Ім'я користувача:
приховано налаштуваннями конфіденційності

Дата перевірки:
02.06.2023 10:49:46 EEST

Дата звіту:
02.06.2023 10:53:17 EEST

Клим_Христина_КН_320
ID перевірки: 1015386557

Тип перевірки: Doc vs Library

ID користувача: 100011372

Назва документа:

Кількість сторінок: 29 Кількість слів: 3770 Кількість символів: 30017 Розмір файлу: 2.26 MB ID файлу: 1015051587

1.83%
Схожість

Найбільша схожість: 0.74% з джерелом з Бібліотеки (ID файлу: 1014439181)

Пошук збігів з Інтернетом не проводився

Вилучення цитат вимкнене

.....

...С..т..о..р..і.н..к.а...3..1.....

0% Цитат

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Назва документа: Клим_Христина_КН_320 ID файлу: 1015051587

1 ПРОБЛЕМА ЗАХИСТУ ІНФОРМАЦІЇ

1.1 Актуальність проблеми захисту інформації

Проблеми захисту інформації включають широкий спектр викликів та загроз, з якими стикаються організації та користувачі інформаційних систем. Нижче перераховано деякі з найважливіших проблем захисту інформації.

Несанкціонований доступ. Це одна з основних загроз для безпеки інформації. Несанкціоновані особи можуть намагатися отримати доступ до конфіденційних даних або систем, щоб скрасти, модифікувати або пошкодити їх. Це може бути здійснено шляхом хакерських атак, фішингу, використання слабких паролів або недостатньої фізичної безпеки.

Втрата даних. Втрата даних може бути наслідком несправностей обладнання, природних катастроф, вторгнень або неправильних дій користувачів. Втрачена або пошкоджена інформація може призвести до серйозних фінансових втрат, втрати довіри клієнтів або порушення вимог законодавства щодо конфіденційності даних.

Вразливості програмного забезпечення. Багато комп'ютерних програм мають вразливості, які можуть бути використані для несанкціонованого доступу або атак на системи. Це можуть бути програмні помилки, слабкі паролі, недостатня перевірка введених даних або недостатня захист мережевого з'єднання.

Соціальний інжиніринг. Соціальний інжиніринг означає використання маніпуляцій та обману людей для отримання доступу до конфіденційної інформації. Це може включати фішингові атаки, вимагання паролів або імітацію співробітників організації.

Шифрування і дешифрування. Правильне використання шифрування та дешифрування є важливим аспектом захисту інформації. Проблеми можуть виникати внаслідок використання слабих алгоритмів шифрування, недостатньої довжини ключів, неправильної реалізації або недостатнього керування ключами.

Загрози мережевої безпеки. Зі зростанням використання мереж та Інтернету

з'явилися нові загрози мережевої безпеки. Це можуть бути мережеві атаки,

деніал- сервіс атаки, шпигунство, віруси, черв'яки тощо.

Софтверні вразливості. Вразливості у програмному забезпеченні можуть бути використані зловмисниками для отримання несанкціонованого доступу до системи. Помилки програмування, недостатня перевірка вхідних даних, небезпечні функції або недостатні заходи безпеки можуть створювати потенційні вразливості. Патчі і оновлення програмного забезпечення використовуються для виправлення виявлених вразливостей.

Внутрішні загрози. Загрози з боку внутрішніх джерел, таких як співробітники або партнери, також є значним ризиком. Несанкціонований доступ або недбале ставлення до конфіденційної інформації з боку осіб, які вже мають доступ до системи, можуть призвести до серйозних проблем безпеки.

Фізична безпека. Захист фізичного обладнання та інфраструктури, таких як серверні кімнати, дата-центри та комунікаційні канали, є необхідним для забезпечення безпеки інформації. Це може включати контроль доступу до приміщень, використання систем відеоспостереження, захист від пожежі та інші фізичні заходи безпеки.

Захист мобільних пристроїв. Зі зростанням використання смартфонів, планшетів та інших мобільних пристроїв, захист даних на цих пристроях стає важливим завданням. Втрату або крадіжку мобільного пристрою може супроводжувати небезпека доступу до конфіденційних даних. Крім того, мобільні пристрої можуть бути вразливі до шкідливих програм або атак з мережі.

Суспільно-правовий аспект. Захист інформації також пов'язаний з суспільно- правовими аспектами. Регулювання щодо захисту персональних даних, конфіденційності інформації та кібербезпеки може варіюватись в різних країнах. Організації повинні дотримуватись відповідних нормативних вимог та стандартів безпеки.

Важливо враховувати ці проблеми та вживати відповідних заходів для захисту інформації. А саме: використання сильних паролів, регулярне оновлення програмного забезпечення, використання надійних криптографічних алгоритмів, навчання користувачів безпеки та реалізацію ефективних політик безпеки.

1.2 Класифікація засобів захисту інформації

Засоби захисту інформації можна класифікувати залежно від способу їх застосування та характеру захищеної інформації. Зазвичай використовують такі класифікації:

За видами інформації:

- Фізичний захист: засоби і методи, спрямовані на фізичну безпеку інформації, такі як контроль доступу, охоронні системи, біометричні системи.
- Захист даних: засоби, спрямовані на захист конфіденційності, цілісності та доступності даних, такі як шифрування, резервне копіювання, антивіруси тощо.
- Захист мережі: засоби, спрямовані на захист мережевих ресурсів, такі як мережеві брандмауери, віртуальні приватні мережі (VPN), системи виявлення вторгнень (IDS/IPS) тощо.

За рівнем захисту:

- Базовий рівень: основні заходи безпеки, такі як паролі, фізичний доступ, файрволи тощо.
- Розширений рівень: використання більш складних методів індентифікації, шифрування даних, застосування політик безпеки тощо.
- Високий рівень: використання передових технологій, багатофакторна аутентифікація, захист від складних загроз, стійкі криптографічні алгоритми. За методами захисту:
- Криптографічний захист: використання шифрування, хешування, цифрових підписів та інших криптографічних методів для захисту інформації.
- Біометричний захист: використання фізичних характеристик, таких як відбитки пальців, розпізнавання обличчя, розпізнавання голосу тощо, для ідентифікації та автентифікації користувачів.
- Соціальний захист: освіта та навчання користувачів щодо загроз і методів захисту, управління правами доступу, політики безпеки тощо.

На рис. 1.1 представлено класифікацію засобів захисту інформації.

Рисунок 1.1 – Класифікація засобів захисту інформації

1.3 Забезпечення безпеки інформації

Забезпечення безпеки інформації є важливим аспектом для захисту конфіденційності, цілісності та доступності даних. Існує кілька принципів та методів, які використовуються для забезпечення безпеки інформації. Основна мета полягає у запобіганні несанкціонованому доступу, зміні чи втраті даних.

Принципи та методи, які використовуються для забезпечення безпеки інформації:

1. Конфіденційність. Принцип конфіденційності вимагає, щоб лише авторизовані особи мали доступ до конфіденційної інформації. Це може бути досягнуто за допомогою шифрування даних та використанням механізмів контролю доступу. Наприклад, криптографічні алгоритми, такі як AES (Advanced Encryption Standard), RSA (Rivest-Shamir-Adleman) та інші, використовуються для захисту конфіденційної інформації.

2. Цілісність. Принцип цілісності ставить за мету забезпечити, щоб дані залишалися незмінними та несуттєвими протягом всього процесу їх збереження та передачі. Для досягнення цілісності даних можуть використовуватися контрольні суми (наприклад, MD5, SHA-1) або цифрові підписи, які дозволяють перевіряти, чи були дані змінені під час передачі або збереження.

3. Доступність. Принцип доступності вимагає, щоб інформація була доступною для авторизованих користувачів, коли вони потребують до неї доступу. Це може включати захист від вторгнень, забезпечення високої доступності систем та резервне копіювання даних. Фізична безпека приміщень та обладнання також відіграє важливу роль у забезпеченні доступності даних.

