Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Изучение устройства и функциональных особенностей шифровальной машины «Энигма»

Студент: Чёрная Я.Р.

ФИТ 3 курс 4 группа

Преподаватель: Нистюк О.А.

Минск 2025

**Теоретические сведения**

Первая шифровальная машина, «Enigma A», появилась в 1923 году и весила около 50 кг. Вскоре была представлена «Enigma B», похожая на первую модель, но обе оказались слишком громоздкими для военного использования. Во время Второй мировой войны было произведено более 100 тысяч таких машин.

«Энигма» – это электромеханическое устройство, состоящее из механической и электрической частей. Механическая часть включает клавиатуру, роторы и ступенчатый механизм, который поворачивает роторы при каждом нажатии клавиши. Электрическая часть включает клавиатуру, коммутационную панель, лампочки и роторы.

Основные блоки «Энигмы»:

* Панель механических клавиш, дающая сигнал для поворота роторов.
* Три или более роторных дисков, каждый с 26 контактами, коммутируемыми в случайном порядке.
* Рефлектор, соединенный с крайним левым ротором.
* Коммутационная панель для изменения электрических соединений двух букв.
* Панель с электрическими лампочками, показывающими выходную букву.

При каждом нажатии клавиши самый правый ротор сдвигается на одну позицию, а при определенных условиях сдвигаются и другие роторы. Это приводит к различным криптографическим преобразованиям, заменяя один символ другим. Механические части замыкают контакты, создавая электрический контур. При нажатии клавиши контур замыкается, ток проходит через цепь и включает лампочку, показывая букву шифртекста. Рефлектор замыкает цепь, возвращая ток по другому пути и зажигая лампочку.

*EK = f (m, В, Re, L, M, R)*.

Чтобы оценить криптостойкость шифра, нужно учитывать все возможные настройки машины. Для этого необходимо рассмотреть следующие свойства «Энигмы»:

* выбор и порядок роторов;
* разводку (коммутацию) роторов;
* настройку колец на каждом из роторов;
* начальное положение роторов в начале сообщения;
* отражатель;
* настройки коммутационной панели.

# Зашифрование с помощью машины «Энигма»

Для реализации зашифрования знать и хранить следующие данные:

* подстановочные алфавиты, соответствующие каждому ротору;
* текущую позицию каждого ротора;
* общее количество смещений каждого ротора;
* количество полных оборотов роторов;
* шаги смещения каждого ротора.

|  |
| --- |
| Rotor rotorL = new Rotor("EKMFLGDQVZNTOWYHXUSPAIBRCJ", 0, 16); // L rotor (шаг 0, перекидывает при Q)  Rotor rotorM = new Rotor("LEYJVCNIXWPBQMDRTAKZGFUHOS", 2, -1); // Beta rotor (без перекидывания)  Rotor rotorR = new Rotor("FKQHTLXOCBJSPDZRAMEWNIUYGV", 2, 7); // R rotor (шаг 2, перекидывает при V)  Reflector reflectorC = new Reflector("FVPJIAOYEDRZXWGCTKUQSBNMHL"); // Reflector C  // Вращение роторов  bool rotateR = true;  bool rotateM = rotors[2].ShouldRotateNext(); // Проверка правого ротора  bool rotateL = rotors[1].ShouldRotateNext(); // Проверка среднего ротора  if (rotateM) rotors[1].Rotate(); // Двойной шаг среднего ротора  if (rotateL) rotors[0].Rotate();  if (rotateR) rotors[2].Rotate(); |

step — как часто ротор поворачивается (из Li-Mi-Ri).

turnover — позиция, при которой ротор заставляет следующий вращаться

Роторы согласно варианту: L=2, M=Beta, R=8, Reflector=C

**Класс Rotor (Ротор).**Моделирует один ротор машины Enigma, отвечая за:

* Шифрование символов при прохождении сигнала
* Вращение согласно заданным правилам
* Активацию следующего ротора при достижении позиции переключения
* EncryptForward() - преобразует символ при движении справа налево
* EncryptBackward() - преобразует символ при движении слева направо
* Rotate() - поворачивает ротор на одну позицию
* ShouldRotateNext() - проверяет, нужно ли активировать следующий ротор

class Rotor

{

private string wiring;

private int position;

private int step;

private int turnover;

public Rotor(string wiring, int step, int turnover)

{

this.wiring = wiring;

this.position = 0;

this.step = step;

this.turnover = turnover;

}

public void SetPosition(int pos)

{

position = pos % 26;

}

public char EncryptForward(char input)

{

int index = (input - 'A' + position) % 26;

char output = wiring[index];

return (char)((output - 'A' - position + 26) % 26 + 'A');

}

public char EncryptBackward(char input)

{

int index = (input - 'A' + position) % 26;

int wiredIndex = wiring.IndexOf((char)('A' + index));

return (char)((wiredIndex - position + 26) % 26 + 'A');

}

public void Rotate()

{

position = (position + 1) % 26;

}

public bool ShouldRotateNext()

{

return position == turnover;

