натомість воно передається безпосередньо з робочої системи та споживається іншою системою в середовищі розробки/тестування.

Існує чимало методів маскування даних. Наприклад, перетасовування окремих знаків у значенні даних. Якщо існує, ідентифікаційний номер, такий як 76498 у робочій базі даних, можна замінити на 84967 у тестовій базі даних. Цей метод дуже простий у реалізації, але його можна застосувати лише до деяких типів даних і він менш безпечний.

Наступним методом є обнулення. Під час перегляду неавторизованим користувачем дані виглядають відсутніми або «нульовими». Але це робить дані менш корисними для розробки та тестування.

Також, існує метод варіації значень. Вихідні значення даних замінюються функцією, наприклад різницею між найменшим і найвищим значенням у ряді. Наприклад, якщо клієнт придбав кілька продуктів, ціну покупки можна замінити діапазоном між найвищою та найнижчою сплаченою ціною. Це може надати корисні дані для багатьох цілей, не розкриваючи вихідний набір даних.

В методі перестановки, за винятком того, що значення даних змінюються в межах одного набору даних. Дані переставляються в кожному стовпці за допомогою випадкової послідовності; наприклад, перемикання між реальними іменами клієнтів у кількох записах клієнтів. Вихідний набір виглядає як реальні дані, але він не відображає справжню інформацію для кожної особи чи запису даних.

Метод псевдонімізації, як визначено в GDPR, — це будь-який метод, який гарантує, що дані не можна використовувати для ідентифікації особистості. Це вимагає видалення прямих ідентифікаторів і, бажано, уникнення кількох ідентифікаторів, які в поєднанні можуть ідентифікувати особу. Крім того, ключі шифрування або інші дані, які можна використовувати для повернення до початкових значень даних, слід зберігати окремо та надійно.

Дуже важливо також знати, як захистити алгоритми створення даних, а також альтернативні набори даних або словники, які використовуються для шифрування або дешифрування даних. Оскільки лише авторизовані користувачі повинні мати доступ до справжніх даних, ці алгоритми слід вважати надзвичайно чутливими. [31]

2.2 Застосування удосконаленого методу для псевдонімізації полів типу ім’я, прізвище та email адреса

Під час аналізу продуктів, що мають імплементоване рішення для шифрування даних зі збереженням формату та псевдонімізації даних, були виявлені лише ознаки простого шифрування або базового маскування. На мою думку виконані рішення, які запропоновані на ринку слабо підходять під загальну концепцію псевдонімізації да-

них. Для того, щоб досягти цілі чіткої псевдонімізації, щоб факт підміни реальних даних був непомітним, вирішено модернізувати підхід до маскування та шифрування для полів типу ім’я, прізвище та email адреса.

Для псевдонімізації поля “ім’я” потрібно замінювати реальне ім’я користувача на інше ім’я, яке буде схожим на інше реальне ім’я. Щоб виконати поставлену задачу, необхідно створити словник з унікальними людськими (в межах словника) іменами, достатньо великий, щоб алгоритм випадково вибраного числа міг обрати ім’я за його порядковим номером для подальшої заміни справжнього імені користувача. Потім в окремий словник з ключами для імен буде записано реальне ім’я користувача (значення) і порядковий номер фейкового імені (ключ) для подальшого процесу де-псевдонімізації. Наприклад: якщо за алгоритмом було згенероване число 23, то у словнику з фейковими іменами шукаємо ім’я за порядковим номером 23. Заміняємо реальне ім’я користувача на псевдонім “Jonathan” та зберігаємо реальне ім’я окремо в словник ключів поряд з згенерованим числом 23.

Процес де-псевдонімізації буде відбуватись у зворотньому напрямку. Підгружаючи файл із зашифрованими та замаскованими даними, програма розділить файл на колонки і зчитає значення в колонці “ім’я”, і знайде значення “Jonathan” у словнику фейкових імен. Відповідно до знайденого імені, за алгоритмом буде взято порядковий номер цього імені у словнику фейкових імен та потім у словнику ключів для імен за ключем порядкового номеру, числа 23, знайдемо реальне ім’я користувача, яке буде значенням у цьому словнику для ключа 23. Таким чином, частина де-псевдонімізації поля “ім’я” буде завершена.

Такий же процес буде і для псевдонімізації поля “прізвище”, де потрібно замінювати реальне прізвище користувача на інше прізвище, яке буде схожим на інше реальне. Для виконання цієї задачі, знову ж таки необхідно створити словник з унікальними людськими прізвищами (в межах словника), достатньо великий, щоб алгоритм випадково вибраного числа міг обрати прізвище за його порядковим номером для подальшої заміни справжнього прізвища користувача. Потім в окремий сло-

вник з ключами для прізвищ буде записано реальне прізвище користувача і порядковий номер фейкового прізвища для подальшого процесу де-псевдонімізації. Наприклад: якщо за алгоритмом було згенероване число 178, то у словнику з фейковими прізвищами шукаємо прізвище за порядковим номером 178. Заміняємо реальне прізвище користувача на псевдонім “Richardson” та зберігаємо реальне прізвище окремо в словник ключів для прізвищ поряд із згенерованим числом 178.

Процес де-псевдонімізації для прізвища буде відбуватись у зворотньому напрямку. Підгружаючи файл із зашифрованими та замаскованими даними, програма розділить файл на колонки і зчитає значення в колонці “прізвище”, і знайде значення “Richardson” у словнику фейкових прізвищ. Відповідно до знайденого прізвища, за алгоритмом буде взято порядковий номер цього прізвища у словнику фейкових прізвищ та потім у словнику ключів для прізвищ за ключем порядкового номеру, числа 178, знайдемо реальне прізвище користувача, яке буде значенням у цьому словнику для ключа 178. Таким чином, частина де-псевдонімізації поля “прізвище” буде завершена.

Щодо удосконаленого методу псевдонімізації для поля “email” адреси, то процес передбачає відокремлення домену (domain) адреси від частини юзернейму (local-part) користувача, тому що маскувати чи шифрувати домен email адреси частіше за все не є виправданим. Процес псевдонімізації email адреси буде спиратися на замасковані ім’я та прізвище, щоб у вихідному (замаскованому і зашифрованому) датасеті дані виглядали як справжні. Для цього візьмемо першу букву замаскованого імені і ціле замасковане прізвище та поєднаємо їх через крапку. Наприклад, для замаскованого юзера з ім’ям Jonathan та прізвищем Richardson частина юзернейма після алгоритму об’єднання буде виглядати наступним чином: J.Richardson. Наступним кроком буде частина генерації 8 значного випадкового числа (наприклад, 74925207), і приєднання цього номеру до згенерованої частини юзернейму email адреси. Також, це 8 значне згенероване число (74925207) буде служити ідентифікатором, тобто ключем, для зберігання реального юзернейму email адреси як значення у словнику ключів для

email адрес для їх подальшої де-псевдонімізації. Тобто, замаскований юзернейм email адреси виглядатиме наступним чином: J.Richardson74925207. Потім до згенерованого юзернейму додається символ @ та домен реальної email адреси і це буде замасковане значення email адреси.

Процес де-псевдонімізації для поля email адреси виглядатиме наступним чином: підгружаючи файл із зашифрованими та замаскованими даними, програма розділить файл на колонки і зчитає значення в колонці “email” адрес, і знайде значення “J.Richardson74925207@gmail.com”. Згідно подальших кроків, алгоритм зіскануює значення email адреси і виявить у ній 8-значне число (74925207), потім у словнику ключів для email адрес буде виконано пошук по ключам і для знайденого ключа 74925207 поряд буде значення реального юзернейму email адреси. Наступним кроком стане приєднання (через символ @) незамаскованого домену email адреси до знайденого значення (реального юзернейму) у словнику ключів для email адрес. Таким чином, частина де-псевдонімізації поля “email” адреса буде завершена.

2.3 Імплементація шифрування зі збереженням даних в полях типу номер телефону та кредитної карти

Щодо аналізу продуктів, що мають імплементоване рішення для шифрування даних зі збереженням формату та їх псевдонімізації стосовно полів мобільних номерів та номерів кредитних карт, то були виявлені лише ознаки шифрування цілого значення або базового маскування таких значень за допомогою маскувальних символів “*”, “#” або “0”. На мою думку виконані рішення, які запропоновані на ринку слабо підходять під загальну концепцію псевдонімізації даних. Для того, щоб досягти цілі чіткої псевдонімізації, щоб факт підміни реальних даних був непомітним, вирішено модернізувати підхід до маскування та шифрування для полів типу номер телефону та кредитної карти.

Для повноцінного маскування та шифрування зі збереженням формату для поля “номер телефону” було обрано наступний алгоритм дій. За основу було взято номер телефону українського формату, де можливі два типи збереження з кодом країни (+380) та без нього. Оскільки в табличних даних, де зберігаються номери телефонів, (в csv файлах) частіше за все, не підтримується форматування для цифрових значень, що починається з 0 або з +, було вирішено не враховувати шифрування таких випадків. А отже, маємо значення довжиною в 12 і 9 символів відповідно.

Для того, щоб забезпечити гарантовано надійне маскування поля номера телефону і 12 символів, було вирішено відділити частину номера телефону, що є кодом країни (2 символа) і також код мобільного оператора (3 символа), тобто, перші 5 символів і не піддавати їх шифруванню, а шифрувати методом зі збереженням формату FPE в режимі FF3-1. Тобто, наприклад, коли програма зчитує мобільний номер з поля у вигляді “380671234567”, то значення “38067” записується як окреме значення і зберігається окремо, а значення “1234567” буде шифруватись і перетворюватись у інше цифрове значення такої ж довжини (7 цифр). Після того до зашифрованого значення приєднуємо значення з кодом країни і кодом оператора і зберігаємо як повноцінне зашифроване і замасковане значення. Під час шифрування методом зі збереженням формату FPE в режимі FF3-1 нам потрібні два значення key та tweak, які будуть збережені у словнику для ключів мобільних телефонів. При розшифруванні береться номер по порядку і проходиться по словнику ключів розшифровуючи значення за значенням, таким чином відбувається процес розшифрування.

Для шифрування поля номера кредитної або дебетової картки потрібно знати специфіку розмежування даних, які він в собі містить.

Часто номер картки ще називають «довгим номером», який зазвичай складається з 16 цифр, але в деяких випадках може містити до 19 цифр. Більш офіційно, він відомий як постійний номер рахунку або «PAN».

Однак це не просто випадкове число. Номер кредитної картки є унікальним і містить інформацію, яка використовується для ідентифікації клієнтського рахунку, картки та того, хто її видав.

Перша цифра вказує на провайдера:

- номери Mastercard починаються з 2 або 5;
- номери карток Visa починаються з 4.

Перші 6 цифр допомагають ідентифікувати емітента картки, відомі як номер ідентифікатора випуску або «IIN». [32] Банки-емітенти призначають цей номер своїм окремим клієнтам. Кожен банк-емітент має близько трильйона потенційних номерів рахунків.[33]

Усі наступні числа, які стосуються конкретного облікового запису, за винятком останнього, відомого як «контрольна цифра». Це допомагає перевірити, чи надано повний номер кредитної картки в правильному порядку щоразу, коли користувач робить покупку, платіж чи переказ.

Цей метод створення номерів кредитних карток використовується в усьому світі та був винайдений інженером IBM Гансом Петером Луном у 1954 році. [32]

Емітенти кредитних карток і мережі використовують математичні інструменти для боротьби з витоком даних та іншими шахрайськими діями. Алгоритм Luhn або mod 10 є одним із таких. Розроблений у 1960-х роках, він використовує ідентифікаційні цифри, такі як номери соціального страхування та кредитної картки, щоб визначити дійсність.

Кредитні картки призначені для миттєвого використання. Ось чому процес перевірки, який використовують банки, повинен негайно шифрувати та розшифровувати конфіденційні дані. Тут на допомогу приходить алгоритм Luhn. За його допомогою можна легко перевірити номери карток і підтвердити їх дійсність.

Алгоритм Luhn простий у використанні. При додаванні номера чека до решти чисел на картці сума повинна дорівнювати 0. Якщо буде введене неправильне число

під час онлайн-покупки, воно буде виявлено відразу, оскільки сума не буде дорівнювати 0.