4. Аутентифікація. Перевірка ідентичності користувача, системи або процесу. Це може включати використання паролів, біометричних методів, криптографічних ключів або інших методів ідентифікації.

5. Аудит і моніторинг. Ведення журналів подій, аналіз активності і виявлення вразливостей або несанкціонованих дій. Це допомагає виявляти і

реагувати на потенційні загрози безпеці, виявляти аномальну активність та встановлювати політики безпеки.

6. Фізична безпека. Захист фізичного обладнання, приміщень та ресурсів, що зберігають інформацію, від несанкціонованого доступу, втрати або пошкодження.

7. Соціальна інженерія. Запобігання маніпуляції або обману людей з метою отримання несанкціонованого доступу до інформації або систем.

Для забезпечення безпеки даних важливо використовувати комплексний підхід, комбінуючи технічні та організаційні заходи, а також постійно вдосконалювати безпеку, враховуючи зростаючі загрози та нові технології.

1.4 Методи шифрування і дешифрування даних при їх передачі

Методи шифрування і дешифрування даних при їх передачі використовуються для забезпечення конфіденційності та цілісності даних під час їх передачі через незахищені канали зв'язку. Нижче наведено опис декількох основних методів шифрування та дешифрування даних:

Симетричне шифрування. У симетричному шифруванні використовується один і той же ключ як для шифрування, так і для дешифрування даних. Популярні алгоритми симетричного шифрування включають AES (Advanced Encryption Standard), DES (Data Encryption Standard) та 3DES (Triple Data Encryption Standard). У цих алгоритмів дані шифруються за допомогою ключа, і той же ключ використовується для розшифрування.

Асиметричне шифрування. Асиметричне шифрування використовує пару ключів: публічний ключ і приватний ключ. Публічний ключ використовується для шифрування даних, а приватний ключ - для їх дешифрування. За допомогою асиметричного шифрування можна створювати цифрові підписи, що дозволяють перевірити автентичність даних. Популярними алгоритмами асиметричного шифрування є RSA, ECC (Elliptic Curve Cryptography) та DSA (Digital Signature Algorithm).

Гібридне шифрування. Гібридне шифрування комбінує симетричне та асиметричне шифрування для отримання оптимального балансу швидкості та безпеки. У цьому підході симетричне шифрування використовується для шифрування самої інформації, а асиметричне шифрування - для захисту симетричного ключа. Таким чином, передача даних відбувається шифрованим симетричним ключем, який обмінюється за допомогою асиметричного шифрування. Прикладом гібридного шифрування є протокол SSL/TLS, який використовується для захищеної передачі даних по Інтернету. На рис 1.2 схематично наведено основні методи шифрування та дешифрування даних.

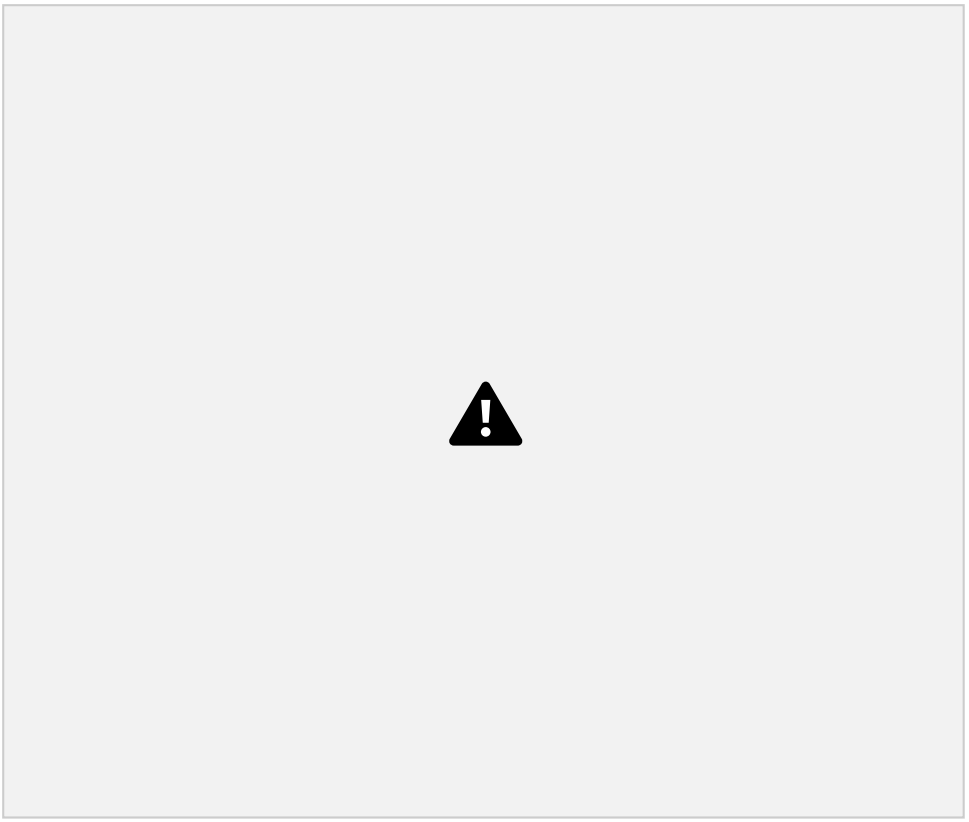


Рисунок 1.2 – Методи шифрування/дешифрування даних

2 ПОБУДОВА КРИПТОГРАФІЧНОГО АЛГОРИТМУ З ВИКОРИСТАННЯМ ШИФРУ ГАММУВАННЯ

2.1 Основні властивості шифру гаммування

Шифр гаммування (stream cipher) є одним з основних методів шифрування, який використовується для забезпечення конфіденційності даних. Він відрізняється від блочних шифрів, оскільки оперує не блоками даних, а потоком символів. Основними властивостями шифру гаммування є:

Послідовність гами. Для шифру гаммування необхідно мати генератор гами, який створює випадкову або псевдовипадкову послідовність символів, яка називається гамою. Гама повинна мати довільну довжину і бути стійкою до криптоаналізу.

Використання XOR. Шифрування здійснюється шляхом використання операції XOR (виключне або) між символами відкритого тексту і гами. Для дешифрування використовується така ж операція XOR між шифротекстом і гамою. Операція XOR виконується над окремими бітами або символами.

Одноразове використання гами. Для забезпечення безпеки шифру гаммування вимагається використовувати гаму тільки один раз (одноразове використання). Кожен символ гами повинен бути використаний лише один раз, і після цього гама повинна бути змінена або видалена.

Синхронізація гами. Для успішного шифрування та дешифрування необхідна синхронізація між відправником і отримувачем щодо використання гами. Обидва боки повинні починати з однакової початкової точки гами та просуватися вперед у такті передачі символів.

Криптографічна стійкість. Шифр гаммування має бути стійким до криптоаналізу, тобто його має бути складно взламати без знання гами. Криптографічна стійкість вимагає, щоб навіть при знанні шифротексту та частини відкритого тексту, неможливо визначити гаму або отримати доступ до інших повідомлень.

Великий ключовий простір. Шифр гаммування використовує ключ для налаштування генератора гами. Ключ повинен мати достатню довжину, щоб забезпечити безпеку шифрування. Більший ключовий простір збільшує складність атаки з використанням перебору ключа.

Швидкість шифрування. Шифр гаммування, як потоковий шифр, може працювати дуже швидко, оскільки шифрування відбувається поодинокими символами або бітами. Це робить його ефективним для шифрування великих обсягів даних в реальному часі.

Розподілення гами. Для використання шифру гаммування у багатокористувацькому середовищі необхідне розподілення гами між усіма сторонами. Це вимагає безпечного обміну ключами та встановлення секретного зв'язку для передачі гами.