}

}

**Класс Reflector (Отражатель).** Обеспечивает рефлексивное преобразование символа, замыкая электрическую цепьОсобенности:

* Не вращается и не изменяет положение
* Обеспечивает взаимно-однозначное преобразование символов
* Критически важен для симметричности шифрования (возможность расшифровки)

class Reflector

{

private string wiring;

public Reflector(string wiring)

{

this.wiring = wiring;

}

public char Reflect(char input)

{

return wiring[input - 'A'];

}

}

**Класс EnigmaMachine.** Координирует работу всех компонентов, реализуя:

* Инициализацию системы роторов
* Установку начальных позиций
* Полный цикл шифрования символа
* Управление вращением роторов

class EnigmaMachine

{

private Rotor[] rotors;

private Reflector reflector;

public EnigmaMachine(Rotor[] rotors, Reflector reflector)

{

this.rotors = rotors;

this.reflector = reflector;

}

public void SetPositions(int[] positions)

{

for (int i = 0; i < rotors.Length; i++)

{

rotors[i].SetPosition(positions[i]);

}

}

public string Encrypt(string input)

{

string output = "";

foreach (char c in input)

{

if (c < 'A' || c > 'Z') continue;

// Вращение роторов

bool rotateR = true;

bool rotateM = rotors[2].ShouldRotateNext(); // Проверка правого ротора

bool rotateL = rotors[1].ShouldRotateNext(); // Проверка среднего ротора

if (rotateM) rotors[1].Rotate(); // Двойной шаг среднего ротора

if (rotateL) rotors[0].Rotate();

if (rotateR) rotors[2].Rotate();

// Прохождение сигнала справа налево

char encryptedChar = c;

for (int i = rotors.Length - 1; i >= 0; i--)

{

encryptedChar = rotors[i].EncryptForward(encryptedChar);

}

// Отражение

encryptedChar = reflector.Reflect(encryptedChar);

// Прохождение сигнала слева направо

for (int i = 0; i < rotors.Length; i++)

{

encryptedChar = rotors[i].EncryptBackward(encryptedChar);

}

output += encryptedChar;

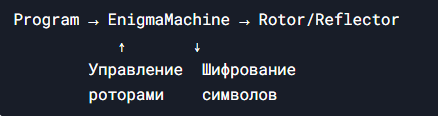
}

return output;

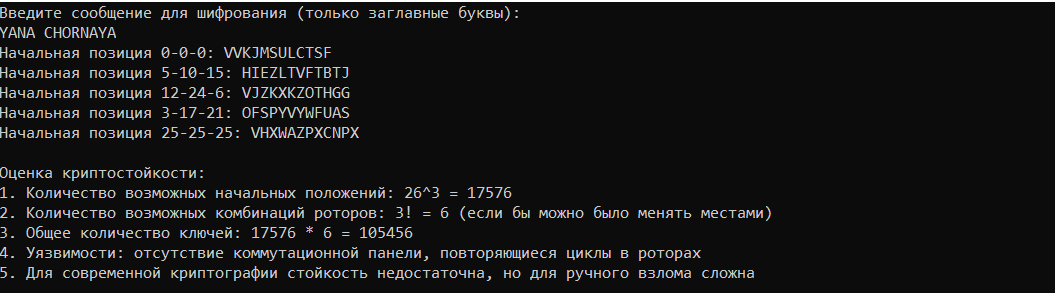
}

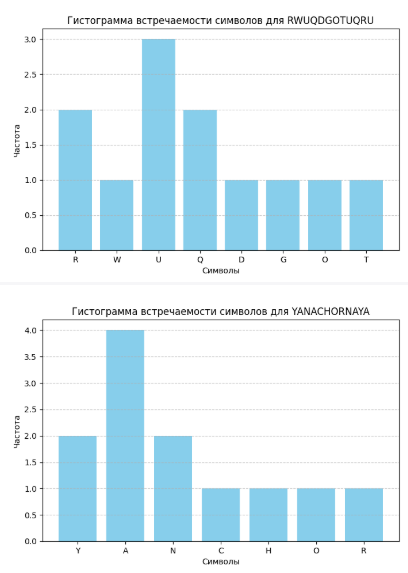
}

Взаимодействие классов



* Rotor знает только о своем состоянии и преобразованиях
* Reflector выполняет только фиксированное отражение
* EnigmaMachine координирует процесс, не зная деталей реализации компонентов
* Program работает только с интерфейсом EnigmaMachine





# Вывод

Машина «Энигма» получила широкое распространение во время Второй Мировой войны за счёт своей высокой криптостойкости. Она достигается за счёт того, что каждый символ шифруется 7 раз (в варианте с тремя роторами без коммутационной панели), причём при шифровании каждого символа получается новый символ, что также повышает криптостойкость.

В сравнении моего варианта реализации с реальной машиной «Энигма», мой вариант является немного менее криптостойким за счёт отсутствия коммутационной панели, которая добавляет от 0 до 2 замен каждого символа. Однако, шифровальная машина «Энигма» все так же является в разы более криптостойкой, чем рассмотренные ранее шифры.