Visa використовує цифру 13 як контрольну суму в більшості випадків, тоді як інші великі мережі використовують останню цифру.[33]

Зазвичай, люди не вникають у глибокий аналіз номерів карток. Для них достатньо всього побачити 16 цифр щоб ідентифікувати, що це саме номер картки. Тому було вирішено шифрувати усі 16 цифр картки, щоб уникнути можливих ризиків методом зі збереженням формату FPE в режимі FF3-1. Тобто, наприклад, коли програма зчитує поле “credit card” і дані карток з нього, то шифрування зі збереженням формату буде виконуватись над 16 цифрами, де в результаті шифрування на виході буде інше зашифроване 16-значне значення.

Під час шифрування методом зі збереженням формату FPE в режимі FF3-1 нам потрібні два значення key та tweak, які будуть збережені у словнику для ключів кредитних карт. При розшифруванні береться номер кредитної карти по порядку і проходиться по словнику ключів розшифровуючи значення за значенням, таким чином відбувається процес розшифрування.

2.4 Імплементація захисту для словників даних за допомогою AES (режим CBC) з ключем шифрування в 128-біт

Розширений стандарт шифрування (англ. Advanced Encryption Standard - AES) — це специфікація для шифрування електронних даних, створена Національним інститутом стандартів і технологій США (NIST) у 2001 році. AES широко використовується сьогодні, і його ключові характеристики наступні:

- AES - це блоковий шифр;
- розмір ключа може бути 128/192/256 біт;
- шифрує дані блоками по 128 біт кожен.

Це означає, що алгоритм приймає 128 біт як вхід і виводить 128 біт зашифрованого шифрованого тексту як вихід. AES базується на принципі мережі заміни-перестановки, що означає, що він виконується за допомогою серії пов'язаних операцій, які включають заміну та перемішування вхідних даних.[34]

AES є ітеративним, а не шифром Фейстеля який використовує алгоритм FPE. Він заснований на «мережі заміни-перестановки». Він складається з серії пов'язаних операцій, деякі з яких передбачають заміну вхідних даних на певні виходи (підстановки), а інші передбачають перетасування бітів (перестановки).

Цікаво, що AES виконує всі свої обчислення на байтах, а не на бітах. Отже, AES розглядає 128 біт блоку відкритого тексту як 16 байт. Ці 16 байтів розташовані в чотирьох стовпцях і чотирьох рядках для обробки як матриця. Кількість раундів в AES є змінною і залежить від довжини ключа. AES використовує 10 раундів для 128-бітних ключів, 12 раундів для 192-бітних ключів і 14 раундів для 256-бітних ключів. Кожен із цих раундів використовує інший 128-бітний раундовий ключ, який обчислюється з оригінального ключа AES.

Авторами AES є Морріс Дворкін, Елейн Баркер, Джеймс Нечватал, Джеймс Фоті, Лоуренс Бассам, Едвард Робак і Джеймс Дрей молодший, був опублікований 26 листопада 2001 року та прийнятий урядом США в 2002 році. Алгоритм AES, зокрема, також відомий як Rijndael [35] , будучи похідним від сімейства алгоритмів шифрування Rijndael, розробленого бельгійськими криптографами Вінсентом Райменом і Джоан Демен.

Головною причиною розробки AES була заміна старого, застарілого та вразливого стандарту шифрування даних (DES), який на той момент був золотим стандартом алгоритму симетричного шифрування для державного та приватного секторів протягом 20 років. По суті, DES починав ставати все більш вразливим до атак грубої сили, і NIST потребував новішого та безпечнішого золотого стандарту.

Під час трирічного конкурсу NIST алгоритми надсилали криптографи з усього світу, і врешті NIST звузило список до п’яти найкращих: MARS, RC6, Rijndael, Serpent і Twofish.

У жовтні 2000 року було оголошено, що Rijndael став переможцем конкурсу. NIST вибрав Rijndael над іншими фіналістами, тому що, згідно з NIST, Rijndael мав найкраще поєднання безпеки, продуктивності, ефективності, можливості реалізації та гнучкості. [36]

Загальновідомим фактом є те, що алгоритм AES все ще неможливо зламати, принаймні в цьому житті. Суперкомп’ютеру знадобляться мільярди – років, щоб зламати навіть 128-бітний ключ AES. Квантові комп’ютери можуть зламати алгоритми AES швидше, але, згідно з деякими джерелами [37] , квантовому комп’ютеру все одно знадобиться приблизно шість місяців, щоб вичерпати можливості 128-бітного ключа AES.

З моменту публікації алгоритми AES через свою фактичну непроникність для атак грубої сили став криптографічним золотим стандартом у всьому світі для надійного шифрування та запобігання несанкціонованому доступу до електронних даних, включаючи, але не обмежуючись, конфіденційну інформацію, контрольовану несекретну інформацію (CUI) [28], та секретну інформацію.

Алгоритм AES використовує 128-бітний симетричний або одноключовий блоковий шифр, який шифрує та розшифровує інформацію. Процес шифрування AES створює зашифрований текст, який є нерозбірливим, фактично нерозбірливим перетворенням відкритих текстових даних, версією інформації, яку люди можуть прочитати та зрозуміти. Вихідні дані процесу шифрування, тобто зашифрований AES текст, не можна прочитати, доки для його розшифровки не буде використано секретний ключ AES.[39]

Процеси шифрування та дешифрування можуть використовувати ключі довжиною 128, 192 та 256 біт для перетворення відкритого тексту в зашифрований текст

і зашифрованого тексту у відкритий текст. Ці процеси відомі як шифрування та дешифрування відповідно.

У комунікаціях, зашифрованих за допомогою AES, відправнику та одержувачу надається однаковий секретний ключ AES, який використовується для перетворення інформації в зашифрований текст, а також у читабельний відкритий текст. Якби цю інформацію перехопив хакер, прочитати її без секретного ключа AES було б неможливо, крім користувачів, які надсилають і отримують зашифровану інформацію.

Опис алгоритму AES:

1. Розширення ключа , яке створює нові ключі, відомі як круглі ключі, для кожного наступного раунду шифрування, використовуючи розклад ключів Rijndael.
2. Додавання раундового ключа, під час якого початковий раундовий ключ додається до суміші даних, які було розділено.
3. Заміна байтів , яка замінює кожен байт іншим байтом на основі поля підстановки Rijndael S-box.
4. Зсув рядка , який переміщує кожен рядок розділених даних на один проміжок ліворуч для другого рядка, два проміжки ліворуч для третього рядка та три пробіли ліворуч для четвертого рядка.
5. Змішування стовпців , яке використовує попередньо встановлену матрицю для множення стовпців розділених даних і створення нового блоку коду.
6. Додавання круглого ключа , під час якого до суміші стовпців додається ще один круглий ключ.

Після цього початкового раунду процес повторюється дев’ять, 11 або 13 разів, залежно від того, чи використовує алгоритм AES ключ довжиною 128 бітів, 192 біти чи 256 бітів. 128-бітове шифрування AES проходить 10 раундів трансформації; 192-бітове шифрування AES проходить 12 раундів трансформації; і 256-бітне шифрування AES проходить 14 раундів трансформації. Наведені вище кроки становлять один

раунд, тому для 128-бітного шифрування AES, 192-бітного шифрування AES і 256-бітного шифрування AES залишилося дев’ять, 11 і 13 раундів відповідно.

Шифрування AES також можна поєднувати з іншими криптоалгоритмами NIST, щоб покращити та посилити захист конфіденційної або секретної інформації компанії чи організації, створюючи непроникний криптографічний коктейль, який протистоїть кібератакам зловмисників, які намагаються отримати доступ до конфіденційної та секретної інформації.

Оскільки після шифрування та процесу псевдонімізації даних були створені 5 словників з ключами відповідно до кожного з полів даних, для їх наступного розшифрування, то виникає питання безпеки саме цих словників, адже вони не можуть зберігатись у відкритому вигляді, оскільки це просто нерозумно.

Для процесу шифрування словників з ключами для полів ім’я, прізвище, email адреса, номер телефону та поля номера кредитних карт було використане 128-бітне шифрування AES в режимі CBC. Після їх шифрування користувачу потрібно зберегти 128 бітний ключ у надійному місці, бо без його наявності повернути дані у вхідний формат буде неможливо. Файл з 128 бітним ключем можна зберігати у надійному Password менеджері або Password Vault-і.

128-бітне шифрування AES відноситься до процесу приховування даних відкритого тексту за допомогою ключа AES довжиною 128 біт. 128-бітне шифрування AES використовує 10 раундів трансформації для перетворення відкритого тексту в зашифрований текст і схвалено Агентством національної безпеки (АНБ) для захисту секретної інформації.

128-бітне шифрування AES також може стосуватися фіксованого розміру блоку алгоритму шифрування AES загалом. Хоча довжина ключів AES – 128, 192 і 256 біт – може змінюватися, розмір блоку даних, зашифрованих за допомогою AES, завжди становить 128 біт.

Зі 128-бітного, 192-бітного та 256-бітного шифрування AES, які поступово використовують більше раундів шифрування для підвищення безпеки, 128-бітне шиф-

рування AES є технічно найменш безпечним. Однак це не означає, що 128-бітне шифрування AES не є безпечним або чудовим вибором для шифрування ваших даних; пам’ятайте, щоб зламати навіть 128-бітний ключ AES, знадобляться мільярди років. Крім того, він використовується для шифрування секретної урядової інформації, тому на даний момент він є непроникним.

3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА РЕАЛІЗАЦІЯ УДОСКОНАЛЕНОГО МЕТОДУ ПСЕВДОНІМІЗАЦІЇ ДАНИХ ТА МЕТОДУ ШИФРУВАННЯ ЗІ ЗБЕРЕЖЕННЯМ ФОРМАТУ ДО ПОЛІВ ДАНИХ У CSV ФАЙЛАХ

3.1 Вибір мови та середовища програмування

Для написання програмного продукту для шифрування даних зі збереженням формату та їх псевдонімізації було використано мову Python.

Python — це інтерпретована об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою, розроблена Гвідо ван Россумом. Спочатку він був випущений у 1991 році. Розроблений, щоб бути легким, а також веселим, назва "Python" є реверансом до британської комедійної групи Monty Python. Python має репутацію мови, зручної для початківців, замінюючи Java як найпоширенішу початкову мову, оскільки вона справляється з більшою частиною складності для користувача, дозволяючи початківцям зосередитися на повному розумінні концепцій програмування, а не на найменших деталях.

Python використовується для веб-розробки на стороні сервера, розробки програмного забезпечення, математики та системних сценаріїв, і популярний для швидкої розробки додатків, а також як мова сценаріїв або з’єднувальна мова для зв’язування існуючих компонентів завдяки своїм високорівневим вбудованим структурам даних, динамічний тип і динамічне зв'язування. Витрати на технічне обслуговування програми знижуються завдяки легкому освоєнню синтаксису та акценту на зручності читання. Крім того, підтримка додаткових модулів і пакетів полегшує модульні програми та повторне використання коду. Python є мовою спільноти з відкритим вихідним кодом, тому багато незалежних програмістів постійно створюють бібліотеки та функціональність для неї.[40]

Python підтримує модулі та пакети, що заохочує модульність програми та повторне використання коду. Інтерпретатор Python і велика стандартна бібліотека дос-

тупні у вихідному або двійковому вигляді безкоштовно для всіх основних платформ і можуть вільно поширюватися.

Багато програмістів використовують мову Python для розробки, тому що вона забезпечує підвищену продуктивність. Оскільки етапу компіляції немає, цикл редагування-тестування-налагодження відбувається неймовірно швидко. Налагоджувати програми на Python легко: помилка чи неправильний вхід ніколи не спричинить помилку сегментації. Натомість, коли інтерпретатор виявляє помилку, він викликає виняток. Якщо програма не вловлює виняток, інтерпретатор друкує трасування стека. Налагоджувач рівня вихідного коду дозволяє перевіряти локальні та глобальні змінні, оцінювати довільні вирази, встановлювати контрольні точки, покроково виконувати код по рядках і так далі. Сам налагоджувач написаний на Python, що свідчить про інтроспективну силу Python.[41]

Випадки використання Python:

- створення веб-додатків на сервері;
- створення робочих процесів, які можна використовувати в поєднанні з програмним забезпеченням;
- підключення до систем баз даних;
- читання та редагування файлів;
- виконання складної математики;
- обробка великих даних;
- швидке створення прототипів;
- розробка готового програмного забезпечення.

У професійному плані Python чудово підходить для серверної веб-розробки, аналізу даних, штучного інтелекту та наукових обчислень. Розробники також використовують Python для створення інструментів продуктивності, ігор і настільних програм.