Застосування шифру гаммування передбачає дотримання цих властивостей, щоб забезпечити безпеку шифрування і захист конфіденційності даних.

2.2 Опис алгоритму на основі шифру гамування

Алгоритм на основі шифру гаммування використовує генератор гами для шифрування та дешифрування даних і складається із таких кроків:

Генерація гами:

Крок 1: Задати початковий ключ (key) для генератора гами.

Крок 2: Застосувати генератор гами для створення послідовності гами.

Крок 3: Зберегти генеровану гаму.

Шифрування:

Крок 1: Задати відкритий текст, який потрібно зашифрувати.

Крок 2: Взяти перший символ відкритого тексту.

Крок 3: Взяти перший символ гами.

Крок 4: Виконати операцію XOR між символом відкритого тексту та символом гами.

Крок 5: Зберегти отриманий шифрований символ.

Крок 6: Повторити кроки 2-5 для решти символів відкритого тексту.

Крок 7: Повернути шифрований текст.

Дешифрування:

Крок 1: Задати шифрований текст, який потрібно дешифрувати.

Крок 2: Взяти перший символ шифрованого тексту.

Крок 3: Взяти перший символ гами.

Крок 4: Виконати операцію XOR між символом шифрованого тексту та символом гами.

Крок 5: Зберегти отриманий символ відкритого тексту.

Крок 6: Повторити кроки 2-5 для решти символів шифрованого тексту.

Крок 7: Повернути відкритий текст.

На рис. 2.1 наведена схема алгоритму на основі шифру гаммування.

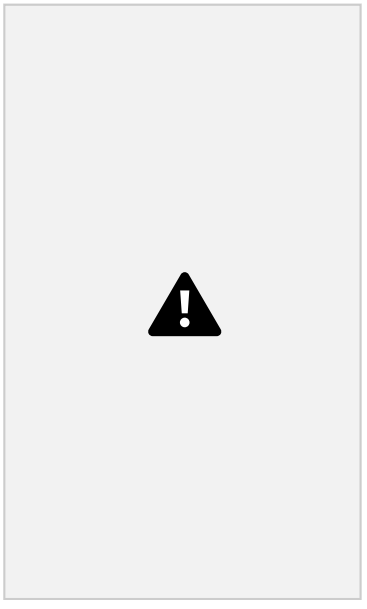


Рисунок 2.1 – Схема процесу шифрування та дешифрування на основі шифру гаммування

2.3 Побудова алгоритму для шифрування текстів з використанням шифру гаммування

Для побудови шифротексту з використанням шифру гаммування потрібно мати відкритий текст і генератор гами, який створює послідовність символів
критого тексту і
.
ння повідомлення.

Рисунок 2.2 – Структурна схема процесу шифрування повідомлення

У табл. 2.1 наведений приклад побудови шифротексту з використанням шифру гаммування:

Відкритий текст: "HELLO"

Гама: "10110"

Таблиця 2.1 – Побудова шифротексту

Відкритий текст	Гама	XOR	Шифротекст
H	1	1	I
E	0	0	E
L	1	1	M
L	1	1	M
O	0	0	O

Отриманий шифротекст: "IEMMO"

Таким чином, відкритий текст "HELLO" був зашифрований у шифротекст "IEMMO" з використанням двійкової гами "10110".

Важливо відзначити, що для безпеки шифру гаммування важливо мати достатньо стійку та випадкову гаму, а також забезпечити її одноразове

використання та правильну синхронізацію між відправником та отримувачем. Для підвищення безпеки шифрування можна використовувати багаторазове гаммування.

2.4 Побудова алгоритму для дешифрування текстів з використанням шифру гаммування

Для розшифрування зашифрованого повідомлення з використанням шифру гаммування потрібно мати шифротекст і генератор гами, який генерує ту саму послідовність символів гами, що була використана при шифруванні. Далі, застосовуючи операцію XOR між символами шифротексту і символами гами, отримуємо відповідні символи відкритого тексту.

На рис. 2.3 представлено структурну схему шифрування повідомлення.

Рисунок 2.3 – Структурна схема процесу шифрування повідомлення

Наприклад, якщо шифротекст "IEMMO" був отриманий за допомогою гами "10110", то розшифровуємо його з тою ж гамою, оскільки алгоритм гаммування є симетричний .

В таблиці 2.2 наведено процес дешифрування шифротексту "IEMMO" з використанням гами "10110".

Таблиця 2.2 – Розшифрування шифротексту "IEMMO"

Шифротекст	Гама XOR	Відкритий текст
I	1	H
E	0	E
M	1	L
M	1	L
O	0	O

Аналіз результатів табл.2 2 показує, що в результаті дешифрування отримано початковий відкритий текст "HELLO". Таким чином, шифротекст "IEMMO" був розшифрований у відкритий текст "HELLO" з використанням гами "10110".

Важливо зазначити, що правильне розшифрування буде тоді, коли відома гама та однакові послідовності символів гами, як при шифруванні.

3 РОЗРОБКА ПРОГРАМНОГО КОДУ ТА АНАЛІЗ РЕЗУЛЬТАТІВ 3.1

Постановка завдання

У дипломній роботі розроблено програми в програмному середовищі мови С++ для шифрування заданого вхідного тексту методом гамування з подальшим його розшифруванням.

Програма шифрування виконує наступні дії:

- зчитує текстове повідомлення із файлу message.txt;
- зчитує гама-ключ із файлу key.txt;
- здійснює шифрування зчитаного тексту методом гаммування;
- записує зашифроване повідомлення у текстовий файл encrypted.txt.

Для виконання завдання програма для шифрування повинна мати наступні функції:

readFromFile: функція для зчитування даних з файлу. Приймає рядок filename - ім'я файлу для зчитування. Повертає вектор символів, який містить дані з файлу.

writeToFile: функція для запису даних у файл. Приймає рядок filename - ім'я файлу для запису та вектор символів data, який містить дані для запису. Не повертає значення.

encryptMessage: функція для шифрування повідомлення. Приймає два вектори символів message і key - повідомлення та ключ. Повертає вектор символів, що містить зашифроване повідомлення.

У функції main необхідно виконати наступні дії:

- Встановити кодування вводу та виведення кирилиці для коректного відображення української мови в консолі;
- Задати шляхи до файлів для зчитування повідомлення, ключа, та запису зашифрованого повідомлення.
- Зчитати повідомлення з файлу за допомогою функції readFromFile.
- Зчитати ключ з файлу за допомогою функції readFromFile.

- Зашифрувати повідомлення за допомогою функції encryptMessage і отримати зашифроване повідомлення.
- Записати зашифроване повідомлення у файл за допомогою функції writeToFile.

Аналогічно програмі шифрування програма дешифрування виконує наступні дії:

- зчитує зашифроване програмою шифрування повідомлення із файлу encrypted.txt;
- зчитує гама-ключ із файлу key.txt;
- здійснює дешифрування зчитаного тексту методом гаммування;

- записує розшифроване повідомлення у текстовий файл decrypted.txt.

Для виконання завдання програма для дешифрування повинна мати наступні функції:

readFromFile: функція для зчитування даних з файлу. Приймає рядок filename - ім'я файлу для зчитування. Повертає вектор символів, який містить дані з файлу.

writeToFile: функція для запису даних у файл. Приймає рядок filename - ім'я файлу для запису та вектор символів data, який містить дані для запису. Не повертає значення.

decryptMessage: функція для шифрування повідомлення. Приймає два вектори символів encryptedMessage і key – попередньо зашифроване повідомлення та ключ. Повертає вектор символів, що містить розшифроване повідомлення. У функції main необхідно виконати наступні дії:

- Встановити кодування вводу та виведення кирилиці для коректного відображення української мови в консолі;
- Задати шляхи до файлів для зчитування попередньо зашифрованого повідомлення, ключа та запису розшифрованого повідомлення.
- Зчитати повідомлення з файлу за допомогою функції readFromFile. - Зчитати ключ з файлу за допомогою функції readFromFile.

