

Правительство Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
(НИУ ВШЭ)

Московский институт электроники и математики им. А.Н. Тихонова

ОТЧЕТ
О ПРАКТИЧЕСКОЙ РАБОТЕ № 2
по дисциплине «Математические основы защиты информации»
МАТРИЧНЫЙ ШИФР ХИЛЛА

Студент гр. БИБ 191
И.Г. Тюрин
г.

Руководитель
Заведующий кафедрой информационной
безопасности киберфизических систем
канд. техн. наук, доцент
_____ О.О. Евсютин
г.

Москва 2021

СОДЕРЖАНИЕ

1. Задание на практическую работу	3
2. Краткая теоретическая часть	4
2.1. Описание шифров	4
2.2. Методы криптоанализа шифров	5
3. Примеры шифрования	6
4. Программная реализация шифров	9
5. Примеры криптоанализа	11
6. Выводы о проделанной работе	13
7. Список использованных источников	14

1. Задание на практическую работу

Целью работы является приобретение навыков программной реализации и криптоанализа применительно к блочному шифру Хилла.

В рамках практической работы необходимо выполнить следующее:

- 1) написать программную реализацию следующих шифров:
 - шифр Хилла;
 - рекуррентный шифр Хилла;
- 2) изучить методы криптоанализа матричных шифров с использованием дополнительных источников;
- 3) провести криптоанализ данных шифров;
- 4) подготовить отчет о выполнении работы.

2. Краткая теоретическая часть

2.1. Описание шифров

Шифр Хилла – блочный шифр, основанный на матричных преобразованиях. В качестве ключа выступает квадратная матрица $n \times n$. Все элементы принадлежат кольцу классов вычетов по модулю мощности алфавита, определитель этой матрицы должен принадлежать группе обратимых элементов этого кольца. Открытый текст разбивается на блоки длины n , после чего каждый блок представляется в виде n -мерного вектор-столбца (символы открытого текста заменяются их номерами в алфавите при отсчёте с нуля). При шифровании каждый блок умножается справа на матрицу ключа, за счёт чего получается блок шифртекста:

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = E_k(X) = K \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

Для расшифрования применяется тот же алгоритм с тем отличием, что производится умножение на обратную матрицу ключа:

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = D_k(Y) = K^{-1} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Рекуррентный шифр Хилла является усилением шифра Хилла, использующим новый ключ для каждого шифруемого блока открытого текста. Первые два ключа задаются вручную, все последующие ключи вычисляются по следующим формулам:

$$K_{i+1} = K_i K_{i-1} \text{ (используется при зашифровании)}$$

$$K_{i+1}^{-1} = K_{i-1}^{-1} K_i^{-1} \text{ (используется при расшифровании)}$$

2.2. Методы криптоанализа шифров

Криптоанализ шифра Хилла имеет довольно большие ограничения. Применять метод грубой силы в высшей степени неразумно, поскольку пространство ключей данного шифра теоретически бесконечно, а на практике оно ограничивается только возможностями пользователей вычислять определители матриц и обратные матрицы при больших размерах n . Частотный криптоанализ также имеет свои пределы, поскольку самым эффективным его применением является анализ n -грамм, что быстро теряет смысл с ростом n , так как чем длиннее строка, тем меньше вероятность того, что она в точности повторится в тексте более одного раза. Таким образом криптоанализ на основе только шифртекста весьма затруднителен, хотя и возможен при достаточно небольших значениях n . Вследствие этого криптоанализ шифра Хилла приобретает практический смысл только при знании n и/или наличия определённой части открытого текста.

3. Примеры шифрования

1. Шифр Хилла

Открытый текст: ARTISTICALLY

$$\text{Ключ: } \begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}$$

Представляем открытый текст в виде номеров букв в алфавите при отсчёте с нуля:

[0, 17, 19, 8, 18, 19, 8, 2, 0, 11, 11, 24]

Разбиваем открытый текст на блоки по 3 символа, так как ключевая матрица имеет размер 3×3 , и представляем блоки в виде вектор-столбцов:

$$X_1 = \begin{pmatrix} 0 \\ 17 \\ 19 \end{pmatrix}, X_2 = \begin{pmatrix} 8 \\ 18 \\ 19 \end{pmatrix}, X_3 = \begin{pmatrix} 8 \\ 2 \\ 0 \end{pmatrix}, X_4 = \begin{pmatrix} 11 \\ 1 \\ 24 \end{pmatrix}$$