Особливості та переваги Python:

– сумісний із різними платформами, включаючи Windows, Mac, Linux, та інші;

– використовує простий синтаксис, який можна порівняти з англійською мовою, що дозволяє розробникам використовувати менше рядків, ніж інші мови програмування;

– працює на системі інтерпретатора, яка дозволяє коду виконуватися негайно, швидке відстеження прототипування;

– можна обробляти процедурним, об’єктно-орієнтованим або функціональним способом.

Синтаксис Python:

– дещо схожий на англійську мову, з математичним впливом, Python створений для зручності читання;

– на відміну від інших мов, які використовують крапку з комою та/або круглі дужки для завершення команди, Python використовує нові рядки для тієї самої функції;

– визначає область (тобто цикли, функції, класи) за допомогою відступів, використовуючи пробіли, а не фігурні дужки.

Мова Python, динамічно типізована мова, є особливо гнучкою, усуває жорсткі правила для побудови функцій і пропонує більшу гнучкість вирішення проблем за допомогою різноманітних методів. Це також дозволяє користувачам компілювати та запускати програми аж до проблемної області, оскільки використовує перевірку типу під час виконання, а не під час компіляції.[41]

Щодо середовища програмування, то для створення додатку для шифрування даних зі збереженням формату та їх псевдонімізації було використано PyCharm.

PyCharm — це гібридна платформа, розроблена JetBrains як IDE для Python. Вона використовується для розробки додатків Python. Деякі з “організацій єдинорога”, такі як Twitter, Facebook, Amazon і Pinterest, використовують PyCharm як своє середовище розробки Python.

PyCharm доступний на платформах Windows, Linux або Mac OS. Крім того, він містить модулі та пакети, які допомагають програмістам розробляти програмне забезпечення за допомогою Python за менший час і з мінімальними зусиллями. Також, його також можна налаштувати відповідно до вимог розробників. Нижче наведені деякі функції, що пропонує PyCharm.

Інтелектуальний редактор коду:

- допомагає писати високоякісний код;
- він складається з кольорових схем для ключових слів, класів і функцій. Це допомагає підвищити читабельність і розуміння коду;
- допомагає легко й швидко виявляти помилки;
- він забезпечує функцію автозаповнення та інструкції для завершення коду.

Навігація по коду:

- допомагає розробникам редагувати та вдосконалювати код із меншими зусиллями та часом;
- за допомогою навігації по коду розробник може легко перейти до функції, класу або файлу;
- програміст може швидко знайти елемент, символ або змінну у вихідному коді;
- крім того, використовуючи режим лінзи, розробник може ретельно перевірити та налагодити весь вихідний код.

Рефакторинг:

- перевагою є внесення ефективних і швидких змін як до локальних, так і до глобальних змінних;
- рефакторинг у PyCharm дозволяє розробникам покращувати внутрішню структуру без зміни зовнішньої продуктивності коду;
- це також допомагає розділяти більш розширені класи та функції за допомогою методу екстракту.

Підтримка багатьох інших веб-технологій:

- він допомагає розробникам створювати веб-програми на Python;
- розробники мають можливість редагувати в реальному часі за допомогою PyCharm IDE. Водночас вони можуть попередньо переглянути створену/оновлену веб-сторінку;
- розробники можуть стежити за змінами безпосередньо у веб-браузері;
- PyCharm також підтримує AnglularJS і NodeJS для розробки веб-додатків.

Підтримка популярних веб-фреймворків Python:

- PyCharm підтримує такі веб-фреймворки, як Django;
- він надає функцію автозаповнення та пропозиції щодо параметрів Django;
- це допомагає в налагодженні кодів Django;
- він також допомагає web2py та Pyramid, іншим популярним веб-фреймворкам.

Підтримка для наукових бібліотек Python:

- PyCharm підтримує наукові бібліотеки Python, такі як Matplotlib, NumPy і Anaconda;
- ці наукові бібліотеки допомагають створювати проекти Data Science і Machine Learning;
- він складається з інтерактивних графіків, які допомагають розробникам зрозуміти дані;
- він здатний інтегруватися з різними інструментами, такими як IPython, Django та Pytest. Ця інтеграція допомагає створювати інноваційні унікальні рішення.[42]

3.2 Реалізація графічного інтерфейсу

Для програмної реалізації візуальної частини додатку для шифрування даних зі збереженням формату та їх псевдонімізації було використано один з модулів Python під назвою Tkinter.

Python має багато фреймворків графічного інтерфейсу, але Tkinter є єдиним фреймворком, вбудованим у стандартну бібліотеку Python.

Tkinter має кілька сильних сторін. Це кросплатформенний код, тому той самий код однаково добре працює в Windows, macOS і Linux. Візуальні елементи відображаються за допомогою власних елементів операційної системи, тому програми, створені за допомогою Tkinter, виглядають так, ніби належать платформі, на якій вони запускаються. Також, Tkinter легкий і відносно безболісний у використанні порівняно з іншими фреймворками. Це робить його влучним вибором для створення програм графічного інтерфейсу користувача на Python, особливо для програм, де сучасний блиск непотрібний, і головним пріоритетом є швидке створення чогось функціонального та кросплатформного.[44]

Отже, оскільки розробка велась на операційній системі Ubuntu, тому форма вікна відрізняється від того, що звикли бачити користувачі операційної системи Windows.

Першим кроком після запуску програми користувачу пропонується вибрати опцію з якою він хоче працювати, це може бути як шифрування так і дешифрування файлу (рис. 3.1).

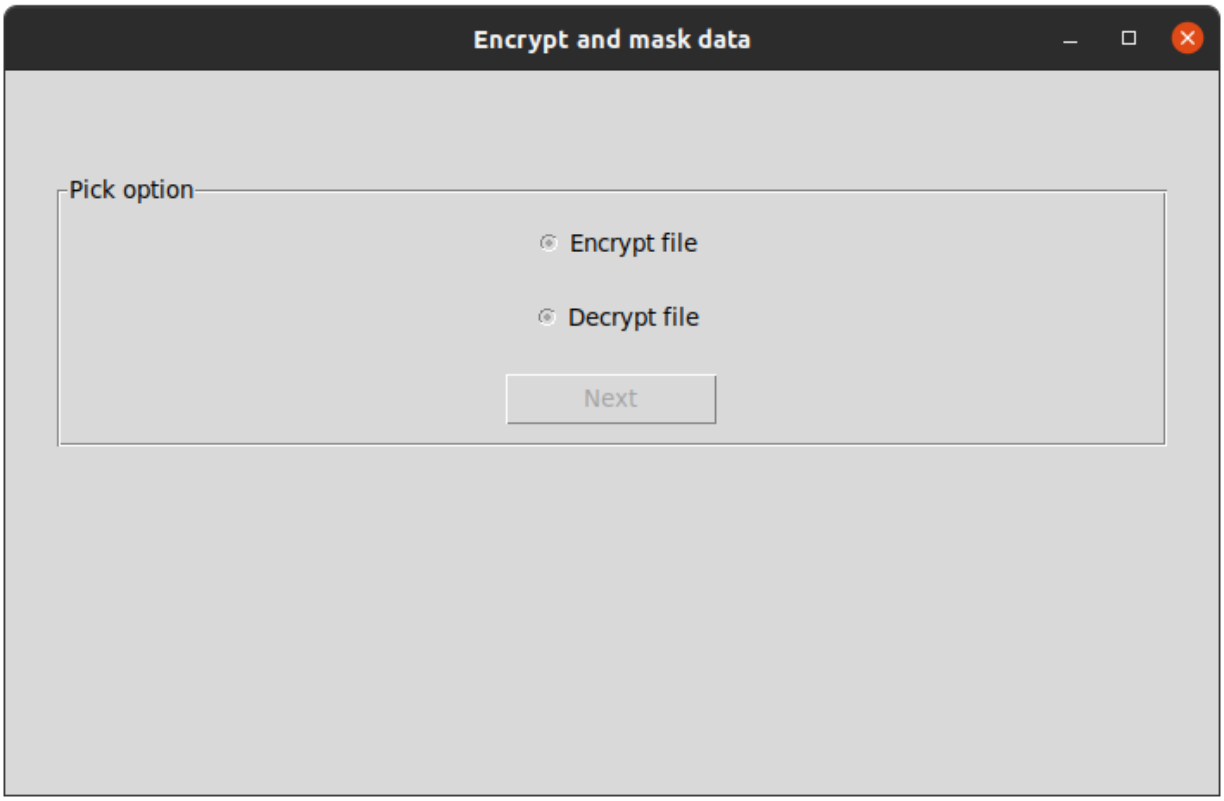


Рисунок 3.1 — Вікно програми при запуску

Почнемо з опції шифрування, коли користувач клікає на radiobutton “Encrypt file” кнопка “Next” стає активною (Рис. 3.2).

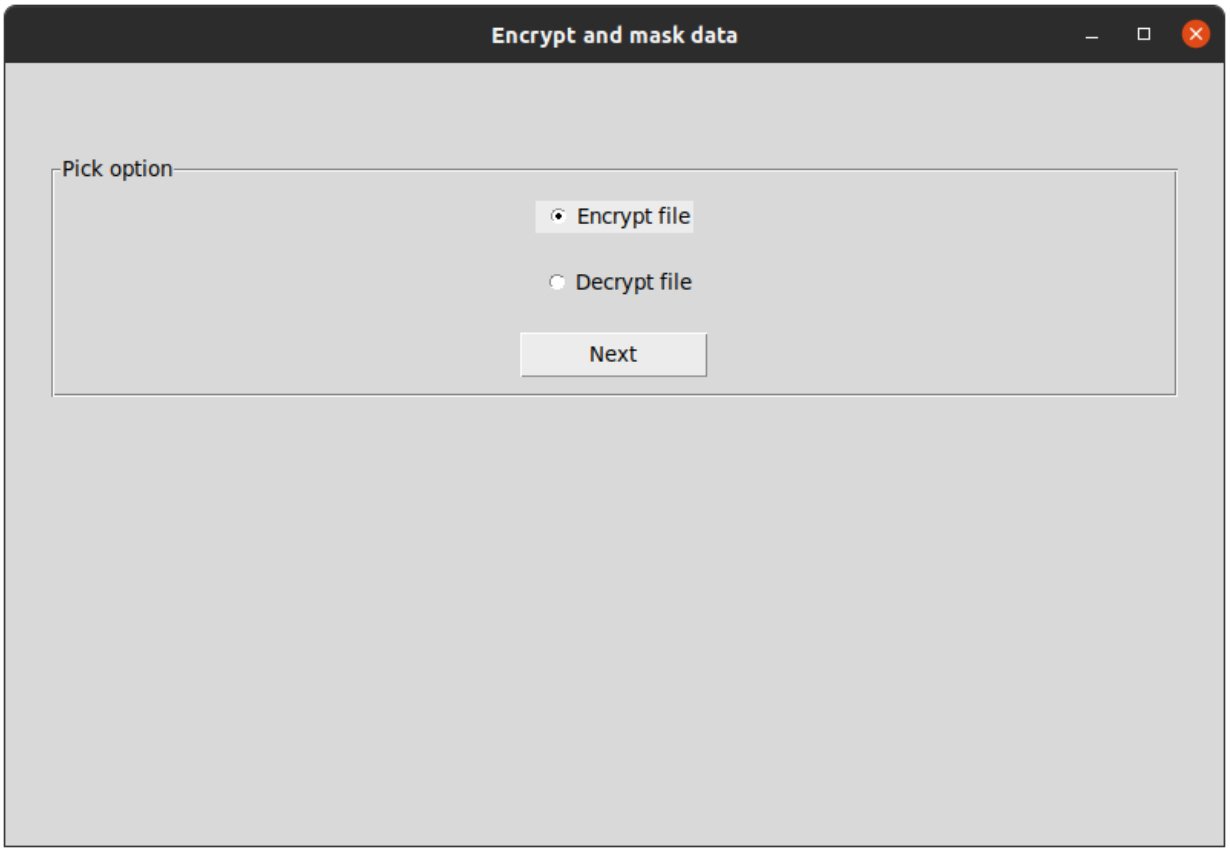


Рисунок 3.2 - Вікно програми при натисненні “Encrypt file”

При її натисканні з'являються інші віджети (опції) для подальшої процедури шифрування файлу. У labelframe-і “Choose file delimiter” користувачу пропонується обрати вид розділювача, який може бути специфічним його табличному-like файлу .csv (comma separated value). Із реалізованих деліметрів (розділювачів) є опція коми, табу, двокрапки та іншого розділювача, який юзер може ввести самостійно. При виборі опції “Other” поле вводу для символу стане активним для подальшого введення потрібного деліметра (рис. 3.3).