- Розшифрувати повідомлення за допомогою функції decryptMessage і отримати розшифроване повідомлення.
- Записати розшифроване повідомлення у файл за допомогою функції writeToFile.

3.2 Опис розробленого програмного коду для шифрування повідомлення методом гаммування

На початку програми шифрування за допомогою директиви #include підключаємо необхідні для її роботи бібліотеки:

#include <iostream> – директива препроцесора для підключення бібліотеки <iostream>, яка містить класи та функції для введення/виведення даних.

#include <fstream> – директива препроцесора для підключення бібліотеки <fstream>, яка містить класи та функції для роботи з файлами.

#include <vector> – директива препроцесора для підключення бібліотеки <vector>, яка містить клас vector для роботи з динамічними масивами.

#include <windows.h> – директива препроцесора для підключення бібліотеки <windows.h>, яка надає доступ до функцій Windows для мови C++. У даному випадку, це потрібно для коректної обробки кирилиці.

using namespace std; – вказує на використання простору імен std, щоб не потрібно було вказувати префікс std:: перед об'єктами, які належать цьому простору імен.

vector<char> readFromFile(const string& filename) – оголошення функції readFromFile, яка зчитує дані з файлу і повертає їх у вигляді вектору vector<char>. ifstream file(filename.c_str(), ios::binary) – створення об'єкту ifstream для роботи з файлом. filename.c_str() перетворює рядок filename в масив символів C style, а ios::binary встановлює режим відкриття файлу у бінарному режимі. if (!file) {...} – перевірка, чи вдалося відкрити файл. Якщо відкриття не вдалося, виводиться повідомлення про помилку, і функція повертає порожній вектор. file.seekg(0, ios::end) – переміщення позиції читання у кінець файлу.

streampos fileSize = file.tellg() – отримання розміру файлу, шляхом отримання поточної позиції читання.

file.seekg(0, ios::beg) – переміщення позиції читання на початок файлу.

vector<char> buf er(fileSize) – створення вектору buffer з розміром, що дорівнює розміру файлу.

file.read(buf er.data(), fileSize) – зчитування даних з файлу у вектор buffer. file.close() – закриття файлу.

return buf er; – повернення вектору buffer з прочитаними даними.

void writeToFile(const string& filename, const vector<char>& data) – оголошення функції writeToFile, яка записує дані у файл.

ofstream file(filename.c_str(), ios::binary) – створення об'єкту ofstream для роботи з файлом. filename.c_str() перетворює рядок filename в масив символів C style, a ios::binary встановлює режим запису файлу у бінарному режимі.

if (!file) {...} – перевірка, чи вдалося відкрити файл. Якщо відкриття не вдалося, виводиться повідомлення про помилку, і функція повертається. file.write(data.data(), data.size())– запис даних з вектора data у файл.

if (file.fail()) {...} – перевірка, чи виникла помилка під час запису у файл. Якщо так, виводиться повідомлення про помилку, і файл закривається.

file.close() – закриття файлу.

vector<char> encryptMessage(const vector<char>& message, const vector<char>& key) – оголошення функції encryptMessage, яка шифрує повідомлення за допомогою шифру гамування.

vector<char> result; – створення порожнього вектору result, який буде містити зашифроване повідомлення.

if (key.empty() || message.empty()) {...} – перевірка, чи ключ або повідомлення є порожніми. Якщо так, виводиться повідомлення про помилку, і функція повертається.

for (size_t i = 0; i < message.size(); ++i) {...} – цикл, який проходиться по кожному символу повідомлення.

char encryptedChar = message[i] ^ key[i % key.size()]; – застосування операції побітового виключного "або" (^) до символу повідомлення та символу ключа для шифрування.

result.push_back(encryptedChar); – додавання зашифрованого символу до

вектору result.

return result; – повернення вектору result з зашифрованим повідомленням. int main() – оголошення головної функції main.

string messageFile = "C:\\DYPLOM\\Encryp_Gamma\\message.txt"; – оголошення рядка messageFile зі шляхом до файлу повідомлення.

string keyFile = "C:\\DYPLOM\\Encryp_Gamma\\key.txt"; – оголошення рядка keyFile зі шляхом до файлу ключа.

string encryptedFile = "C:\\DYPLOM\\Encryp_Gamma\\encrypted.txt"; – оголошення рядка encryptedFile зі шляхом до файлу зашифрованого повідомлення. vector<char> message = readFromFile(messageFile); – зчитування даних з файлу повідомлення та збереження їх у векторі message.

vector<char> key = readFromFile(keyFile); – зчитування даних з файлу ключа та збереження їх у векторі key.

vector<char> encryptedMessage = encryptMessage(message, key); – зашифрування повідомлення за допомогою ключа та збереження зашифрованого повідомлення у векторі encryptedMessage.

writeToFile(encryptedFile, encryptedMessage); – запис зашифрованого повідомлення у файл.

cout << "Шифрування завершено." << endl; – виведення повідомлення "Шифрування завершено." у консоль.

return 0; – повернення значення 0, що означає успішне завершення програми.

На рис. 3.1 представлено структурну схему алгоритму шифрування відкритих текстів на основі шифру гаммування.

