\*\*Практическое задание по информационной безопасности\*\*

\*\*Курс: 3\*\*

\*\*Язык: C#\*\*

\*\*Тема: Комбинированная защита данных с использованием криптографии и стеганографии\*\*

---

### \*\*Цель задания\*\*

Реализовать приложение, которое шифрует текст, встраивает его в изображение с помощью стеганографии, а затем извлекает и расшифровывает данные, обеспечивая целостность и конфиденциальность.

---

### \*\*Задачи\*\*

1. \*\*Шифрование данных\*\*

- Зашифруйте введенный пользователем текст с использованием алгоритма \*\*AES-256-GCM\*\*.

- Ключ шифрования должен генерироваться на основе пароля пользователя через \*\*PBKDF2\*\* с солью.

- Выведите тег аутентификации (authentication tag) для проверки целостности.

2. \*\*Стеганография\*\*

- Внедрите зашифрованные данные и тег аутентификации в контейнер изображения формата \*\*BMP\*\* методом \*\*LSB (Least Significant Bit)\*\*.

- Убедитесь, что изменения в изображении незаметны визуально.

3. \*\*Извлечение и расшифровка\*\*

- Извлеките данные из изображения.

- Проверьте целостность с помощью тега аутентификации, используя \*\*постоянное сравнение во времени\*\* (constant-time comparison).

- Расшифруйте текст, если проверка успешна.

---

### \*\*Требования к реализации\*\*

- \*\*Безопасность\*\*:

- Используйте `SecureString` для обработки паролей.

- Генерируйте случайную соль для PBKDF2.

- Не сохраняйте ключи или пароли в открытом виде в памяти.

- \*\*Обработка ошибок\*\*:

- Обрабатывайте исключения (например, повреждение изображения, неверный пароль).

- Выводите информативные сообщения об ошибках.

- \*\*Без админских прав\*\*:

- Работа с файлами (изображениями) должна выполняться в пользовательских директориях.

- Используйте только стандартные библиотеки .NET или NuGet-пакеты, не требующие установки (например, `System.Security.Cryptography`).

---

### \*\*Примерный алгоритм работы\*\*

1. \*\*Шифрование\*\*:

- Пользователь вводит текст и пароль.

- Генерация ключа: `Rfc2898DeriveBytes` + случайная соль.

- Шифрование AES-GCM: получение зашифрованных данных и тега.

2. \*\*Стеганография\*\*:

- Чтение BMP-файла в байтовый массив.

- Внедрение данных в младшие биты пикселей (LSB).

- Сохранение нового изображения.

3. \*\*Расшифровка\*\*:

- Извлечение битов из BMP-файла.

- Проверка тега аутентификации через `CryptographicOperations.FixedTimeEquals`.

- Расшифровка данных, если проверка пройдена.

---

### \*\*Проверка задания\*\*

1. Протестируйте приложение на разных изображениях.

2. Убедитесь, что:

- Изменения в изображении незаметны.

- Расшифровка невозможна при изменении пароля или повреждении тега.

- Сравнение тегов выполняется за постоянное время.

---

### \*\*Подсказки\*\*

- Для работы с AES-GCM используйте `AesGcm` из `System.Security.Cryptography`.

- Для обработки BMP-файлов можно прочитать байты через `FileStream` и манипулировать ими напрямую.

- Пример LSB-встраивания: замена младших битов RGB-каналов пикселей.

---

\*\*Оценка\*\*:

- Корректность шифрования/расшифровки — 40%.

- Реализация стеганографии — 30%.

- Обработка ошибок и безопасность — 30%.

---

Это задание требует глубокого понимания криптографии и стеганографии, а также навыков работы с памятью и файлами в C#. Удачи!