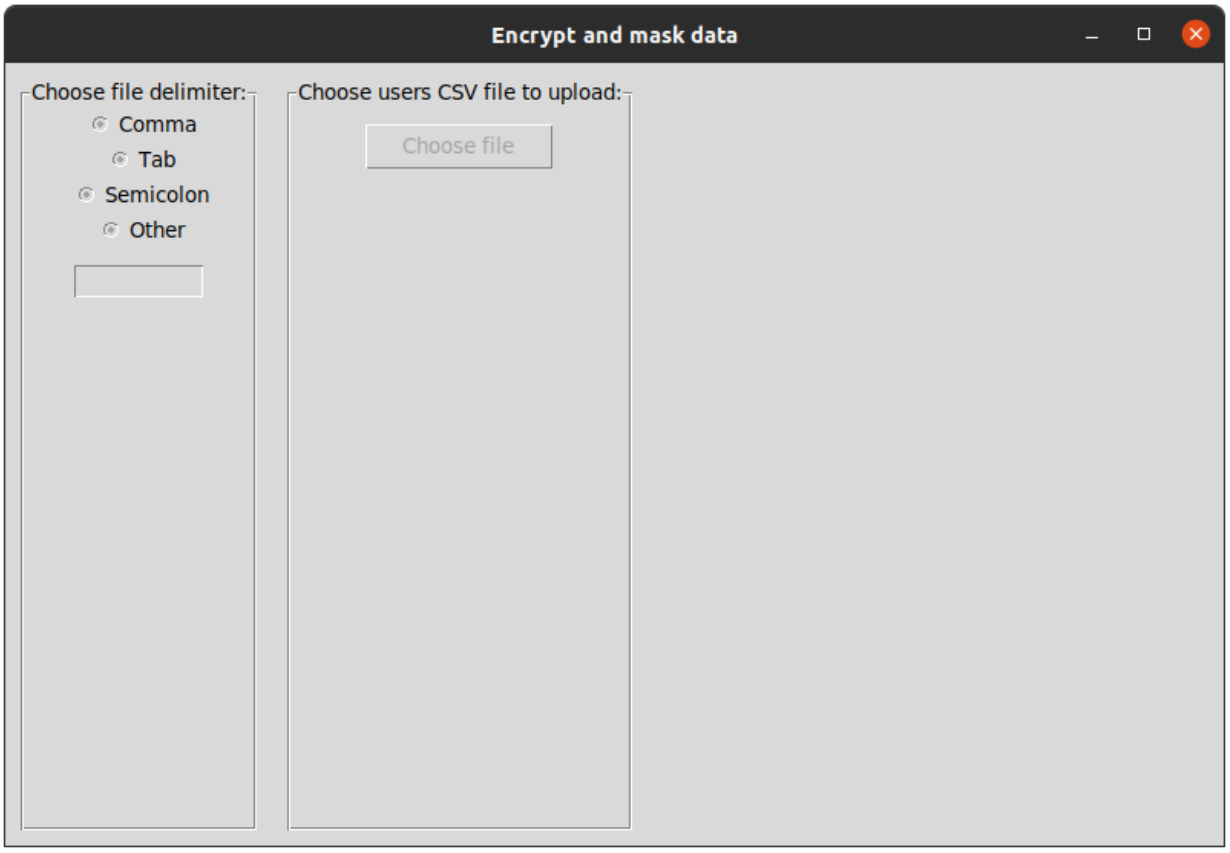


Рисунок 3.3 — Вікно програми з вибором деліметрів

Після вибору деліметра кнопка “Choose file” у labelframe-і “Choose users CSV file to upload” стане активною (рис. 3.4).

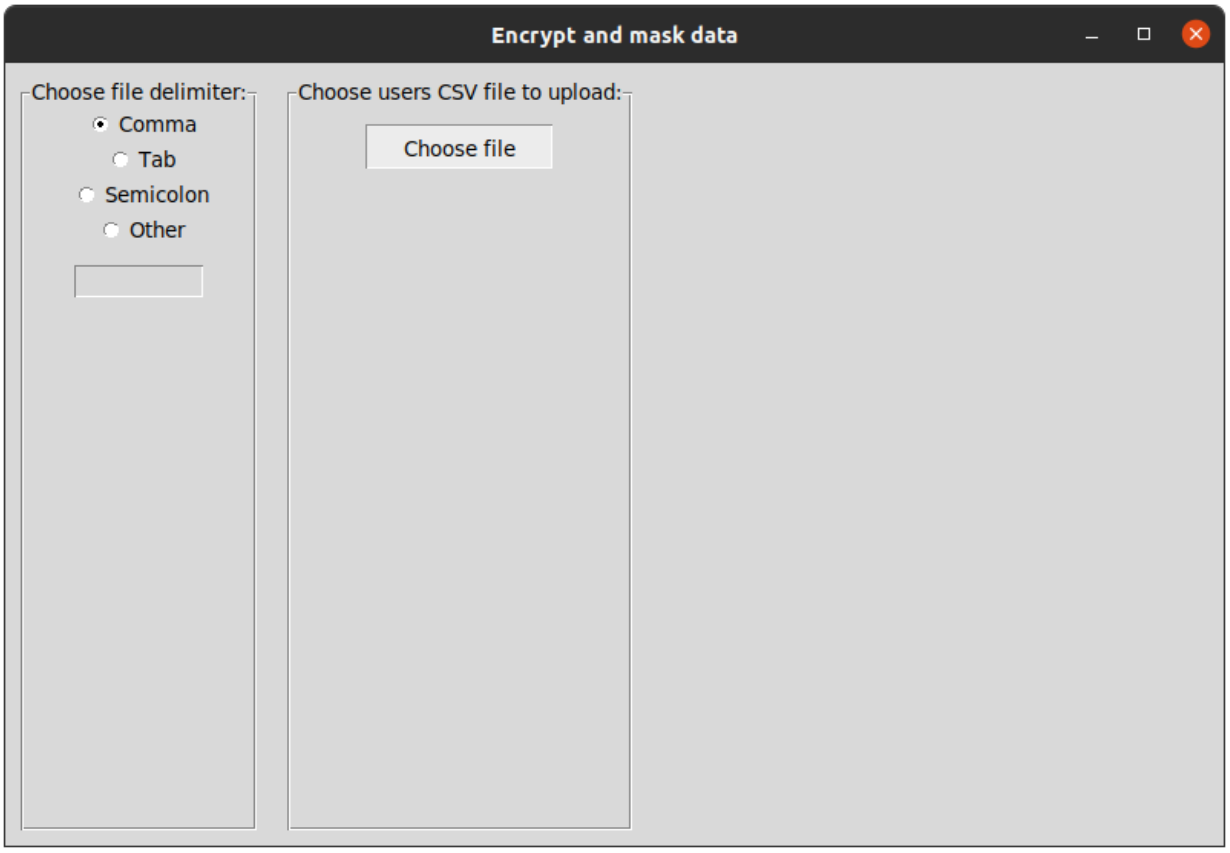


Рисунок 3.4 — Вікно програми з вибраним деліметром

Далі користувач вибирає потрібний .csv файл згідно попередньо обраного розділювача і завантажує його в програму (рис. 3.5).

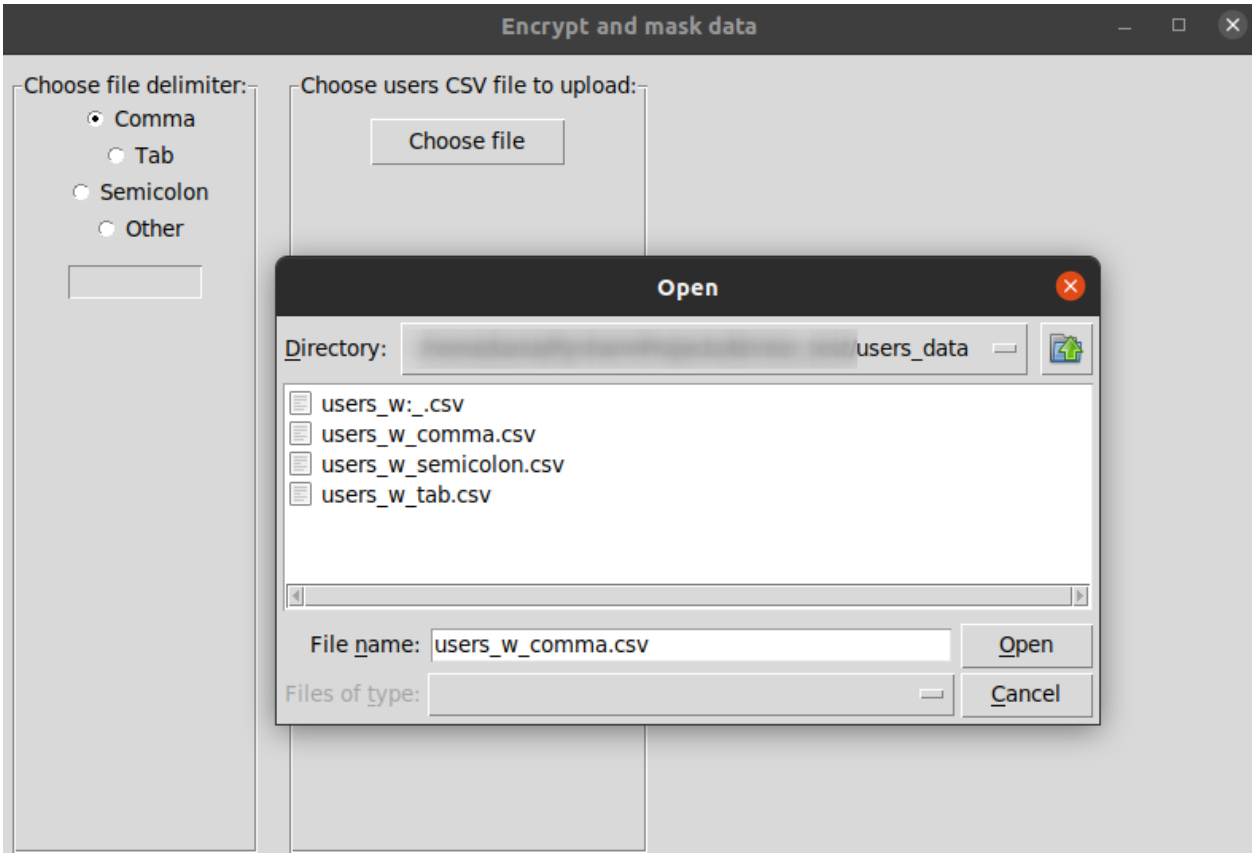


Рисунок 3.5 — Діалогове вікно вибору файла з юзерами

Після завантаження файлу у вікні з’являються додаткові віджети. Чотири labelframe-а містять такі віджети як treeview, labels і dropdown lists, та кнопки.

У верхньому labelframe-і буде показано treeview у якому буде прев’ю підгруженого юзером файлу. Усі колонки, котрі наявні у .csv у файлі будуть відображені у цьому віджеті (рис. 3.6).