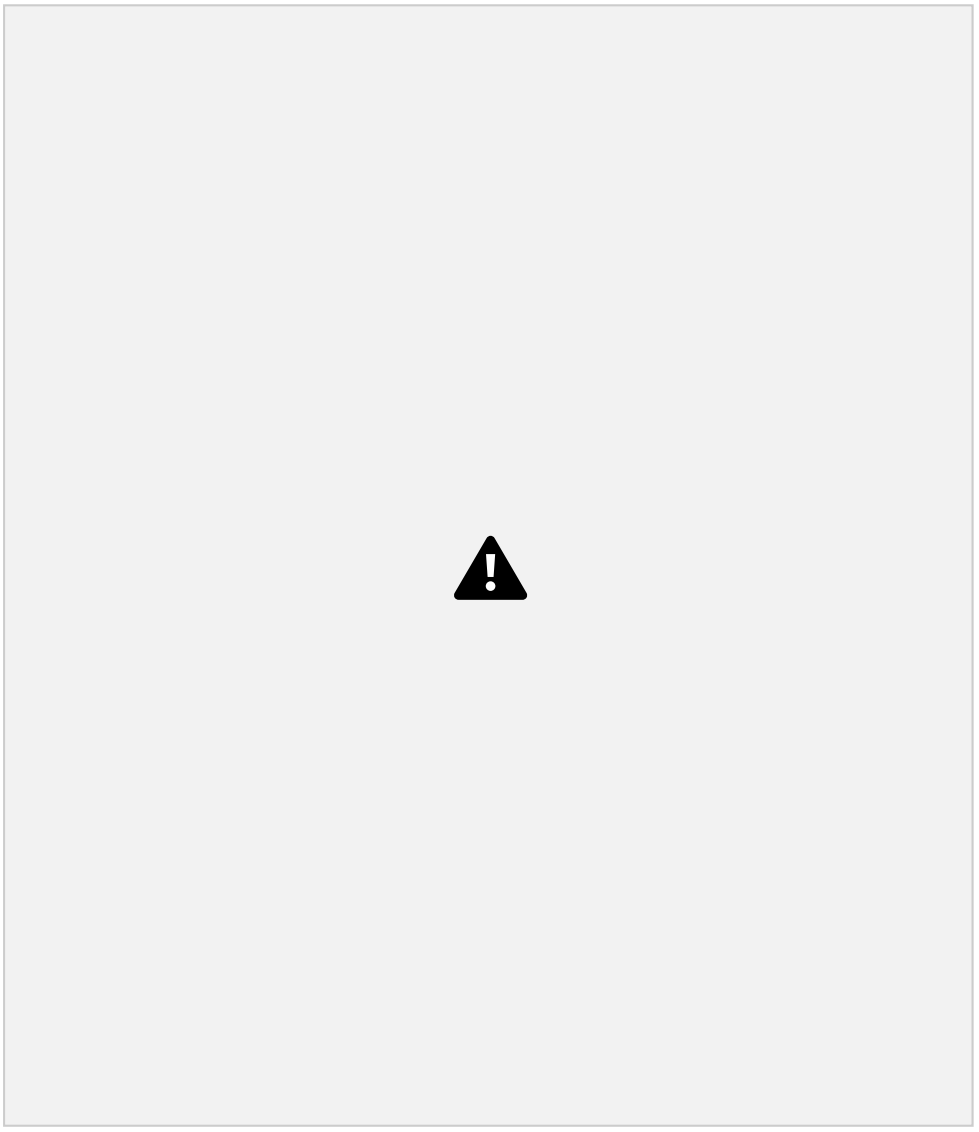


Рисунок 3.1 – Структурна схема алгоритму шифрування відкритих текстів на основі шифру гаммування

3.3 Опис розробленого програмного коду для дешифрування повідомлення методом гаммування

Принцип роботи програми для дешифрування повідомлень методом гаммування наступний:

Функція readFromFile зчитує дані з файлу у вигляді послідовності символів. Функція writeToFile записує дані у файл.

Функція decryptMessage виконує дешифрування повідомлення за допомогою ключа. Для цього вона використовує операцію побітового виключного "або" між кожним символом повідомлення та відповідним символом ключа.

vector<char> decryptMessage(const vector<char>& encryptedMessage, const vector<char>& key) – оголошення функції decryptMessage, яка розшифровує зашифроване повідомлення за допомогою ключа.

vector<char> result; – Оголошення вектору result, який буде містити розшифроване повідомлення.

if (key.empty() || encryptedMessage.empty()) {...} – перевірка, чи ключ або зашифроване повідомлення є порожніми. Якщо так, виводиться повідомлення про помилку, і функція повертає порожній вектор.

char decryptedChar = encryptedMessage[i] ^ key[i % key.size()]; – Розшифрування символу з зашифрованого повідомлення за допомогою ключа. Використовується операція побітового виключного "або".

result.push_back(decryptedChar); – додавання розшифрованого символу до вектору result.

return result; – повернення вектору result, який містить розшифроване повідомлення.

У функції main вказуються шляхи до файлів з зашифрованим повідомленням, ключем та файлу, в який буде записано розшифроване повідомлення.

string encryptedFile = "C:\\DYPLOM\\Decrypt_Gamma\\encrypted.txt"; – оголошення рядка encryptedFile з шляхом до зашифрованого файлу.

string keyFile = "C:\\DYPLOM\\Decrypt_Gamma\\key.txt"; – оголошення рядка keyFile з шляхом до файлу з ключем.

string decryptedFile = "C:\\DYPLOM\\Decrypt_Gamma\\decrypted.txt"; – оголошення рядка decryptedFile з шляхом до файлу, в який буде записано розшифроване повідомлення.

За допомогою функції readFromFile зчитується зашифроване повідомлення та ключ з файлів.

vector<char> encryptedMessage = readFromFile(encryptedFile); – зчитування зашифрованого повідомлення з файлу у вектор encryptedMessage за допомогою функції readFromFile.

vector<char> key = readFromFile(keyFile); – зчитування ключа з файлу у вектор key за допомогою функції readFromFile.

Виконується дешифрування повідомлення за допомогою функції decryptMessage.

vector<char> decryptedMessage = decryptMessage(encryptedMessage, key); – розшифрування зашифрованого повідомлення encryptedMessage за допомогою ключа key за допомогою функції decryptMessage. Результат зберігається у векторі decryptedMessage.

Розшифроване повідомлення записується у файл за допомогою функції writeToFile.

writeToFile(decryptedFile, decryptedMessage); – запис розшифрованого повідомлення decryptedMessage у файл decryptedFile за допомогою функції writeToFile.

На екран виводиться повідомлення про завершення розшифрування.

cout << "Розшифрування завершено." << endl; – виведення повідомлення про завершення розшифрування на екран.

return 0; – повернення значення 0, що вказує на успішне завершення програми.

Програма завершує роботу.

Структурну схему алгоритму дешифрування зашифрованих текстів на основі шифру гаммування представлено на рис. 3.2.

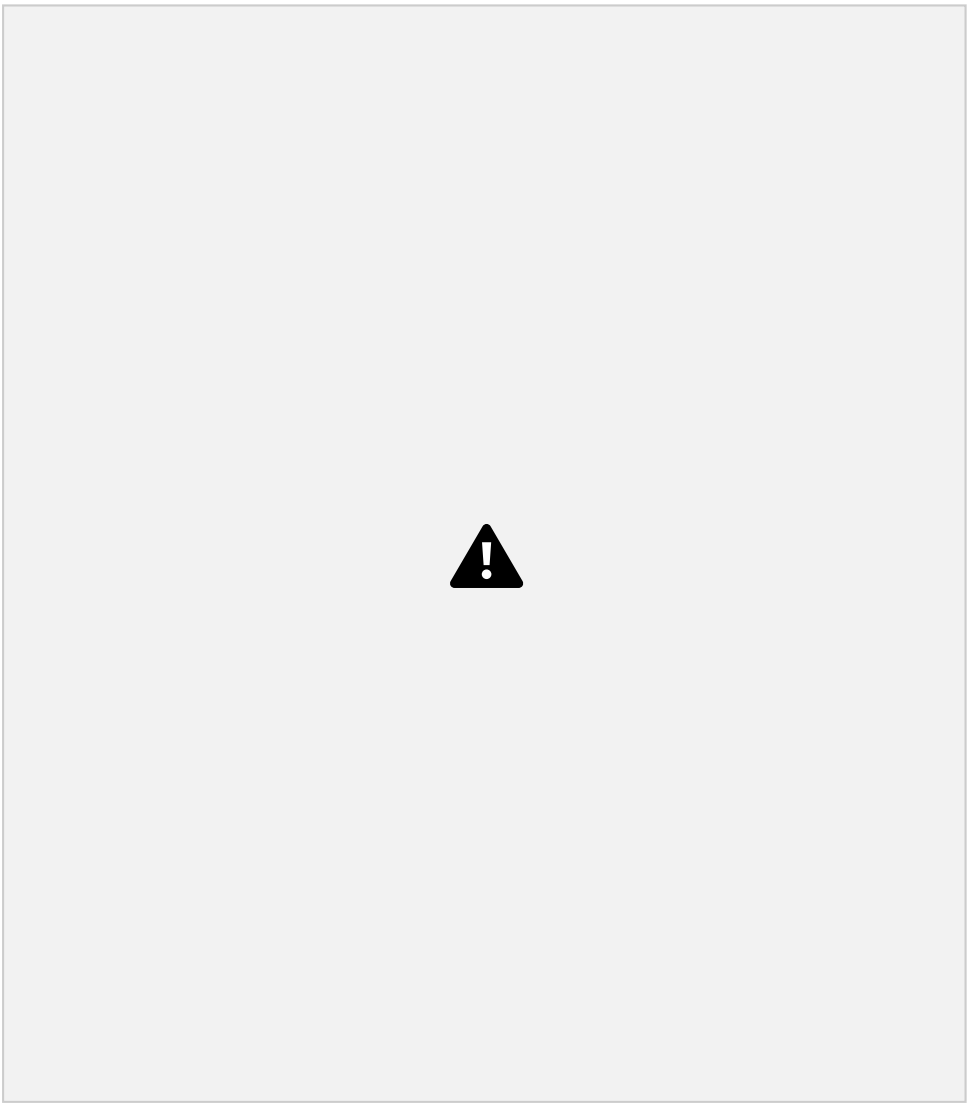


Рисунок 3.2 – Структурна схема алгоритму дешифрування зашифрованих текстів на основі шифру гаммування

3.4 Блок-схеми розроблених програм та окремих функцій

Алгоритми роботи основної функції та окремих власних функцій представлені блок-схемами.