К каждому вектор-столбцу применяем операцию шифрования:

$$Y_1 = \begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 0 \\ 17 \\ 19 \end{pmatrix} = \begin{pmatrix} 11 \\ 20 \\ 2 \end{pmatrix}, Y_2 = \begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 8 \\ 18 \\ 19 \end{pmatrix} = \begin{pmatrix} 5 \\ 10 \\ 23 \end{pmatrix},$$
$$Y_3 = \begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 8 \\ 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 18 \\ 6 \\ 12 \end{pmatrix}, Y_4 = \begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 11 \\ 1 \\ 24 \end{pmatrix} = \begin{pmatrix} 16 \\ 13 \\ 13 \end{pmatrix}$$

Собрав полученные блоки шифртекста в единый текст и преобразовав числа к буквам, получим шифртекст LUCFKXSGMQNN

Для расшифрования найдём матрицу, обратную к ключевой:

$$K^{-1} = |K|^{-1} K^* = 25^{-1} \begin{pmatrix} 18 & 21 & 16 \\ 5 & 18 & 5 \\ 5 & 14 & 18 \end{pmatrix} = 25 \begin{pmatrix} 18 & 21 & 16 \\ 5 & 18 & 5 \\ 5 & 14 & 18 \end{pmatrix} = \begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix}$$

$$|K| - \det(A)$$

$|K|^{-1}$ – обратное элемент к $\det(A)$ по модулю

K^* - союзная матрица.

К каждому из полученных выше вектор-столбцов блоков шифртекста применим операцию расшифрования:

$$X_1 = \begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix} \begin{pmatrix} 11 \\ 20 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 17 \\ 19 \end{pmatrix}, X_2 = \begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix} \begin{pmatrix} 8 \\ 18 \\ 19 \end{pmatrix} = \begin{pmatrix} 8 \\ 18 \\ 19 \end{pmatrix},$$

$$X_3 = \begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix} \begin{pmatrix} 18 \\ 6 \\ 12 \end{pmatrix} = \begin{pmatrix} 8 \\ 2 \\ 0 \end{pmatrix}, X_4 = \begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix} \begin{pmatrix} 11 \\ 1 \\ 24 \end{pmatrix} = \begin{pmatrix} 11 \\ 1 \\ 24 \end{pmatrix}$$

Собрав полученные блоки открытого текста в единый текст и преобразовав числа к буквам, получим открытый текст ARTISTICALLY

2. Рекуррентный шифр Хилла

Открытый текст: ARITHMETICAL

Ключ: $\begin{pmatrix} 4 & 3 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 16 & 3 \\ 21 & 17 \end{pmatrix}$

Представляем открытый текст в виде номеров букв в алфавите при отсчёте с нуля:

$$[0, 17, 8, 19, 7, 12, 4, 19, 8, 2, 0, 11]$$

В открытом тексте 12 символов.

Разбиваем открытый текст на блоки по 2 символа, так как ключевая матрица имеет размер 2×2 , и представляем блоки в виде вектор-столбцов:

$$X_1 = \begin{pmatrix} 0 \\ 17 \end{pmatrix}, X_2 = \begin{pmatrix} 8 \\ 19 \end{pmatrix}, X_3 = \begin{pmatrix} 7 \\ 12 \end{pmatrix}, X_4 = \begin{pmatrix} 4 \\ 19 \end{pmatrix}, X_5 = \begin{pmatrix} 8 \\ 2 \end{pmatrix}, X_6 = \begin{pmatrix} 0 \\ 11 \end{pmatrix}$$

Блоков 6, следовательно необходимо вычислить 4 ключа на основе первых двух:

$$K_3 = \begin{pmatrix} 16 & 3 \\ 21 & 17 \end{pmatrix} \begin{pmatrix} 4 & 3 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 15 & 25 \\ 24 & 3 \end{pmatrix}, K_4 = \begin{pmatrix} 15 & 25 \\ 24 & 3 \end{pmatrix} \begin{pmatrix} 4 & 3 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 11 & 1 \\ 5 & 22 \end{pmatrix}$$

$$K_5 = \begin{pmatrix} 11 & 1 \\ 5 & 22 \end{pmatrix} \begin{pmatrix} 15 & 25 \\ 24 & 3 \end{pmatrix} = \begin{pmatrix} 7 & 18 \\ 5 & 9 \end{pmatrix}, K_6 = \begin{pmatrix} 7 & 18 \\ 5 & 9 \end{pmatrix} \begin{pmatrix} 11 & 1 \\ 5 & 22 \end{pmatrix} = \begin{pmatrix} 11 & 13 \\ 22 & 21 \end{pmatrix}$$