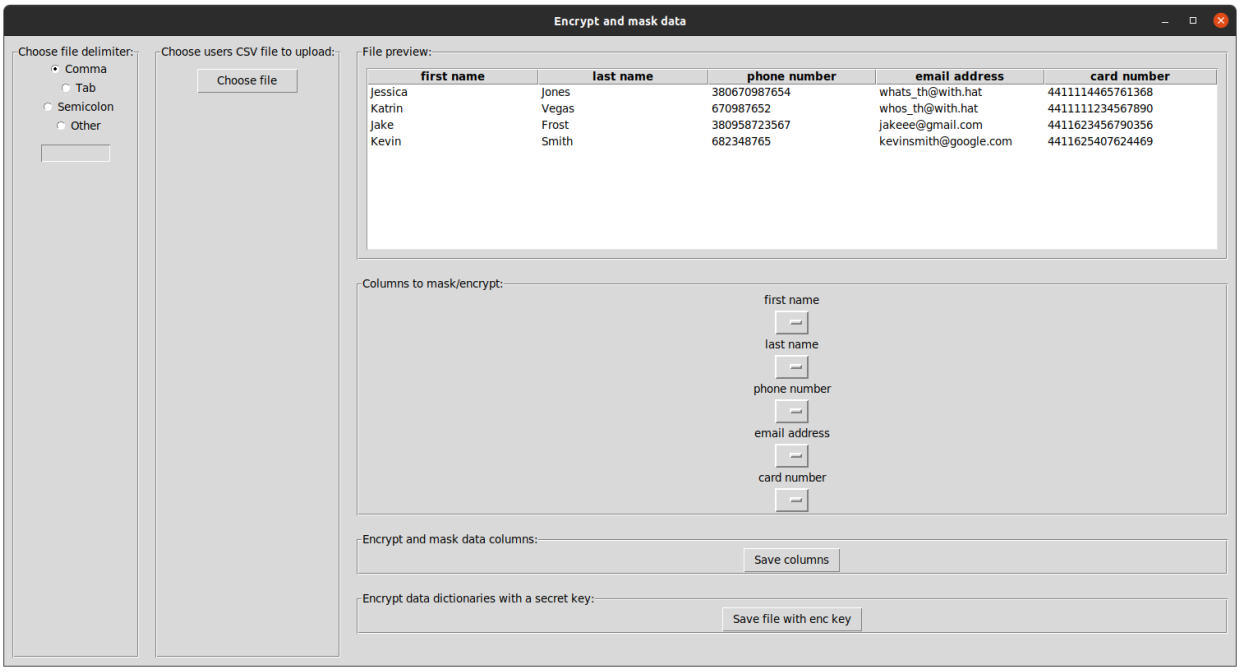


Рисунок 3.6 — Вікно програми з підвантаженим файлом

У наступному labelframe-і наявні динамічно згенеровані labels, та dropdown lists відповідно до кількості колонок у підвантаженому .csv файлі. Labels відповідають назвам колонок, а у опціях dropdown lists наявні назви методів якими користувач може зашифрувати або замаскувати відповідну до label-а колонку. Опціями dropdown lists є First name, Last name, Card number, Email, Phone (рис. 3.7).

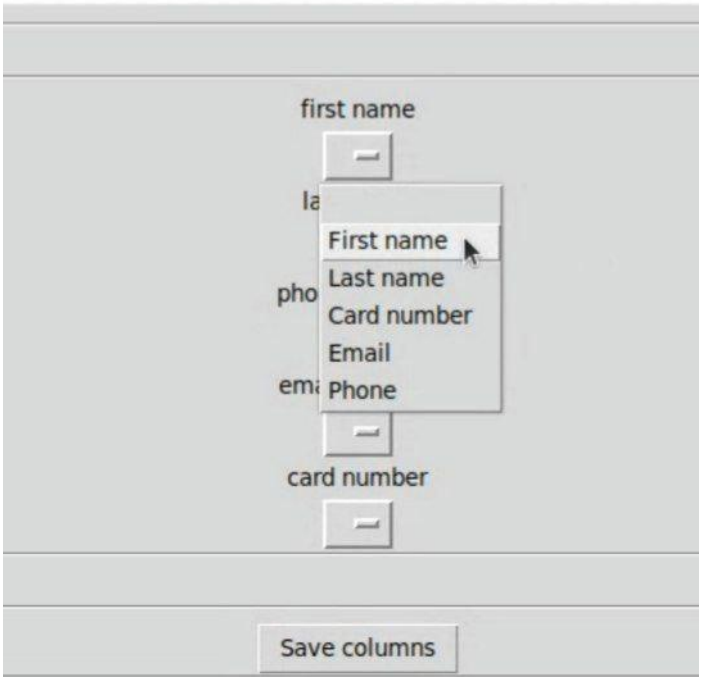


Рисунок 3.7 — Опції вибору у dropdown list-ах

Вибираємо підходящі опції dropdown lists до labels назв колонок відповідно(рис. 3.8):

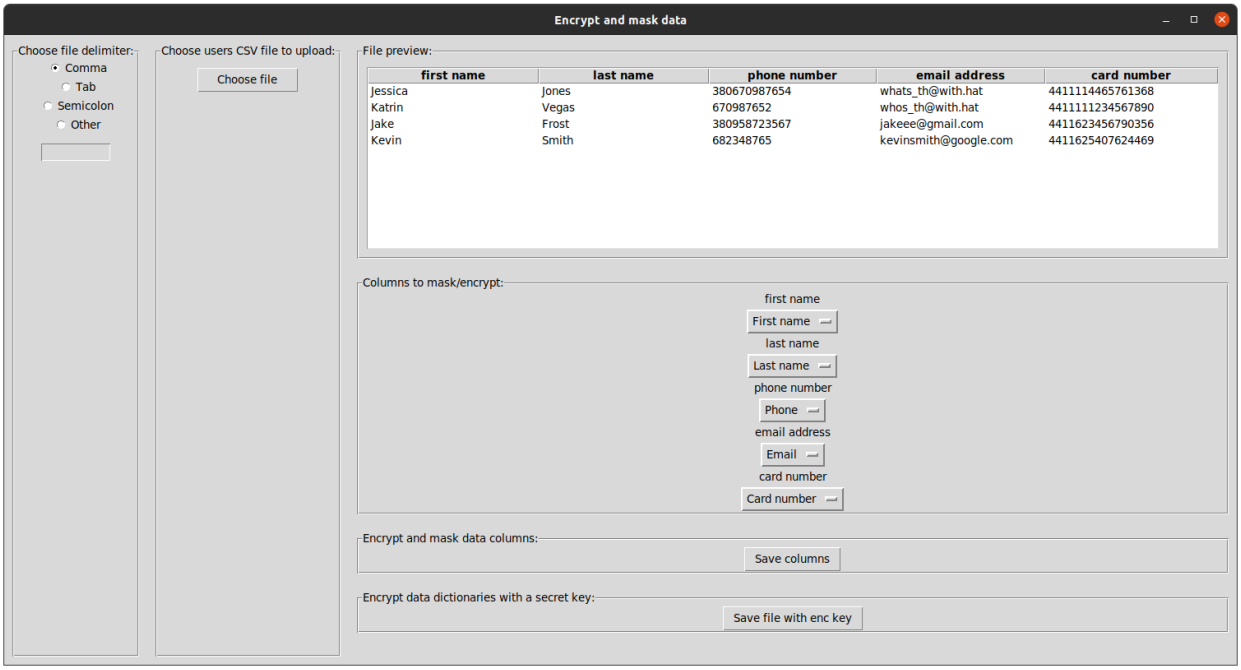


Рисунок 3.8 — Вікно з вибраними опціями у dropdown list-ах

Потім користувач натискає на кнопку “Save columns” та зберігає encrypted.csv файл разом з файлами словників з ключами відповідно до вибраних опцій в dropdown lists. Наступним кроком користувачу потрібно натиснути кнопку “Save file with enc key”, що зашифрує словники з ключами та збереже один файл ключа під назвою mykey.key для майбутнього розшифрування цих словників. Метод шифрування словників описаний у пункті 2.3.

Для порівняння вхідних і вихідних (замаскованих та зашифрованих даних наведені наступні скріншоти датасетів (рис 3.9) та (рис. 3.10) відповідно.

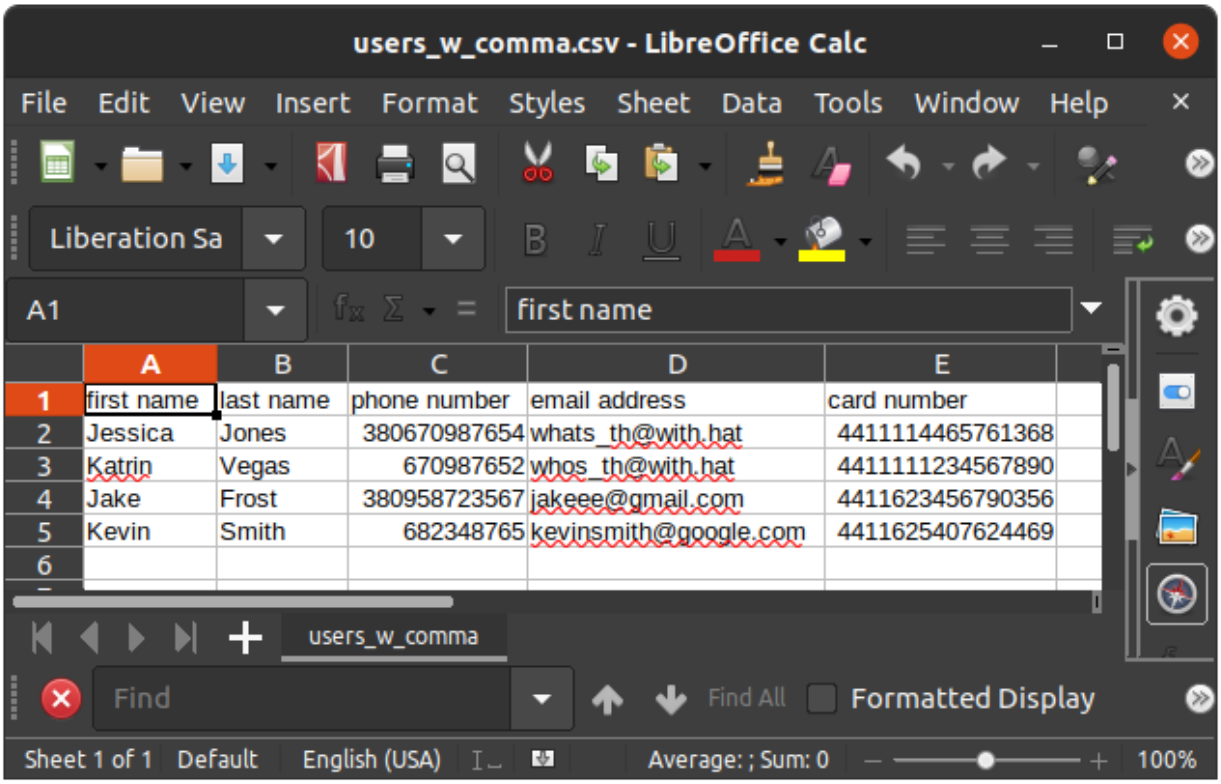


Рисунок 3.9 — Вхідний файл з реальними даними користувачів users_w_comma.csv

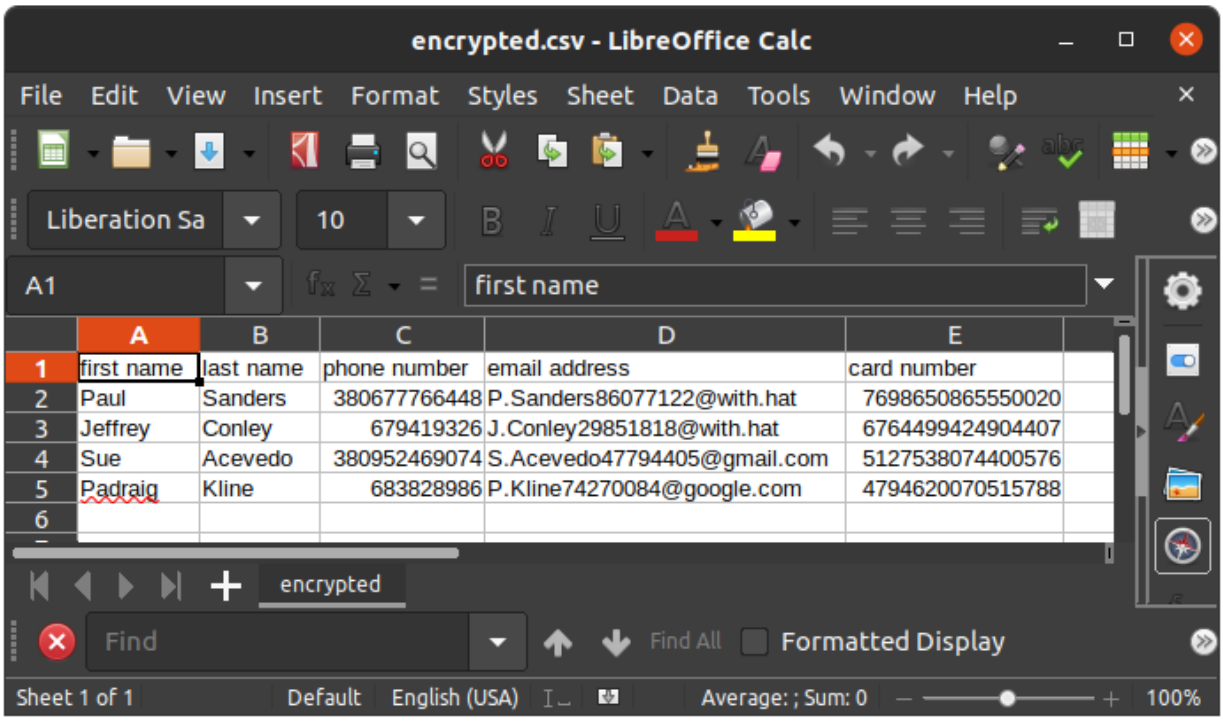


Рисунок 3.10 — Вихідний файл з замаскованими та зашифрованими даними користувачів encrypted.csv

Наступним пунктом є дешифрування та де-псевдонімізація даних. Для цього користувач при запуску програми вибирає опцію “Decrypt file” та клацає на активну кнопку “Next” (рис. 3.11).