На рис. 3.3 зображено блок-схему головної програми шифрування відкритого тексту методом гаммування.

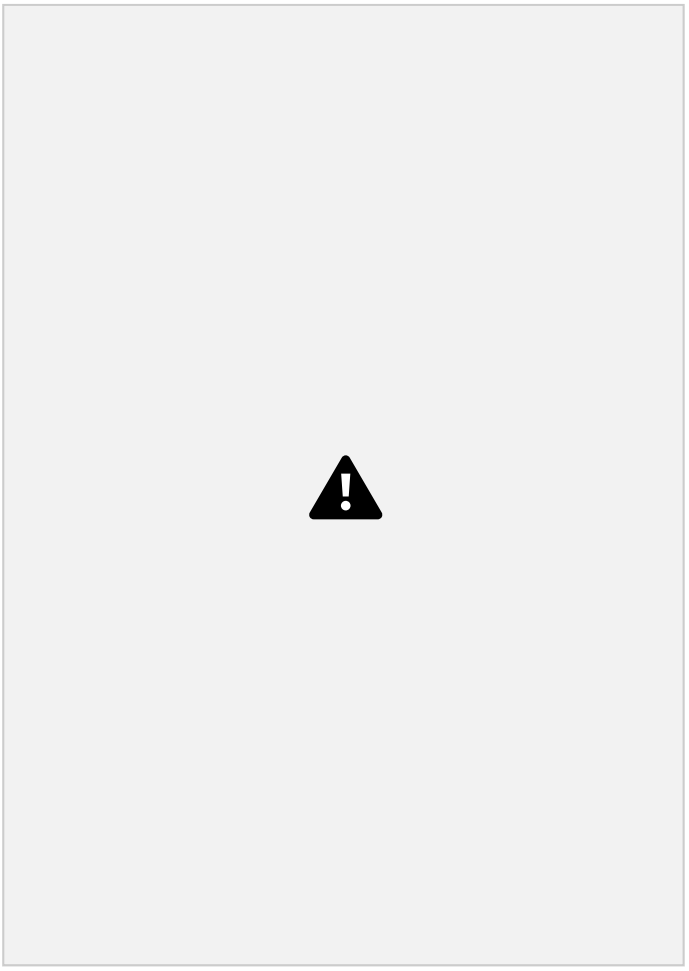


Рисунок 3.3 – Блок-схема головної програми шифрування відкритого тексту методом гаммування

На рис. 3.4 зображено блок-схему головної програми дешифрування шифротексту методом гаммування.

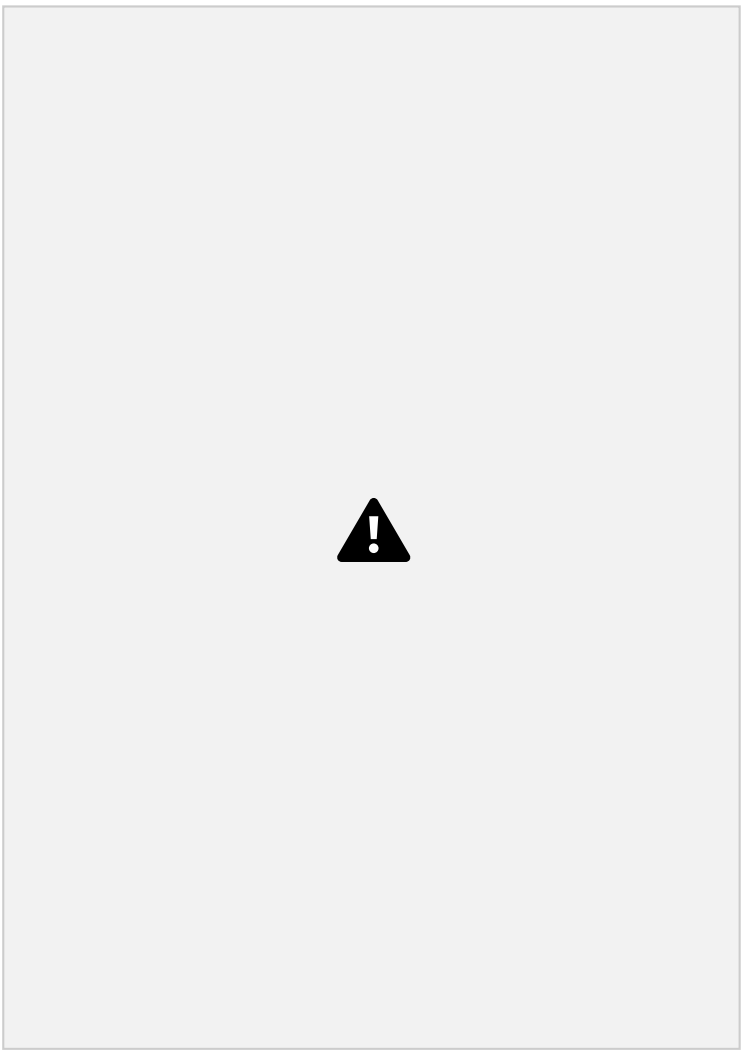


Рисунок 3.4 – Блок-схема головної програми дешифрування шифротексту методом гаммування

На рис. 3.5 зображено блок-схему функції шифрування відкритого тексту шифром гаммування.

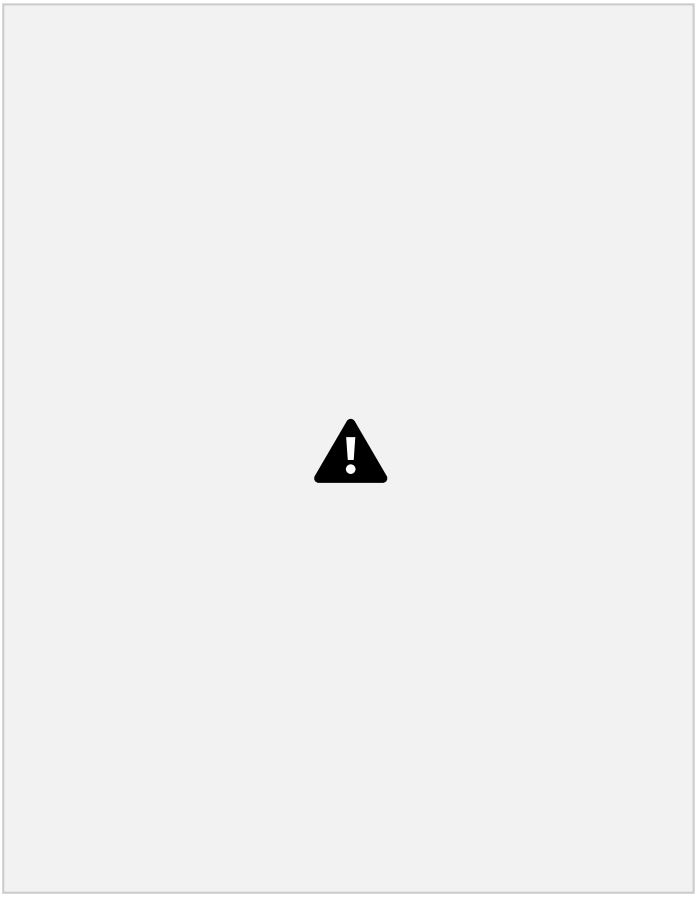


Рисунок 3.5 – Блок-схема функції для шифрування відкритого тексту шифром гаммування

На рис. 3.6 зображено блок-схему функції дешифрування шифротексту шифром гаммування.

