\*\*Практическое задание: "Разработка и тестирование системы шифрования данных в C# с анализом уязвимостей"\*\*

---

### \*\*Цель:\*\*

Реализовать кастомный алгоритм симметричного шифрования в C# и провести анализ его уязвимостей, предложив улучшения для повышения безопасности.

---

### \*\*Условия:\*\*

1. \*\*Разработка алгоритма:\*\*

- Создайте \*\*оригинальный алгоритм симметричного шифрования\*\* (например, на основе замены битов, перестановок или комбинации простых операций).

- Алгоритм должен включать:

- Генерацию ключа (например, из строки пользователя).

- Функцию шифрования/дешифрования текстовых данных.

- Обработку возможных ошибок (например, некорректный ключ).

2. \*\*Требования к алгоритму:\*\*

- Не используйте встроенные криптографические библиотеки (.NET Crypto API) для реализации основных операций (только для вспомогательных задач, если необходимо).

- Убедитесь, что алгоритм:

- Защищен от простых атак (например, частотный анализ).

- Поддерживает обработку данных переменной длины.

3. \*\*Тестирование и анализ уязвимостей:\*\*

- Проведите тестирование алгоритма на следующие аспекты:

- \*\*Стойкость к атакам "known-plaintext":\*\* Попытайтесь восстановить ключ, зная пару "открытый-зашифрованный" текст.

- \*\*Реакция на изменение ключа:\*\* Проверьте, как алгоритм работает с ключами разной длины или с ошибками в ключе.

- \*\*Анализ времени выполнения:\*\* Оцените, может ли время шифрования/дешифрования выдать информацию о структуре алгоритма.

- Используйте \*\*только собственные методы тестирования\*\* (без сторонних утилит для криптоанализа).

4. \*\*Ограничения:\*\*

- Код должен выполняться в среде разработки (например, Visual Studio или Rider) без прав администратора.

- Запрещено использовать сторонние библиотеки для криптографии (кроме стандартных классов C# для работы с файлами/строками).

---

### \*\*Требования к отчету:\*\*

1. \*\*Описание алгоритма:\*\*

- Структура алгоритма, этапы шифрования/дешифрования.

- Метод генерации ключа и его особенности.

2. \*\*Результаты тестирования:\*\*

- Результаты тестов на уязвимости (например, успешность атаки "known-plaintext").

- Визуализация или описание временных характеристик алгоритма.

3. \*\*Анализ уязвимостей:\*\*

- Определите слабые места вашего алгоритма.

- Предложите способы их устранения (например, добавление соли, усложнение перестановок).

4. \*\*Рекомендации по улучшению:\*\*

- Как можно сделать алгоритм более безопасным?

- Сравните ваш алгоритм с стандартными методами (AES, DES) и укажите преимущества/недостатки.

5. \*\*Исходный код:\*\*

- Приложите реализацию алгоритма на C# с комментариями.

---

### \*\*Важные условия:\*\*

- \*\*Этические аспекты:\*\* Не публикуйте алгоритм в открытых источниках до анализа его уязвимостей.

- \*\*Оригинальность:\*\* Не повторяйте известные алгоритмы (например, Caesar, Vigenère).

- \*\*Документация:\*\* В отчете четко обоснуйте выбор методов и логику алгоритма.

---

### \*\*Почему задание не решается напрямую?\*\*

- Требуется \*\*оригинальная реализация алгоритма\*\*, которую я не могу предугадать.

- Анализ уязвимостей зависит от конкретной реализации студента, а не от общих принципов.

- Творческий подход к тестированию и улучшениям делает задание уникальным для каждого участника.

---

### \*\*Что развивает задание:\*\*

- \*\*Программирование на C#\*\* (работа с алгоритмами, обработка данных).

- \*\*Криптографические принципы\*\* (понимание принципов безопасности алгоритмов).

- \*\*Анализ уязвимостей\*\* (предвидение атак и улучшение защиты).

- \*\*Критическое мышление\*\* (поиск слабых мест в собственной разработке).

\*\*Примечание:\*\* Могу помочь с общими советами по криптографии или C#, но конкретное решение требует самостоятельного творческого подхода.