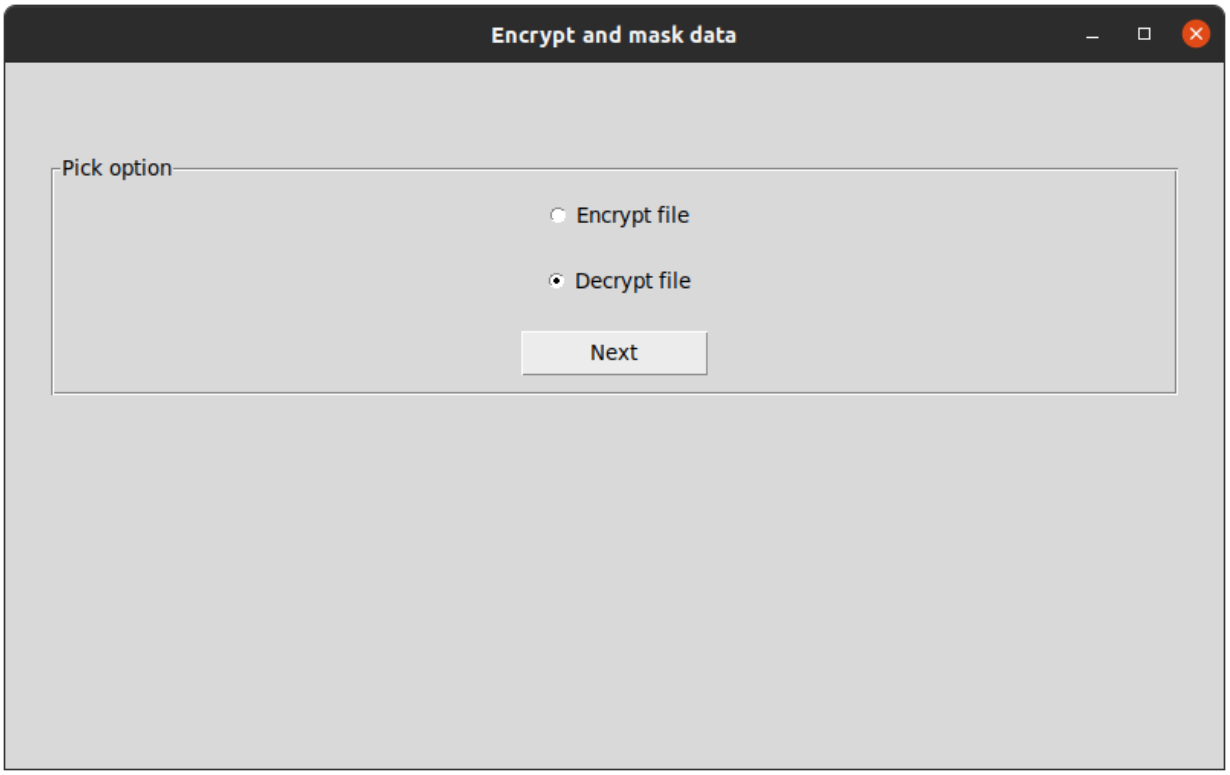


Рисунок 3.11 — Вікно програми при запуску з обраною опцією “Decrypt file”

Далі користувачу потрібно підвантажити файл encrypted.csv клікнувши на кнопку “Upload file” в labelframe-і “Upload file with masked and encrypted data:” (рис. 3.12)

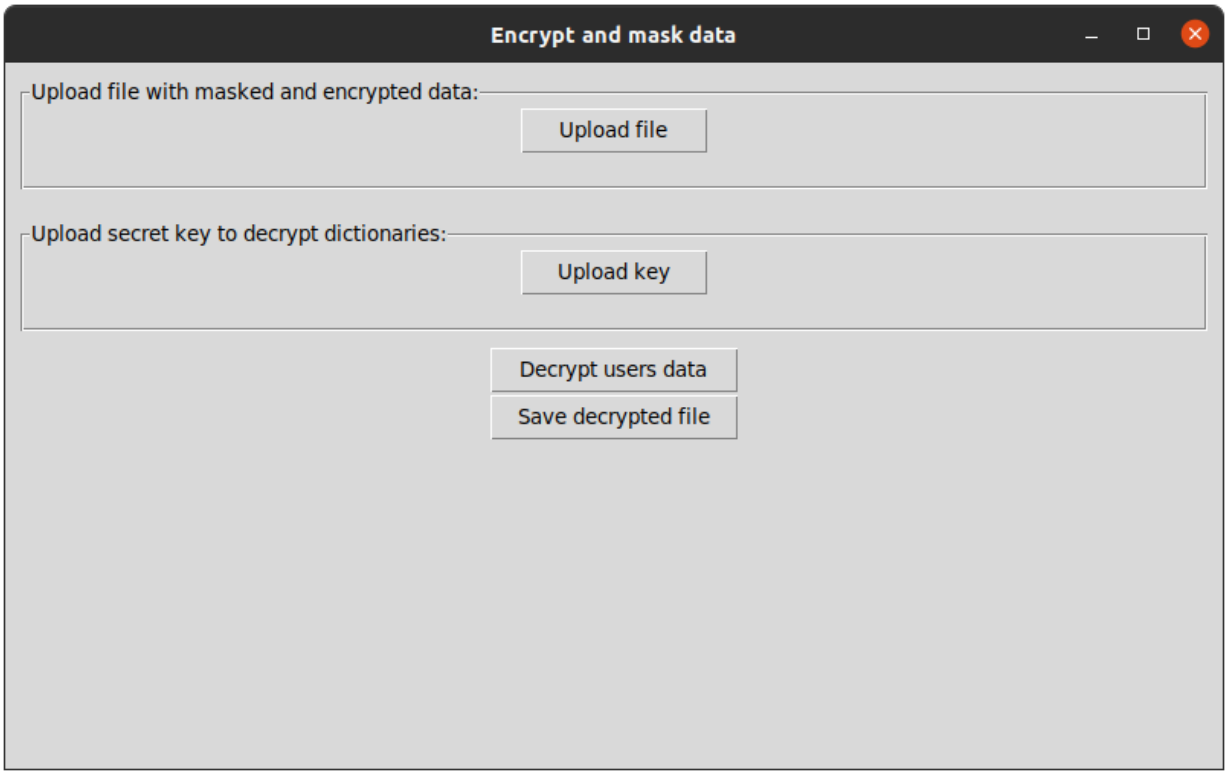


Рисунок 3.12 — Вікно програми при запуску в режимі дешифрування

Відкриється діалогове вікно, де юзер обере файл encrypted.csv (рис. 3.13).

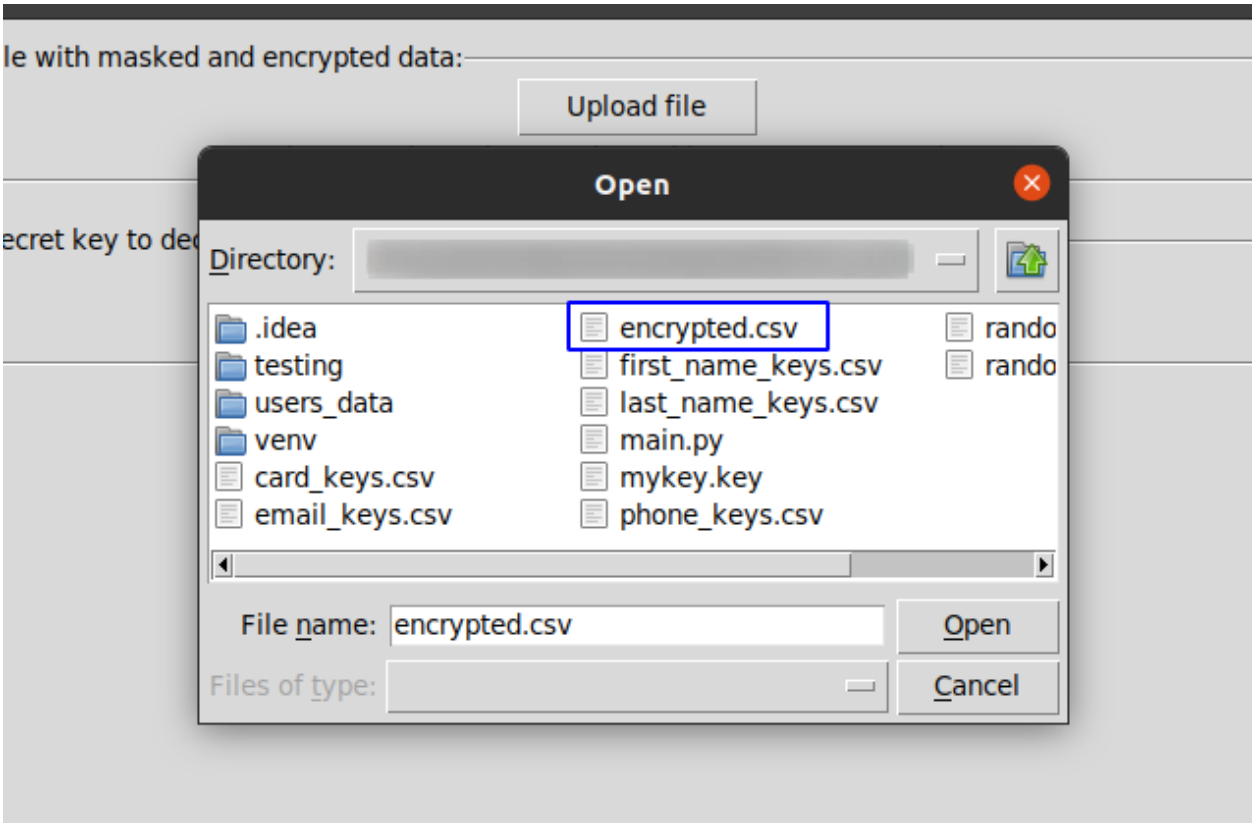


Рисунок 3.13 — Діалогове вікно вибору файлу з замаскованими даними

На label-і під кнопкою “Upload file” з’явиться шлях до підгруженого файлу, що буде сигналізувати, про успішно підгружений файл.

Наступним кроком буде натискання кнопки “Upload key” у labelframe-і “Upload secret key to decrypt dictionaries” (рис. 3.14).

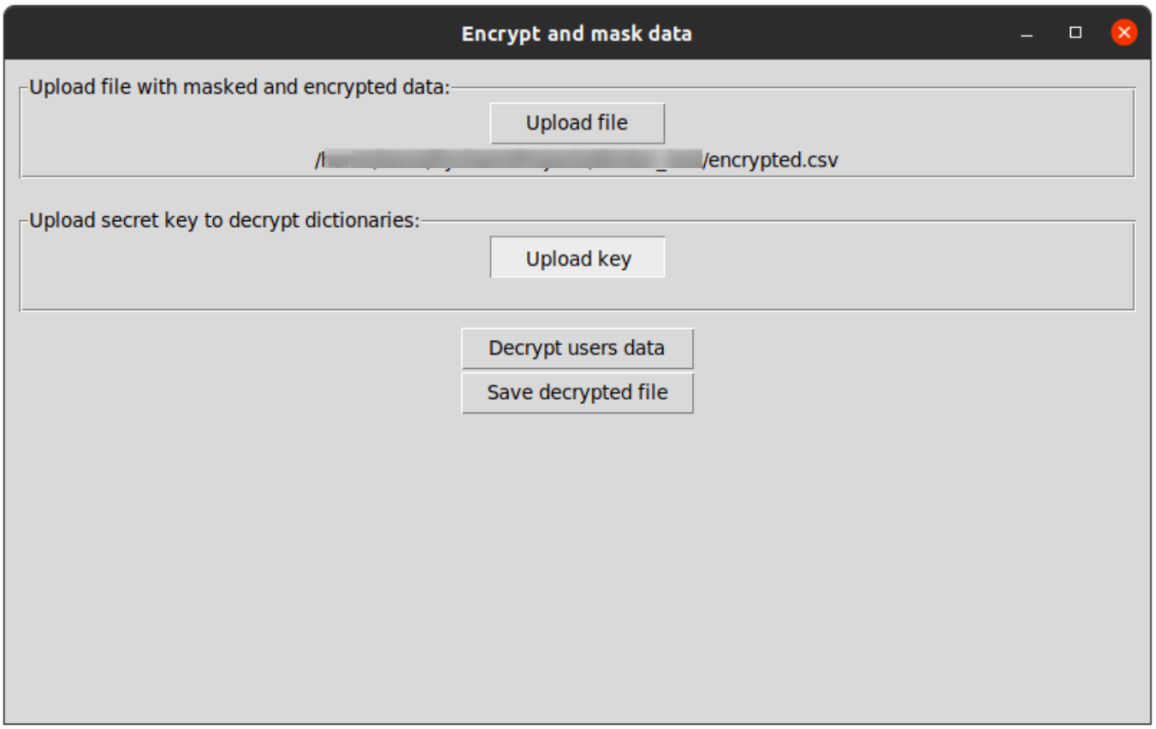


Рисунок. 3.14 — Вікно програми при завантаженому файлі ecrypted.csv

Відкриється діалогове вікно і користувачу потрібно вибрати файл “mykey.key” з дирукторії (рис. 3.15).