К каждому вектор-столбцу применяем операцию шифрования:

$$Y_1 = \begin{pmatrix} 4 & 3 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 17 \end{pmatrix} = \begin{pmatrix} 25 \\ 17 \end{pmatrix}, Y_2 = \begin{pmatrix} 16 & 3 \\ 21 & 17 \end{pmatrix} \begin{pmatrix} 8 \\ 19 \end{pmatrix} = \begin{pmatrix} 3 \\ 16 \end{pmatrix}, Y_3 = \begin{pmatrix} 16 & 3 \\ 21 & 17 \end{pmatrix} \begin{pmatrix} 7 \\ 12 \end{pmatrix} = \begin{pmatrix} 15 \\ 22 \end{pmatrix},$$

$$Y_4 = \begin{pmatrix} 15 & 25 \\ 24 & 3 \end{pmatrix} \begin{pmatrix} 4 \\ 19 \end{pmatrix} = \begin{pmatrix} 11 \\ 22 \end{pmatrix}, Y_5 = \begin{pmatrix} 11 & 1 \\ 5 & 22 \end{pmatrix} \begin{pmatrix} 8 \\ 2 \end{pmatrix} = \begin{pmatrix} 14 \\ 6 \end{pmatrix}, Y_6 = \begin{pmatrix} 7 & 18 \\ 5 & 9 \end{pmatrix} \begin{pmatrix} 0 \\ 11 \end{pmatrix} = \begin{pmatrix} 13 \\ 23 \end{pmatrix}$$

Собрав полученные блоки шифртекста в единый текст и преобразовав числа к буквам, получим шифртекст ZRDXPDEJKONS

Для расшифрования найдём матрицы, обратные к ключам:

$$K_1^{-1} = |K_1|^{-1} K_1^* = 1^{-1} \begin{pmatrix} 1 & 23 \\ 25 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 23 \\ 25 & 4 \end{pmatrix},$$

$$K_2^{-1} = |K_2|^{-1} K_2^* = 1^{-1} \begin{pmatrix} 17 & 23 \\ 5 & 16 \end{pmatrix} = \begin{pmatrix} 17 & 23 \\ 5 & 16 \end{pmatrix}$$

Найдём остальные обратные матрицы ключей:

$$K_3^{-1} = \begin{pmatrix} 1 & 23 \\ 25 & 4 \end{pmatrix} \begin{pmatrix} 24 & 9 \\ 11 & 4 \end{pmatrix} = \begin{pmatrix} 17 & 23 \\ 20 & 7 \end{pmatrix}, K_4^{-1} = \begin{pmatrix} 24 & 9 \\ 11 & 4 \end{pmatrix} \begin{pmatrix} 17 & 23 \\ 20 & 7 \end{pmatrix} = \begin{pmatrix} 16 & 17 \\ 7 & 21 \end{pmatrix}$$

$$K_5^{-1} = \begin{pmatrix} 17 & 23 \\ 20 & 7 \end{pmatrix} \begin{pmatrix} 16 & 17 \\ 7 & 21 \end{pmatrix} = \begin{pmatrix} 17 & 18 \\ 5 & 19 \end{pmatrix}, K_6^{-1} = \begin{pmatrix} 16 & 17 \\ 7 & 21 \end{pmatrix} \begin{pmatrix} 17 & 18 \\ 5 & 19 \end{pmatrix} = \begin{pmatrix} 19 & 13 \\ 16 & 5 \end{pmatrix}$$

К каждому из полученных выше вектор-столбцов блоков шифртекста применим операцию расшифрования:

$$X_1 = \begin{pmatrix} 1 & 23 \\ 25 & 4 \end{pmatrix} \begin{pmatrix} 25 \\ 17 \end{pmatrix} = \begin{pmatrix} 12 \\ 0 \end{pmatrix}, X_2 = \begin{pmatrix} 24 & 9 \\ 11 & 4 \end{pmatrix} \begin{pmatrix} 8 \\ 19 \end{pmatrix} = \begin{pmatrix} 19 \\ 7 \end{pmatrix}, X_3 = \begin{pmatrix} 17 & 23 \\ 20 & 7 \end{pmatrix} \begin{pmatrix} 7 \\ 12 \end{pmatrix} = \begin{pmatrix} 4 \\ 12 \end{pmatrix},$$

$$X_4 = \begin{pmatrix} 24 & 9 \\ 11 & 4 \end{pmatrix} \begin{pmatrix} 4 \\ 19 \end{pmatrix} = \begin{pmatrix} 0 \\ 19 \end{pmatrix}, X_5 = \begin{pmatrix} 17 & 18 \\ 5 & 19 \end{pmatrix} \begin{pmatrix} 14 \\ 6 \end{pmatrix} = \begin{pmatrix} 8 \\ 2 \end{pmatrix}, X_6 = \begin{pmatrix} 19 & 13 \\ 16 & 5 \end{pmatrix} \begin{pmatrix} 13 \\ 23 \end{pmatrix} = \begin{pmatrix} 18 \\ 0 \end{pmatrix}$$

