Министерство образования Республики Беларусь

Учреждение образование

«Белорусский государственный технологический университет»

Кафедра информационных систем и технологий

**Отчет к лабораторной работе**:

«Исследование криптографических шифров на основе перестановки символов»

Выполнил:

студент 3 курса 8 группы ФИТ

Романович Никита Русланович

Проверил:

Берников В. О.

Минск 2020

**1 Теоретическая часть**

Идея создания шифровального устройства высказана голландцем Гуго Кох де Дельфтю (в некоторых источниках – Гуго Александр Кох, Hugo Alexander Koch) еще в 1919 г. В 1920 году он же изобрел первую роторную шифровальную машинку. Параллельно с этим немец Артур Шербиус (Arthur Scherbius) изучал проблему криптостойкости (в нашем современном понимании) шифровальных машин. Он же получил патент на такую машину, которая получила название «Enigma» (от греч. – загадка). Основная особенность Энигмы – все знали в то время алгоритм шифрования, но никто не мог подобрать нужный ключ.

Первая шифровальная машина, Enigma A, появилась на рынке в 1923 году. Это была большая и тяжелая машина со встроенной пишущей машинкой и весом около 50 кг. Вскоре после этого была представлена Enigma B, очень похожая на Enigma A. Вес и размеры этих машин сделали их непривлекательными для использования в военных целях.

По достоинству шифровальную машину оценили в немецкой армии. В 1925 году её принял на вооружение сначала военно-морской флот (модель Funkschlussen С), а в 1930-м – и Вермахт (Enigma I). Общее количество шифраторов, произведённых до и во время Второй мировой войны, превысило 100 тысяч. Применялись они всеми видами вооруженных сил Германии, а также военной разведкой и службой безопасности.

Идея коллеги А. Шербиуса, Вилли Корна (Willi Korn), позволила создать компактную и намного более легкую Enigma C. Особенностью этой модели было наличие ламповой панели. В 1927 году Enigma D была представлена и коммерциализирована в нескольких версиях с различными роторами и продана военным и дипломатическим службам многих стран Европы.

В Enigma D было три обычных ротора и один отражатель (рефлектор), которые можно было установить в одном из 26 положений (по числу букв используемого алфавита). Именно эта модель стала основным прототипом многих известных версий машин Энигма, которые использовала Германия в годы Второй Мировой войны.

**1.2 Шифры Энигмы**

Во время Второй мировой войны немецкие операторы использовали специальную (тайную) шифровальную книгу для установки роторов и настроек колец. Операторы Энигмы (шифровальщики и дешифровальщики выполняли следующие основные операции.

Пример 1.

Зашифрование сообщения.

1. Установить начальную стартовую позицию роторов (предположим, их 3), согласно текущей кодовой таблице (коду дня). Например, WZA.
2. Выбрать случайный ключ сообщения, например, SXT. Затем оператор устанавливал роторы в стартовую позицию WZA.
3. Зашифровывать ключ сообщения SXT. Предположим, что в результате зашифрования ключа получится UHL.
4. Далее оператор ставил ключ сообщения (SXT) как начальную позицию роторов и зашифровывал собственно сообщение. После этого он отправлял стартовую позицию (WZA) и зашифрованный ключ (UHL) вместе с сообщением.

Пример 2.

Зашифрование сообщения.

1. Установить стартовую позицию роторов согласно коду дня. Например, если код был "HUA", роторы должны быть инициализированы на "H", "U" и "A" соответственно.
2. Выбрать случайный код с тремя буквами, например, ACF.
3. Зашифровать текст "ACFACF" (повторный код), используя начальную установку роторов шага 1. Например, предположим, что зашифрованный код – "OPNABT".
4. Установить стартовые позиции роторов к OPN (половина зашифрованного кода). 5. Присоединить зашифрованные шесть букв, полученных на шаге 2 (OPNABT), в конец к начальному сообщению.
5. Зашифровать сообщение, включая код с 6 -ю буквами. Передать зашифрованное сообщение.

Военная модель Энигмы использовала только 26 букв. Прочие символы заменялись редкими комбинациями букв. Пробел пропускался либо заменялся на «X». Символ «X» также использовался для обозначения точки либо конца сообщения. Некоторые особые символы использовались в отдельных вооруженных частях, например, Wehrmacht заменял запятую двумя символами ZZ и вопросительный знак – словом FRAGE либо буквосочетанием FRAQ, a Kriegsmarine М4 запятой соответствовала буква «У». Как мы отмечали выше, Энигма строится на основе подстановочных шифров, подобных на шифр Цезаря, в котором, как известно, ключ сообщения, который должен знать получатель, – это просто смещение между двумя алфавитами.