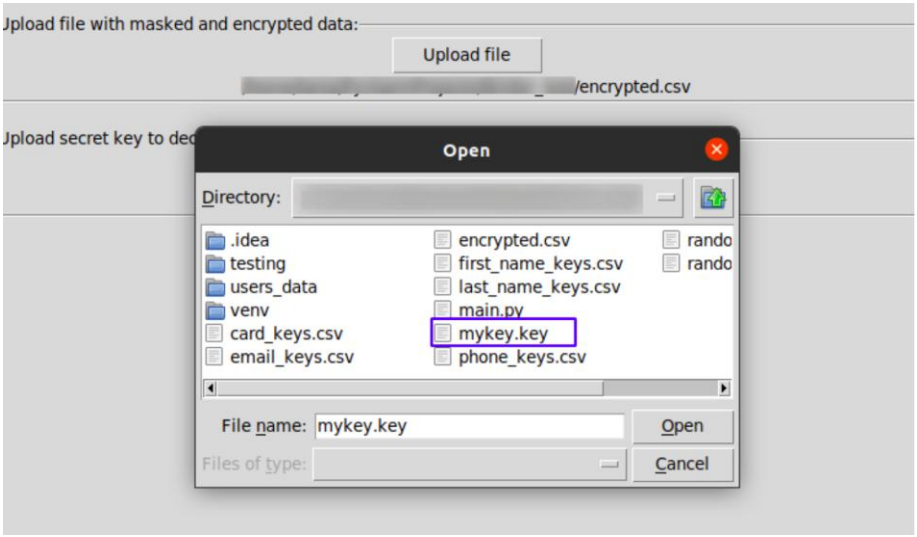


Рисунок 3.15 — Діалогове вікно вибору файлу ключа розшифрування

Після вибору файлу під кнопкою “Upload key” на label-і з’явиться шлях до підвінтаженого ключа. Після цього юзеру потрібно натиснути кнопку “Decrypt users data” (рис. 3.16).

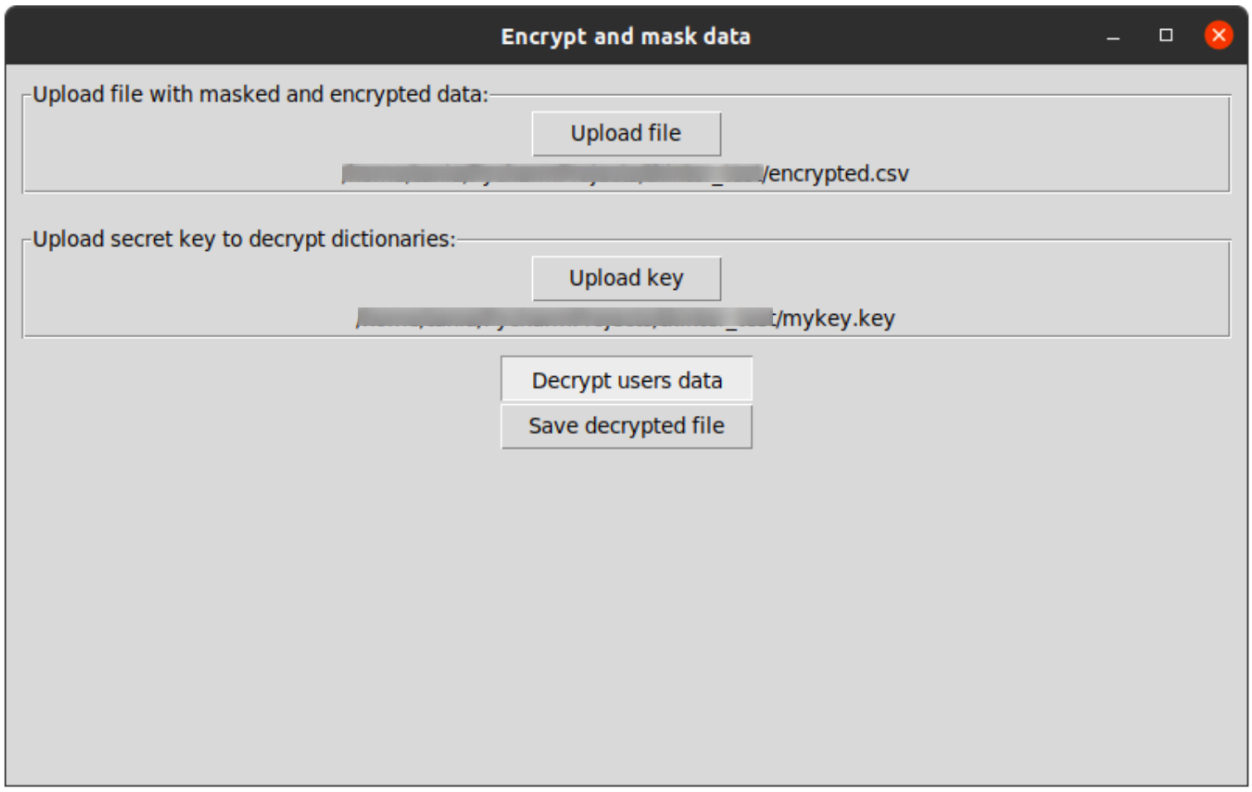


Рисунок 3.16 — Вікно програми при завантаженому ключу і натиснутій кнопці Decrypt users data

І зачекати на повідомлення від системи, що процес дешифрування і демаскування завершено (рис. 3.17).

Рисунок 3.17 — Вікно сповіщення про успішне розшифрування даних

Після цього юзер може зберегти розшифрований файл, натиснувши на “Save decrypted file” (рис. 3.18).

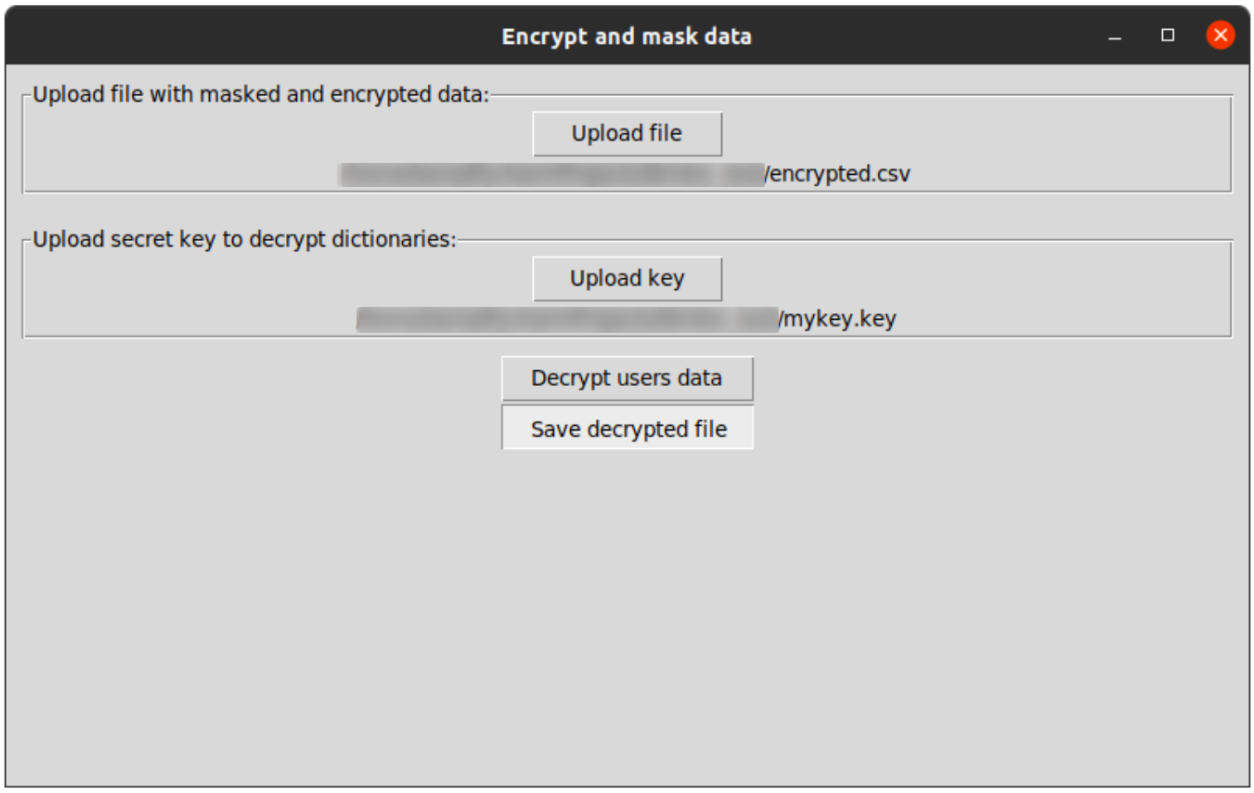


Рисунок 3.18 — Вікно програми при натиснутій кнопці Save decrypted file

Відкриється діалогове вікно, де юзеру буде запропоновано ввести ім’я розшифрованого файлу (рис. 3.19).

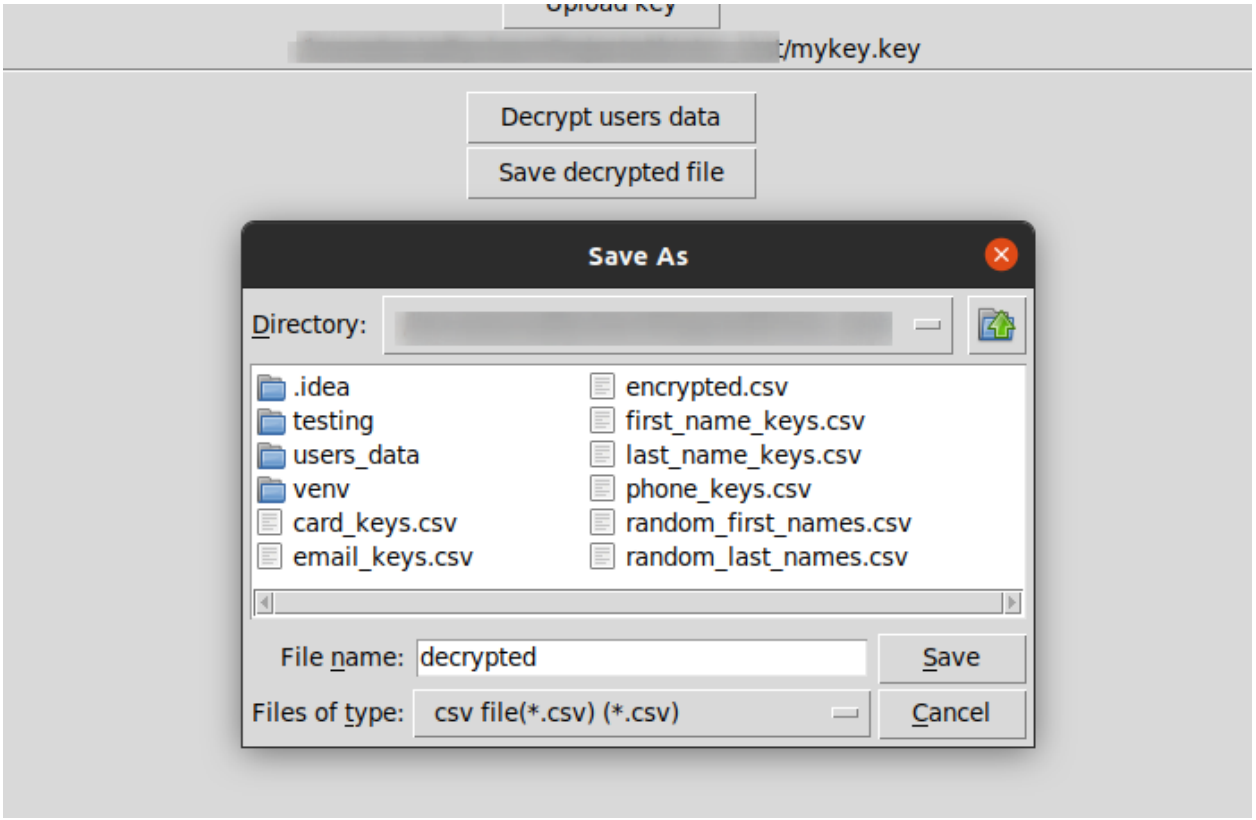


Рисунок 3.19 — Діалогове вікно для збереження розшифрованого файлу

Після цього файл буде збережено у вибраній директорії і таким чином процес дешифрування та де-псевдонімізації даних завершено.