Собрав полученные блоки открытого текста в единый текст и преобразовав числа к буквам, получим открытый текст ARITHMETICAL

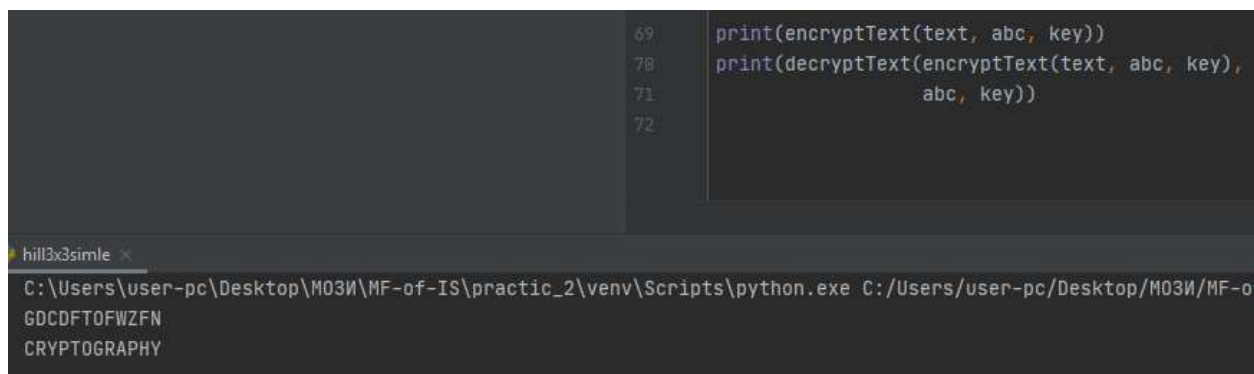
4. Программная реализация шифров

Результаты работы программы для вышеприведённых примеров шифрования:

1) Шифр Хилла

Открытый текст: CRYPTOGRAPHY

Ключ: $\begin{pmatrix} 1 & 2 & 2 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$



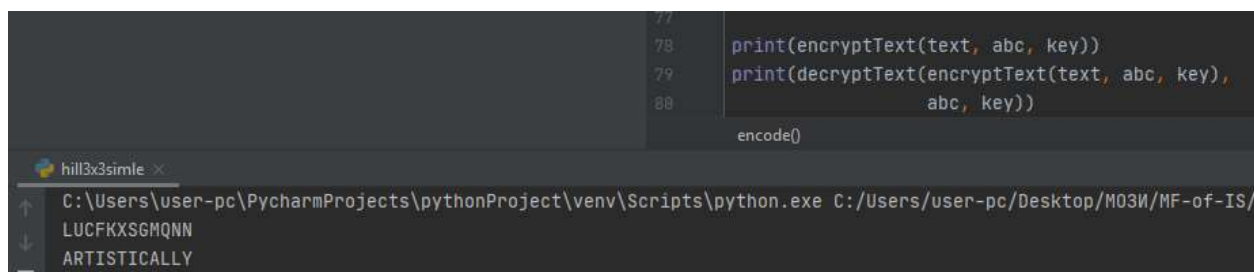
The screenshot shows a Python script in a code editor with the following lines: 69: print(encryptText(text, abc, key)) 70: print(decryptText(encryptText(text, abc, key), 71: abc, key)) 72: . Below the code, the terminal output for a program named 'hill3x3simle' is shown. It displays the encrypted text 'GDCDFTQFWZFN' and the original text 'CRYPTOGRAPHY'.