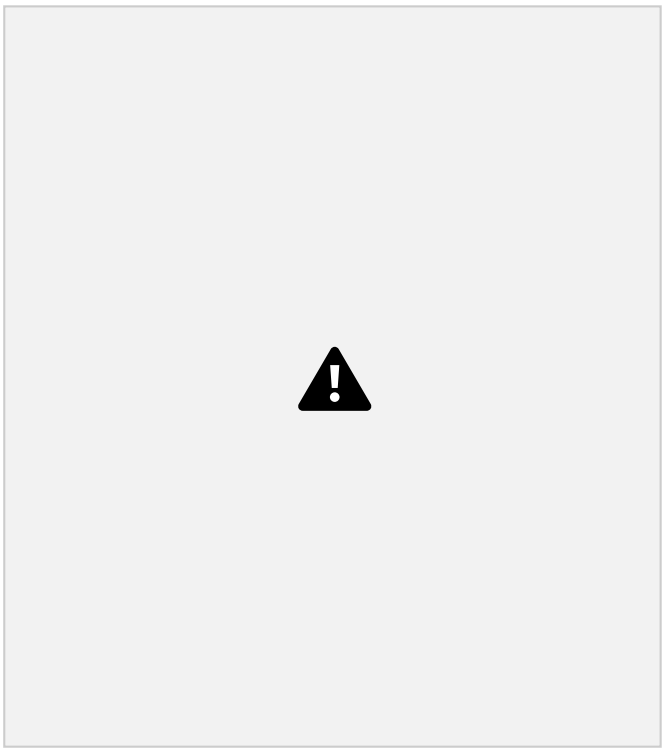


Рисунок 3.6 – Блок-схема функції для дешифрування шифротексту шифром гаммування

Повний текст програми шифрування відкритих текстів наведено в додатку А, а повний текст програми дешифрування ши фротекстів наведено в додатку Б.

3.5 Результати роботи програми шифрування відкритого повідомлення

Вхідними даними для програми шифрування є відкритий текст та випадкова двійкова гама. Вони читаються з файлів message.txt та key.txt. На рис. 3.7 показано вхідний текстовий файл (message.txt), вміст якого призначено для шифрування.

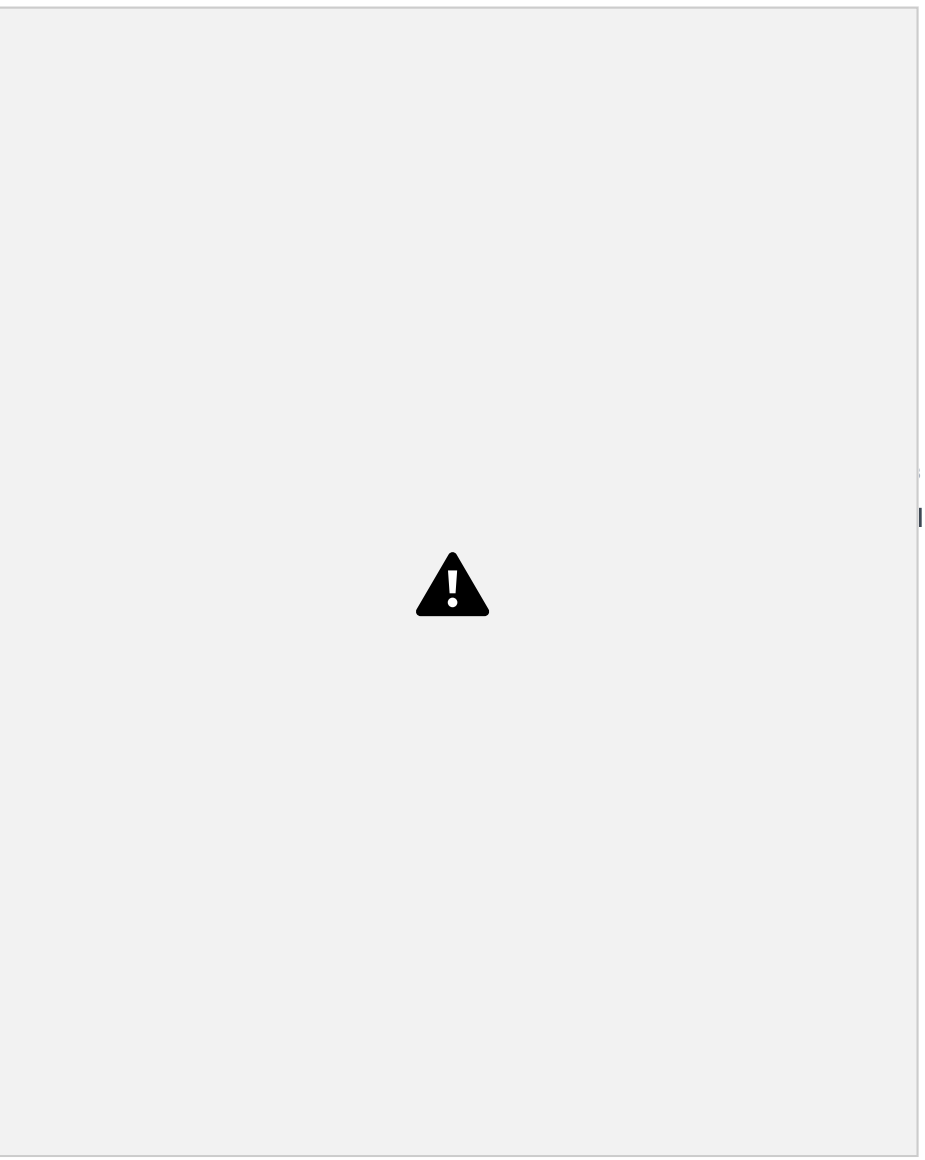
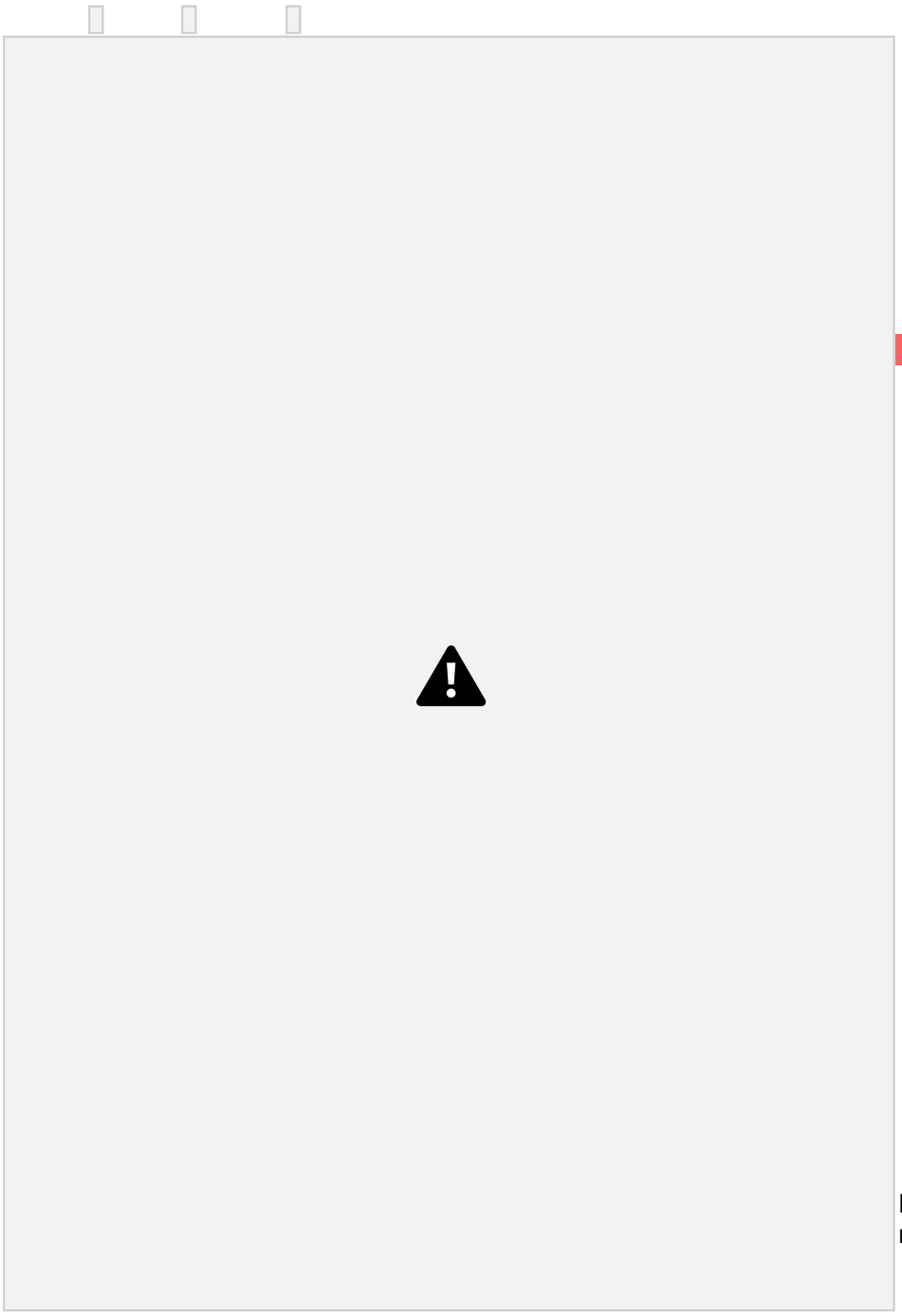


