АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ ДЕШИФРОВАНИЯ ТРАНСПОЗИЦИОННОГО ШИФРА В PYTHON

Валимов Мунир Даниярович[1], Ахмедова Ирода Нурмухаммедовна[2]

[1,2] Ташкентский Международный университет образования (TIUE)
E-mail: [1]akhmedovairoda11@gmail.com; [2]munirvalimov10@gmail.com
DOI: 10.61587/mmit.tiue.uz.v1i1.392

Аннотация. В статье проводится анализ производительности дешифрования транспозиционного шифра, реализованного на языке Python. Выполнено сравнение времени шифрования и дешифрования, выявлены причины увеличения вычислительных затрат при восстановлении данных. Также рассмотрен частотный анализ символов, подтверждающий уязвимость перестановочных шифров к методам криптоанализа, и исправлена ошибка реализации алгоритма дешифрования.

Ключевые слова: *транспозиционный шифр, дешифрование, криптография, Python, производительность алгоритмов, частотный анализ, криптоанализ, информационная безопасность, перестановочные шифры, инверсия ключа.*

DECRYPTION PERFORMANCE ANALYSIS OF A TRANSPOSITION CIPHER IN PYTHON

Valimov Munir Daniyarovich[1], Akhmedova Iroda Nurmukhamedovna[2]

[1,2] Tashkent International University of Education (TIUE)
E-mail: [1]akhmedovairoda11@gmail.com; [2]munirvalimov10@gmail.com

Abstract. This article analyzes the decryption performance of a transposition cipher implemented in Python. Encryption and decryption times are compared, and the reasons for the increased computational costs during data recovery are identified. A frequency analysis of symbols is also considered, confirming the vulnerability of transposition ciphers to cryptanalytic methods, and an implementation error in the decryption algorithm is corrected.

Keywords: *transposition cipher, decryption, cryptography, Python, algorithm performance, frequency analysis, cryptanalysis, information security, permutation ciphers, key inversion.*

Введение

Процесс дешифрования является зеркальным, но часто более сложным этапом в работе криптографических систем. В области кибербезопасности важно не только быстро зашифровать данные, но и обеспечить их корректное и эффективное восстановление легитимным пользователем. Изучение дешифрования позволяет выявить «узкие места» в алгоритмах и оценить их устойчивость к внешним атакам. Целью работы является сравнительный анализ эффективности дешифрования по сравнению с шифрованием, а также проверка теоретического положения о сохранении частотных характеристик символов в перестановочных шифрах.

Методология

Исследование базируется на функции columnar_decrypt, реализованной на Python. Методика включает три этапа:

1. Замер времени восстановления исходного текста из зашифрованного массива данных объемом 10 000 знаков.

2. Проведение частотного анализа: подсчет распределения букв в исходном и зашифрованном текстах для проверки идентичности их статистических профилей.
3. Проверка корректности кода (Code Review) для выявления логических ошибок при работе с индексами ключа. Все измерения проводились в идентичной программной среде (Python 3.12, Ubuntu 24.04), что позволило получить объективные сравнительные данные.

Результаты

Экспериментально установлено, что дешифровка работает существенно медленнее шифрования. При обработке текста объемом 10 000 символов процесс шифрования занял 0,467 мс, в то время как дешифрование — 0,991 мс. Разница в два раза объясняется тем, что при восстановлении текста программе требуется выполнять сложные вычисления для определения позиции каждого символа в исходной матрице, в то время как шифрование происходит путем последовательного считывания столбцов. Важным результатом стал частотный анализ: гистограммы распределения символов до и после шифрования оказались абсолютно идентичными. Это подтверждает теоретическую уязвимость перестановочных шифров перед методами статистического анализа. При изучении исходников обнаружено серьёзный баг, который полностью ломает функцию дешифрования. В момент, когда пользователь соглашается расшифровать текст, программа пытается выполнить строку:

```
ordered_key = sorted.rows
```

Здесь сразу две проблемы. Во-первых, переменная sorted не определена нигде в программе. Во-вторых, даже если бы она существовала, у неё нет никакого атрибута rows. В результате Python выдаёт ошибку, и программа завершается аварийно.