Рисунок 4.1 – Результаты работы Алгоритма, реализующего простой шифр Хилла

2) Шифр Хилла

Открытый текст: ARTISTICALLY

Ключ: $\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}$



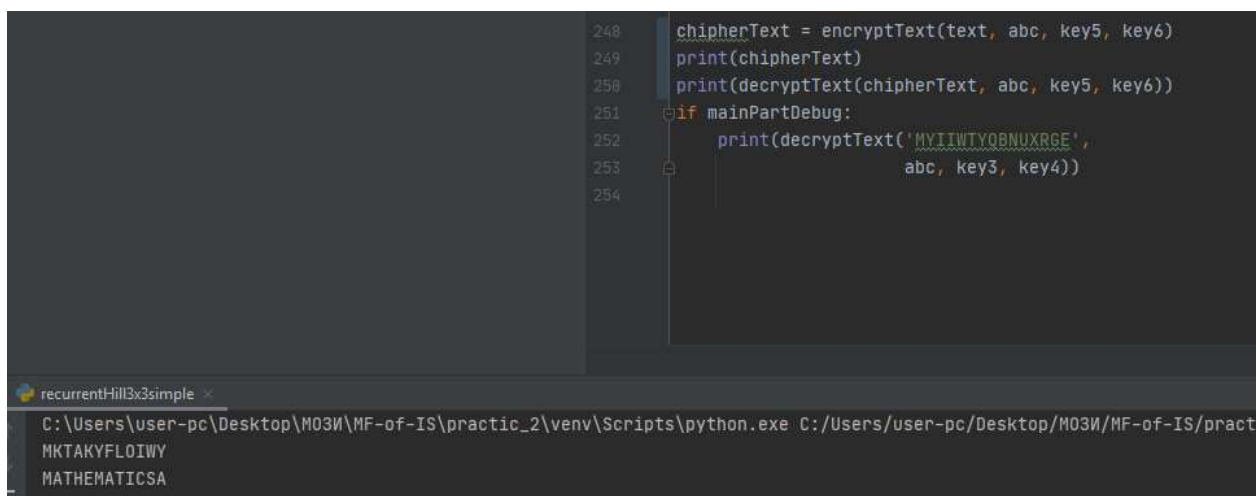
The screenshot shows a Python script in a code editor with the following lines: 77: 78: print(encryptText(text, abc, key)) 79: print(decryptText(encryptText(text, abc, key), 80: abc, key)) encode() . Below the code, the terminal output for a program named 'hill3x3simle' is shown. It displays the encrypted text 'LUCFKXSGMQNN' and the original text 'ARTISTICALLY'.

Рисунок 4.2 – Результаты работы Алгоритма, реализующего простой шифр Хилла

3) Рекуррентный шифр Хилла

Открытый текст: MATHEMATICS

Ключ: $\begin{pmatrix} 1 & 1 \\ 3 & 4 \end{pmatrix}, \begin{pmatrix} 4 & 3 \\ 1 & 1 \end{pmatrix}$



```
248 cipherText = encryptText(text, abc, key5, key6)
249 print(cipherText)
250 print(decryptText(cipherText, abc, key5, key6))
251 if mainPartDebug:
252     print(decryptText('MYIIWTYQBNUXRG',
253                       abc, key3, key4))
254
```

recurrentHill3x3simple

C:\Users\user-pc\Desktop\M03И\MF-of-IS\practic_2\venv\Scripts\python.exe C:/Users/user-pc/Desktop/M03И/MF-of-IS/practic_2/re

MKTAKYFLOIYW

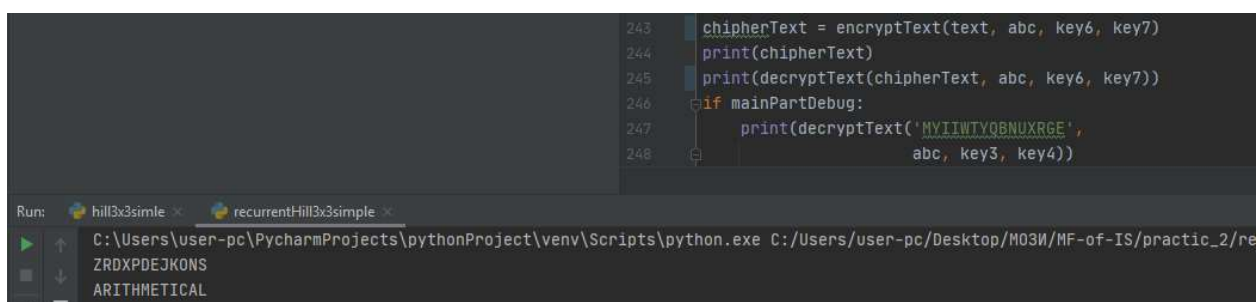
MATHEMATICS

Рисунок 4.3 – Алгоритм, реализующий рекуррентный шифр Хилла

4) Рекуррентный шифр Хилла

Открытый текст: ARITHMETICAL

Ключ: $\begin{pmatrix} 4 & 3 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 16 & 3 \\ 21 & 17 \end{pmatrix}$



```
243 cipherText = encryptText(text, abc, key6, key7)
244 print(cipherText)
245 print(decryptText(cipherText, abc, key6, key7))
246 if mainPartDebug:
247     print(decryptText('MYIIWTYQBNUXRG',
248                       abc, key3, key4))
```

Run: hill3x3simple recurrentHill3x3simple

C:\Users\user-pc\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/user-pc/Desktop/M03И/MF-of-IS/practic_2/re