3.3 Реалізація методу шифрування зі збереженням формату та псевдонімізації даних

Метод шифрування зі збереженням формату та псевдонімізації даних у розробленому додатку було реалізовано за допомогою використання наступних бібліотек Python:

- codecs - цей модуль визначає базові класи для стандартних кодеків Python (кодери та декодери) і надає доступ до внутрішнього реєстру кодеків Python, який керує процесом пошуку кодека та обробки помилок [44];

– string - цей модуль допомагає швидко отримати доступ до деяких рядкових констант [45];

– cryptography.fernet >>Fernet - цей модуль криптографічного пакета cryptography має вбудовані функції для генерації ключа, шифрування відкритого тексту в зашифрований текст і розшифрування зашифрованого тексту в відкритий текст за допомогою методів шифрування та розшифрування відповідно. Модуль fernet гарантує, що даними, зашифрованими з його допомогою, неможливо буде надалі маніпулювати або зчитувати без ключа[46];

– pandas - цей модуль надає готові до використання високопродуктивні структури даних і інструменти аналізу даних. Він працює поверх NumPy і широко використовується для аналізу даних[47];

– csv - цей модуль реалізує класи для читання та запису табличних даних у форматі CS[48];

– random - цей модуль реалізує генератори псевдовипадкових чисел для різних розподілів. Для цілих чисел існує рівномірний вибір із діапазону [49];

– os - цей модуль надає функції для створення та видалення каталогу (папки), отримання його вмісту, зміни та визначення поточного каталогу тощо [50];

– binascii - цей модуль містить низку методів для перетворення двійкових представлень у кодуванні ASCII [51];

– fff3 >> FF3Cipher - цей модуль необхідний для шифрування із збереженням формату в Python, містить в собі реалізацію затверджених NIST алгоритмів FF3 і FF3-1 Format Preserving Encryption (FPE). Цей пакет реалізує алгоритм FF3 для шифрування зі збереженням формату, як описано в публікації NIST 800-38G за березень 2016 року та переглянутиф 28 лютого 2019 року з чернеткою оновлення для FF3-1 [52];

– Crypto.Cipher >> AES - цей модуль реалізує симетричний шифр AES[53];

– numpy - це модуль, що використовується для роботи з масивами. Він також має функції для роботи в області лінійної алгебри, перетворення Фур'є та матриць [54];

– pandas >> read_csv - цей модуль дозволяє прочитати файл зі значеннями, розділеними комами (csv) у DataFrame. Також підтримує додаткову ітерацію або розбиття файлу на частини [55].

ВИСНОВКИ

Під час написання дипломної роботи були проаналізовані ключові поняття в рамках аналізу предметної області, а саме: персональна інформація, шифрування зі збереженням формату та псевдонімізація даних. При огляді існуючих режимів шифрування зі збереженням формату, був виділений режим FF3-1, який був реалізований у програмному додатку.

Також був проведений аналіз продуктів з імплементованим методами псевдонімізації та шифруванням зі збереженням формату та виділені їх сильні та слабкі сторони.

Після аналізу сильних та слабких сторін продуктів на ринку, що реалізували шифрування зі збереженням формату та псевдонімізацію даних, було покращено методи псевдонімізації та маскування полів даних: типу ім’я, прізвище та email адреса та розумне застосування методу FPE в режимі FF3-1 для полів типу номер телефону та кредитної карти.

Було також розроблене та імплементоване рішення для захисту словників даних за допомогою AES (режим CBC) з ключем шифрування в 128-біт.

В заключення, був розроблений додаток для шифрування даних зі збереженням формату та удосконаленими методами псевдонімізації на високорівневій мові програмування Python з використанням допоміжних бібліотек.

ПЕРЕЛІК ПОСИЛАНЬ

1. PII Protect Cybersecurity | How to Secure your Data URL: https://thecyphere.com/blog/pii-protect/
2. What is format-preserving encryption (FPE) and its benefits? URL: https://www.ubiqsecurity.com/what-is-format-preserving-encryption-fpe-and-its-benefits%EF%BF%BC/
3. What is Personally Identifiable Information? Definition + Examples URL: https://www.upguard.com/blog/personally-identifiable-information-pii
4. КОНФІДЕНЦІЙНА ІНФОРМАЦІЯ, ІНФОРМАЦІЯ ПРО ОСОБУ ТА ПЕРСОНАЛЬНІ ДАНІ: СПІВВІДНОШЕННЯ І РЕГУЛЮВАННЯ URL: https://cedem.org.ua/analytics/konfidentsijna-informatsiya-informatsiya-pro-osobu-ta-personalni-dani-spivvidnoshennya-i-regulyuvannya/
5. PII Security & Internet Privacy Monitoring URL: https://www.idshield.com/pii-protection-online-data-privacy-monitoring-idshield/
6. Format-preserving encryption use cases, benefits, alternative URL: https://www.techtarget.com/searchsecurity/tip/Format-preserving-encryption-use-cases-benefits-alternative
7. PSEUDONYMIZATION AND ANONYMIZATION OF PERSONAL DATA URL: https://complior.se/pseudonymization-and-anonymization-of-personal-data-what-is-the-difference/
8. Data masking: Anonymisation or pseudonymisation? URL: https://www.grcworldforums.com/data-management/data-masking-anonymisation-or-pseudonymisation/12.article
9. Pseudonymization according to the GDPR [definitions and examples] URL: https://dataprivacymanager.net/pseudonymization-according-to-the-gdpr/

10. Advanced Encryption Standard (AES) URL: https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard
11. Breaking the FF3 Format-Preserving Encryption Standard over Small Domains URL: https://link.springer.com/chapter/10.1007/978-3-319-63715-0_23
12. Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption URL: https://csrc.nist.gov/publications/detail/sp/800-38g/rev-1/draft
13. What is Format Preserving Encryption (FPE)? Is Format Preserving Encryption secure? URL: https://www.encryptionconsulting.com/education-center/what-is-fpe/
14. Cryptanalysis of Feistel-Based Format-Preserving Encryption URL: https://eprint.iacr.org/2020/1311.pdf
15. FF3-1 Tweak Usage Documentation URL: https://developer.hashicorp.com/vault/docs/secrets/transform/ff3-tweak-details
16. Recent Cryptanalysis of FF3 URL: https://csrc.nist.gov/news/2017/recent-cryptanalysis-of-ff3
17. Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption URL: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-38g.pdf
18. Format Preserving Encryption (FPE) and Pseudonymization Demonstration URL: https://www.youtube.com/watch?v=szeKl6PZ1wE&ab_channel=IRITheCoSortCo
19. IRI FieldShield - Consistent RDB Masking URL: https://www.youtube.com/watch?v=7nxp3TjsCqU&ab_channel=IRITheCoSortCo
20. IRI CoSort IRI, The CoSort Company Pricing URL: https://sourceforge.net/software/product/IRI-CoSort/
21. Transform Secrets Engine URL: https://developer.hashicorp.com/vault/tutorials/adp/transform
22. HCP Vault Pricing URL: https://cloud.hashicorp.com/products/vault/pricing

23. COMFORTE DATA PROTECTION URL: https://www.comforte.com/fileadmin/Collateral/SB_comforte_Data_Protection.pdf
24. Application Level Encryption Overview URL: https://www.futurex.com/application-level-encryption/
25. Voltage SecureData URL: https://www.xmartsolutions.com.br/site/datasheets/voltageSecureData.pdf
26. Encrypting Private Data In Hacking the Code, 2004 Employing Symmetric Cryptography URL: https://www.sciencedirect.com/topics/computer-science/symmetric-crypt-?graphy#:~:text=Symmetric%20cryptography%2C%20known%20also%20as,and%20to%20decrypt%20the%20data.
27. Roeder, Tom. "Symmetric-Key Cryptography" URL: http://www.cs.cornell.edu/courses/cs5430/2010sp/TL03.symmetric.html
28. Symmetric vs. Asymmetric Encryption: What's the Difference? URL: https://www.trentonsystems.com/blog/symmetric-vs-asymmetric-encryption
29. Efficient Format Preserving Encrypted Databases URL: https://www.researchgate.net/publication/285711616_Efficient_Format_Preserving_Encrypted_Databases
30. Format-preserving encryption: Overview and NIST specification URL: https://www.tandfonline.com/doi/full/10.1080/01611194.2016.1169457
31. Data Masking URL: https://www.imperva.com/learn/data-security/data-masking/
32. Credit card numbers explained URL: https://www.lloydsbank.com/credit-cards/help-and-guidance/credit-card-numbers-explained.html#:~:text=1.,and%20is%20unique%20to%20you
33. Credit Card Numbers: What Do They Mean? URL: https://www.forbes.com/advisor/credit-cards/what-does-your-credit-card-number-mean/

34. Advanced Encryption Standard (AES) URL: https://www.geeksforgeeks.org/advanced-encryption-standard-aes/
35. The Rijndael Block Cipher URL: https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf
36. What Is AES Encryption? URL: https://www.trentonsystems.com/blog/aes-encryption-your-faqs-answered
37. The Clock Is Ticking for Encryption URL: https://www.computerworld.com/article/2550008/the-clock-is-ticking-for-encryption.html
38. Controlled Unclassified Information (CUI) URL: https://www.archives.gov/cui/about
39. Federal Information Processing Standards Publication 197 November 26, 2001 Announcing the ADVANCED ENCRYPTION STANDARD (AES) URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=901427
40. What is Python? URL: https://www.teradata.com/Glossary/What-is-Python
41. What is Python? Executive Summary URL: https://www.python.org/doc/essays/blurb/
42. What is PyCharm? URL: https://intellipaat.com/blog/what-is-pycharm/
43. Python GUI Programming With Tkinter URL: https://realpython.com/python-gui-tkinter/
44. codecs — Codec registry and base classes URL: https://www.cmi.ac.in/~madhavan/courses/prog2-2015/docs/python-3.4.2-docs-html/library/codecs.html
45. string — Common string operations URL: https://docs.python.org/3/library/string.html

46. Fernet (symmetric encryption) using Cryptography module in Python URL: https://www.geeksforgeeks.org/fernet-symmetric-encryption-using-cryptography-module-in-python/
47. Python Pandas Module Tutorial URL: https://www.digitalocean.com/community/tutorials/python-pandas-module-tutorial
48. csv — CSV File Reading and Writing URL: https://docs.python.org/3/library/csv.html
49. random — Generate pseudo-random numbers URL: https://docs.python.org/3/library/random.html
50. Python - OS Module URL: https://www.tutorialsteacher.com/python/os-module
51. binascii — Convert between binary and ASCII URL: https://docs.python.org/3/library/binascii.html
52. FF3 - Format Preserving Encryption in Python ff3 1.0.1 URL: https://pypi.org/project/ff3/
53. Module AES URL: https://www.dlitz.net/software/pycrypto/api/2.6/Crypto.Cipher.AES-module.html
54. NumPy Introduction URL: https://www.w3schools.com/python/numpy/numpy_intro.asp
55. Pandas.read_csv URL: https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html