Правильный код должен выглядеть так:

```
inverse_key = [0] * len(key)

for i, k in enumerate(key):

    inverse_key[k] = i
```

Этот фрагмент создаёт новый массив той же длины и заполняет его обратными индексами, что позволяет корректно восстановить исходный порядок столбцов.

Расшифровка: алгоритм инверсии ключа

Процесс расшифровки столбцового транспозиционного шифра основан на построении обратной перестановки к исходному ключу. Как отмечают авторы учебника по классической криптографии, «шифры перестановки не изменяют состав символов сообщения, а лишь меняют их порядок, поэтому частотные характеристики открытого и зашифрованного текстов совпадают» [3]. Именно это свойство определяет как уязвимость данного класса шифров, так и особую роль корректной реализации алгоритма дешифрования.

Пусть задан ключ перестановки key = [k0, k1, ..., k(n−1)], где ki □ {0, ..., n−1} и все значения различны. При шифровании столбец с индексом i записывается на позицию ki в зашифрованный текст. Для дешифрования необходимо построить обратную перестановку inverse_key такую, что inverse_key[ki] = i для каждого i. Это позволяет однозначно восстановить исходный порядок столбцов матрицы. Реализация на Python выглядит следующим образом:

```
inverse_key = [0] * len(key)

for i, k in enumerate(key): inverse_key[k] = i
```

Полная функция дешифрования с использованием корректной инверсии ключа принимает вид:

```
def columnar_decrypt(encrypted, key):

    n = len(key); n_rows = len(encrypted) // n

    matrix = [['']*n for _ in range(n_rows)]

    index = 0

    for k in key:

        for i in range(n_rows): matrix[i][k] = encrypted[index]; index += 1

    return ''.join(''.join(row) for row in matrix)
```

Ключевое отличие исправленной версии состоит в том, что символы зашифрованного текста записываются непосредственно в позицию matrix[i][k], соответствующую исходному столбцу k. Это обеспечивает точное восстановление двумерной матрицы, из которой затем построчно считывается открытый текст. Как подчёркивает Schneier, «корректность реализации криптографического алгоритма не менее важна, чем его теоретическая стойкость» [2]. Ошибка в индексировании при обратной перестановке полностью нарушает процесс восстановления данных, даже при верном ключе.

Более высокая вычислительная сложность дешифрования по сравнению с шифрованием объясняется необходимостью двухэтапного обхода матрицы: сначала заполнение по столбцам в порядке ключа, затем чтение по строкам. При шифровании же достаточно одного прохода по столбцам. Мао В. указывает, что «асимметрия трудоёмкости прямого и обратного криптографических преобразований является характерной особенностью многих симметричных алгоритмов» [4]. Экспериментально полученное двукратное превышение времени дешифрования (0,991 мс против 0,467 мс) полностью согласуется с данной теоретической оценкой.

Практический анализ частотного распределения символов подтверждает теоретическое положение о том, что перестановочные шифры сохраняют статистический профиль открытого текста. Согласно материалам учебного пособия по криптоанализу классических шифров, «при дешифровании текста используют частотные характеристики открытого текста; однако для получения устойчивой картины длина послания должна быть существенно больше ключа» [3]. Это делает транспозиционные шифры уязвимыми при достаточной длине перехваченного сообщения.

Дискуссия

Дешифрование методом перестановки является более ресурсоемкой операцией, чем шифрование, что необходимо учитывать при проектировании систем реального времени. Несмотря на простоту реализации, данные шифры обладают низкой криптостойкостью из-за неизменности частотности символов.

Практическая ценность работы заключается в исправлении критических ошибок реализации и обосновании необходимости комбинирования перестановки с другими методами защиты (например, заменой).

Список литературы

1. Саломаа А. Криптография с открытым ключом. — М.: Мир, 1996. — 318 с.
2. Schneier B. Applied Cryptography. Protocols, Algorithms, and Source Code in C. 2nd Edition. — Wiley, 1996. — 758 p.
3. Перестановочный шифр // Википедия. — URL: https://ru.wikipedia.org/wiki/Перестановочный_шифр (дата обращения: 20.05.2025). — Раздел «Дешифрование».
4. Мао В. Современная криптография: теория и практика / пер. с англ. Д. А. Клюшина. — М.: Вильямс, 2005. — 763 с. — ISBN 978-5-8459-0847-6.