ZRDXPDEJKONS

ARITHMETICAL

Рисунок 4.4 – Алгоритм, реализующий рекуррентный шифр Хилла

5. Примеры криптоанализа

Шифр Хилла

С другими примерами программ, реализующими взлом шифра Хилла можно ознакомиться здесь: <https://github.com/ChesleyTan/Hill-Cipher-Cracker>

Для эффективного криптоанализа шифра Хилла нужно знать n и определённую часть открытого текста, размер которой зависит от n . Рассмотрим случай, в котором нам известно, что $n = 2$ и где-то в открытом тексте встречается последовательность символов «OF THE». Пусть шифртекст выглядит следующим образом:

FUPCMTGZKYUKBQFJHUKTZKKIXTTA

STCLMNHIBIOIBNFRKWONTCYHFULCQSR

Поскольку при $n = 2$ шифр Хилла кодирует открытый текст блоками по 2 символа, должен быть верным один из следующих случаев:

ST	CL	MN	IB	IO	IB	NF	RK	WO	NT	CY	HF	UL	CQ	SR
OF	TH	E •	••	••	••	••	••	••	••	••	••	••	••	••
•O	FT	HE	••	••	••	••	••	••	••	••	••	••	••	••
••	OF	TH	E •	••	••	••	••	••	••	••	••	••	••	••
••	••	•O	FT	HE	••	••	••	••	••	••	••	••	••	••
••	••	••	OF	TH	E •	••	••	••	••	••	••	••	••	••
••	••	••	•O	FT	HE	••	••	••	••	••	••	••	••	••
••	••	••	••	OF	TH	E •	••	••	••	••	••	••	••	••
••	••	••	••	•O	FT	HE	••	••	••	••	••	••	••	••
••	••	••	••	••	OF	TH	E •	••	••	••	••	••	••	••
••	••	••	••	••	•O	FT	HE	••	••	••	••	••	••	••
••	••	••	••	••	••	OF	TH	E •	••	••	••	••	••	••
••	••	••	••	••	••	•O	FT	HE	••	••	••	••	••	••

и так далее.

Если верна вторая строка, то мы имеем $CL \rightarrow FT$, $MN \rightarrow HE$, что можно записать следующим образом:

$$D \begin{bmatrix} C \\ L \end{bmatrix} = \begin{bmatrix} F \\ T \end{bmatrix} \Rightarrow D \begin{bmatrix} 2 \\ 11 \end{bmatrix} = \begin{bmatrix} 5 \\ 19 \end{bmatrix} \pmod{26}$$

$$D \begin{bmatrix} M \\ H \end{bmatrix} = \begin{bmatrix} H \\ E \end{bmatrix} \Rightarrow D \begin{bmatrix} 12 \\ 7 \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \end{bmatrix} \pmod{26}$$

Цель – определить матрицу D, являющуюся ключом шифрования. Для этого сначала объединим два вышеприведённых равенства в одно:

$$D \begin{bmatrix} 2 & 12 \\ 11 & 7 \end{bmatrix} = \begin{bmatrix} 5 & 7 \\ 19 & 4 \end{bmatrix} \pmod{26}$$

$$D = \begin{bmatrix} 5 & 7 \\ 19 & 4 \end{bmatrix} \left(\begin{bmatrix} 2 & 12 \\ 11 & 7 \end{bmatrix} \right)^{-1} \pmod{26}$$

Если матрица, составленная из блоков шифртекста необратима, то наше предположение о положении участка OFTHE в открытом тексте заведомо неверное. В данном случае она не обратима.

Это означает, что изначальное предположение было неверным. Для нахождения открытого текста необходимо перебрать все возможные положения участка OFTHE и проделать вышеописанные вычисления для каждого положения. Для данного случая верное смещение участка по тексту равно 21: YH→FT, FU→HE. Проведя необходимые вычисления, получим следующую матрицу:

$$D = \begin{bmatrix} 16 & 3 \\ 21 & 18 \end{bmatrix}$$

Попытка расшифровать шифртекст с её помощью приведёт нас к осмысленному тексту:

FORONETOGOTOTNETEMPLEOFTHEKING

Рекуррентный шифр Хилла

Для рекуррентного шифра Хилла был найден только один метод криптоанализа – метод повторного шифрования. Этот метод требует наличия у злоумышленника шифртекста, ключа и шифратора, алгоритм действия которого ему неизвестен. В этом случае путём многократного шифрования на одном и том же ключе может быть получен открытый текст. Так для примера, приведённого в разделе ручного шифрования, необходимо провести 83 итерации шифрования, чтобы получить исходный открытый текст.