Рис. 3.7 – Початковий відкритий текст

На рис. 3.8 представлено текстовий файл (key.txt) із гама ключем для шифруваня вхідного текстового повідомлення.

Рис. 3.8 – Гамма ключ

Результатом роботи програми є одержання шифротексту записаного у файл encrypted.txt. На рис. 3.9 зображено вміст текстового файлу (encrypted.txt) після гама шифрування вхідного повідомлення.



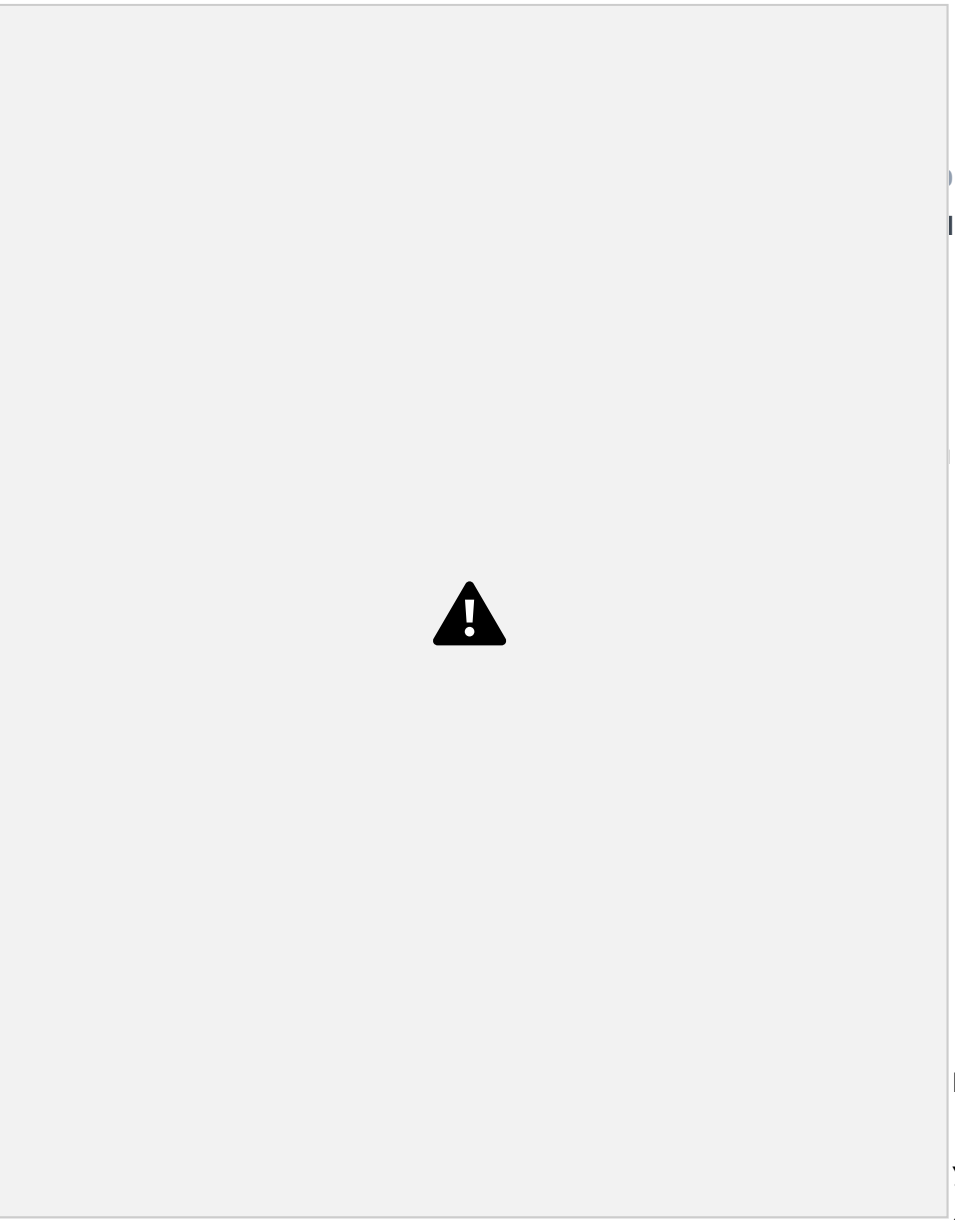
Підміна символів Коментарі

Рис. 3.9 – Вхідне текстове повідомлення після гамма шифрування

3.6 Результати роботи програми дешифрування шифротексту

На рис. 3.10 показано вміст текстового файлу (decrypted.txt) після дешифрування зашифрованого вхідного повідомлення.

Рис. 3.10 – Дешифроване текстове повідомлення



програми отримуємо

Рис. 3.11 – Коректне виконання програми

У випадку некоректного завершення роботи програми отримуємо повідомлення про помилку (рис. 3.12).

Рис. 3.12 – Повідомлення про помилку

Схожість

Схожість Цитати Посилання Вилучений

текст Підміна символів Коментарі

Назва документа: Клим_Христина_КН_320 **ID файлу:** 1015051587

Схожість

1 Студентська робота ID файлу: 1014439181 Навчальний заклад: Lviv Polytechnic National University 0.74% 4 Джерело

2 Студентська робота ID файлу: 1008261400 Навчальний заклад: National Aviation University 0.34% 46 Джерело

3 Студентська робота ID файлу: 1000097393 Навчальний заклад: Lviv Polytechnic National University 0.34% 4 Студентська робота ID файлу: 1013046526 Навчальний заклад: Lviv Polytechnic National University 0.24% 3 Джерело

5 Студентська робота ID файлу: 1005781358 Навчальний заклад: National Technical University of Ukraine "Kyi... 0.21% 6 Студентська робота ID файлу: 1014817082 Навчальний заклад: Interregional Academy of Personnel Managem... 0.21% 7 Студентська робота ID файлу: 1014863196 Навчальний заклад: State University Kyiv National Economic Univ... 0.21% 6 Джерело