6. Выводы о проделанной работе

Шифр Хилла практически неуязвим для атак типа «только шифртекст», однако не имеет высокой стойкости по отношению к другим типам атак. Рекуррентный шифр Хилла исправляет некоторые недостатки шифра Хилла, но всё ещё имеет существенную уязвимость перед повторным шифрованием.

Методы криптоанализа, применимые к шифру Хилла и рекуррентному шифру Хилла опираются на то, что злоумышленник помимо шифртекста обладает некоторой информацией о ключе и об открытом тексте. Это является основным их недостатком. При наличии этой информации криптоанализ не составляет большого труда.

7. Список использованных источников

1. Традиционные шифры с симметричным ключом [электронный ресурс] – URL: <https://intuit.ru/studies/courses/552/408/lecture/9355?page=5>
2. Cryptanalysis of the Hill Cipher [электронный ресурс] – URL: <http://practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-hill-cipher/>
3. Хилл шифр – Hill cipher [электронный ресурс] – URL: https://ru.qaz.wiki/wiki/Hill_cipher

ПРИЛОЖЕНИЕ А.

Простой шифр Хилла

```
import numpy as np
from sympy import Matrix

def encode(text, alphabet):
    return [alphabet.index(c) for c in text]

def decode(data, alphabet):
    return "".join([alphabet[v] for v in data])

def get_inverse_key(key, M):
    return np.array(Matrix(*key.shape, key.reshape(-1)).inv_mod(M))

def encryptText(text, alphabet, key):
    M = len(alphabet)
    data = encode(text, alphabet)
    r = np.dot(np.array(data).reshape(-1, key.shape[0]), key.T) % M
    #print(r)
    pt = decode(r.flatten(), alphabet)
    return pt

def decryptText(text, alphabet, key):
    M = len(alphabet)
    data = encode(text, alphabet)
    keyinv = get_inverse_key(key, M)
    r = np.dot(np.array(data).reshape(-1, key.shape[0]), keyinv.T) % M
    pt = decode(r.flatten(), alphabet)
    return pt

# text = 'ГНГНЪДЛНВПЪМЕЩЬ'
text = 'ARTISTICALLY'
abc = 'АВВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ'
abc = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

key = np.array([
    [1, 2, 2],
    [4, 5, 6],
    [7, 8, 9]
])

key = np.array([
    [6, 24, 1],
    [13, 16, 10],
    [20, 17, 15]
])

print(encryptText(text, abc, key))
print(decryptText(encryptText(text, abc, key),
                  abc, key))
```

Рекуррентный шифр Хилла

```
import numpy as np
from sympy import Matrix

getInversDebug = False
encryptTextDebug = False
decryptTextDebug = False
createDecryptKeyDebug = False
mainPartDebug = False

def encode(text, alphabet):
    return [alphabet.index(c) for c in text]

def decode(data, alphabet):
    return "".join([alphabet[v] for v in data])

def get_inverse_key(key, M):
    if getInversDebug:
        print(np.array(Matrix(*key.shape, key.reshape(-1)).inv_mod(M)))

    return np.array(Matrix(*key.shape, key.reshape(-1)).inv_mod(M))

def encryptText(text, alphabet, key1, key2):
    keySize = key1.shape[1]
    text = checkText(text, keySize)
    M = len(alphabet)
    N = (len(text) // keySize) + 1
    if encryptTextDebug:
        print('M, N:', M, N)
    key = createKey(key1, key2, N, alphabet)
    data = encode(text, alphabet)
    if encryptTextDebug:
        print('data:', data)

    cnt = 0
    res = np.zeros(shape=(N - 1, keySize), dtype='int_')
    for i in np.array(data).reshape(-1, keySize):
        if encryptTextDebug:
            print(i)
            print('res[cnt]:', res[cnt])
            print(key[cnt].T)
        res[cnt] = np.dot(i, key[cnt].T) % M
        cnt += 1

    if encryptTextDebug:
        print('res', res)
    pt = decode(res.flatten(), alphabet)
    return pt

def decryptText(text, alphabet, key1, key2, ):
    keySize = key1.shape[1]
    text = checkText(text, keySize)
    M = len(alphabet)
    N = (len(text) // keySize) + 1
    if decryptTextDebug:
        print('N:', N)
    key = createDecryptKey(key1, key2, N, alphabet)
```



```

if decryptTextDebug:
    print('Key:', key)
data = encode(text, alphabet)

cnt = 0
res = np.zeros(shape=(N - 1, keySize), dtype='int_')
for i in np.array(data).reshape(-1, keySize):
    res[cnt] = np.dot(i, key[cnt].T) % M
    if decryptTextDebug:
        print(key[cnt])
    cnt += 1
if decryptTextDebug:
    print(data)
    print(res)
pt = decode(res.flatten(), alphabet)
return pt

def createKey(key1, key2, sizeStop, alphabet):
    M = len(alphabet)
    keyDim = key1.shape[0] if (key1.shape[0] == key1.shape[1] and
                               key2.shape[0] == key2.shape[1]) else 0
    if keyDim == 0:
        raise ValueError("Матрицы ключей должны быть квадратными")

    key = np.concatenate((key1, key2), axis=0)

    for _ in range(sizeStop - 2):
        mult = np.dot(key2, key1) % M
        key1 = np.copy(key2)
        key2 = np.copy(mult)
        key = np.concatenate((key, mult), axis=0)

    key.shape = (-1, keyDim, keyDim)
    return key

def createDecryptKey(key1, key2, sizeStop,
                     alphabet='АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ'):
    if createDecryptKeyDebug:
        print('SizeStop:', sizeStop)
    M = len(alphabet)
    keyDim = key1.shape[0] if (key1.shape[0] == key1.shape[1] and
                               key2.shape[0] == key2.shape[1]) else 0
    if keyDim == 0:
        raise ValueError("Матрицы ключей должны быть квадратными")

    key1inv = get_inverse_key(key1, M)
    key2inv = get_inverse_key(key2, M)
    keyInv = np.concatenate((key1inv, key2inv), axis=0)
    if createDecryptKeyDebug:
        print('KeyInv: ', keyInv)

    for _i in range(sizeStop - 2):
        if createDecryptKeyDebug:
            print('key1do:', _i, key1)
            print('key2do:', _i, key2)
            print('key1doInv:', _i, key1inv)
            print('key2doInv:', _i, key2inv)
            print(get_inverse_key(key1, M))
        multInv = np.dot(key1inv, key2inv) % M
        mult = np.dot(key2, key1) % M
        if createDecryptKeyDebug:

```

```

        print('multInv:', _i, multInv)
        print('mult:', _i, mult)
    key1inv = np.copy(key2inv)
    key2inv = np.copy(multInv)
    key1 = np.copy(key2)
    key2 = np.copy(mult)

    if createDecryptKeyDebug:
        print('key1:', _i, key1)
        print('key2:', _i, key2)

    keyInv = np.concatenate((keyInv, multInv), axis=0)

    keyInv.shape = (-1, keyDim, keyDim)
    if createDecryptKeyDebug:
        print(keyInv)
    return keyInv

def checkText(text, keySize):
    if keySize == 3:
        if len(text) % 3 == 0:
            return text
        elif len(text) % 3 == 1:
            return text + 'AA'
        else:
            return text + 'A'
    elif keySize == 2:
        if len(text) % 2 == 0:
            return text
        else:
            return text + 'A'
    else:
        raise KeyError('Ключ задан неверно. Предусмотрен ключ только 2x2 или 3x3.')

# text = 'ГНГНЪДЛНВПЪМЕЩЬ'
# text = 'ГНГНЪДЖХКЕТЖРЮТ'
text = 'CRYPTOGRAPHY'
text = 'MATHEMATICS'
text = 'ARITHMETICAL'
abc = 'АВВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ'
abc = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

key1 = np.array([
    [1, 2, 0],
    [0, 1, 4],
    [1, 2, 2]
])

key2 = np.array([
    [1, 2, 0],
    [0, 1, 4],
    [1, 2, 2]
])

key3 = np.array([
    [1, 2, 2],
    [4, 5, 6],
    [7, 8, 9]
])

```

```

key4 = np.array([
    [9, 8, 7],
    [6, 5, 4],
    [2, 2, 1]
])

key5 = np.array([
    [1, 1],
    [3, 4]
])

key6 = np.array([
    [4, 3],
    [1, 1]
])

key7 = np.array([
    [16, 3],
    [21, 17]
])

key8 = np.array([
    [11, 15, 12],
    [15, 2, 15],
    [17, 15, 19]
])

keyTest = np.array([
    [12, 10, 25],
    [2, 17, 0],
    [17, 22, 25]
])

keylinv = np.array([
    [25, 2, 25],
    [18, 7, 2],
    [18, 8, 25]
])

key2inv = np.array([
    [1, 16, 25],
    [6, 15, 20],
    [19, 4, 6]
])

if mainPartDebug:
    print(np.dot(keylinv, key2inv) % 26)
    print(get_inverse_key(key6, 26))
    print(get_inverse_key(keyTest, 26))

cipherText = encryptText(text, abc, key6, key7)
print(cipherText)
print(decryptText(cipherText, abc, key6, key7))
if mainPartDebug:
    print(decryptText('MYIIWTYQBNUXRGE',
                      abc, key3, key4))

```