Принято считать, что в основе шифра Энигмы лежит динамический шифр Цезаря. Более сложная система использует случайный ряд символов для нижнего алфавита. Принцип, положенный в основу этой «случайности», имеет много общего с перестановочными шифрами. Например, ниже показан принцип подстановке, основанный на взаимной перестановке во втором (нижнем) алфавите в 13 парах символов, расположенных случайным образом:

А B C D Е F G Н I J К L М N O Р Q R S T U V W Х Y Z

J W M R Z L N T U A O F C G K S Y D P H I X B V Q E

Этот принцип случайности использовался и при изготовлении роторов и рефлекторов для Энигмы. Всего за время Второй мировой войны немцами было изготовлено восемь роторов и четыре рефлектора, но одновременно могло использоваться ровно столько, на сколько была рассчитана машина. Техническую спецификацию на все произведенные роторы и рефлекторы можно найти в [12, 21]. Ниже на рис. 4.5 и 4.6 представлены эти спецификации соответственно на роторы и на рефлекторы.

**1.3 Оценка криптостойкости Энигмы**

Для получения общего представления об особенностях работы криптоаналитиков над шифрами Энигмы полезно ознакомиться с содержанием материалов.

Как мы неоднократно подчеркивали, преобразование «Энигмы» для каждой буквы может быть определено математически как результат подстановок. Рассмотрим трехроторную модель Энигмы. Положим, что символом В обозначаются операции с использованием коммутационной панели, соответственно символы Re – отражателя, а L, M и R – обозначают действия левых, средних и правых роторов соответственно. Тогда процесс зашифрования символа m c использованием некоторой ключевой информации К формально можно записать в следующем виде: EК = 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.

Используются различные варианты подсчета всех возможных состояний перечисленных конструктивных модулей машины [27]. К сожалению для немцев, взломщики шифра союзников знали машину, роторы и внутреннюю разводку этих роторов. Поэтому им нужно было учитывать только возможные способы настройки Энигмы. Такая априорная информация о конструктивных особенностях устройства для шифрования (вспомним об основных постулатах О. Керкгоффса [2]) в нашем случае снижает уровень (теоретический) криптостойкости (до практического). Немецкие криптологи полагали, что один ротор может быть подключен 4 х 1026 различными способами. Сочетание трех роторов и отражателя позволяет получить астрономические цифры возможных вариантов подстановок. Для союзников, которые знали конструкции роторов, число различных вариантов существенно уменьшалось.

Рассмотрим пример для трехроторной Энигмы Вермахта с отражателем (по умолчанию – B, см. рис. 4.6) и выбором из 5 роторов. Использовались 10 штекерных кабелей на коммутационной панели (количество кабелей по умолчанию, поставляемых с машиной).

Чтобы выбрать 3 ротора из возможных 5, существует 60 комбинаций (5 х 4 х 3). Каждый ротор (его внутренняя проводка) может быть установлен в любом из 26 положений. Следовательно, с 3 роторами имеется 17 576 различных положений ротора (26 x 26 x 26). Кольцо на каждом роторе содержит маркировку ротора (что здесь неважно) и выемку, которая влияет на шаг перемещения расположенного левее ротора. Каждое кольцо может быть установлено в любом из 26 положений. Поскольку слева от третьего (наиболее левого) ротора нет ротора, на расчет влияют только кольца самого правого и среднего ротора. Это дает 676 комбинаций колец (26 х 26).

Коммутационная панель обеспечивает самый большой набор возможных настроек. Для первого кабеля одна сторона может иметь любое из 26 положений, а другая сторона - любое из 25 оставшихся положений (одна буква коммутируются с другой). Однако, поскольку комбинация и ее обратная сторона идентичны (AB такая же, как BA), мы должны игнорировать все двойные числа во всех возможных комбинациях для одного кабеля, предоставляя (26 x 25) / (1! х 21) или 325 уникальных способов коммутаций одним кабелем. Для двух кабелей: есть (26 x 25) комбинаций – для первого кабеля и, поскольку два разъема уже используются, то получается (24 x 23) комбинаций – для второго кабеля. Следуя этой простой логике, получается (26 x 25 x 24 x 23) / (2! X 22) = 44 850 уникальных способов коммутаций с использованием двух кабелей. Для трех кабелей – (26 x 25 x 24 x 23 x 22 x 21) / (3! х 23) = 3 453 450 комбинаций и так далее. Таким образом, с использованием 10 кабелей на коммутационной панели получаются 150 738 274 937 250 различных комбинаций. Формула, где n равно количеству кабелей, равна 26! / (26 - 2n)! · n! · 2n. Численно это дает:60 x 17 576 x 676 x 150 738 274 937 250 = 107 458 687 327 250 619 360 000 или 1,07 x 1023.

Таким образом, практически рассматриваемая версия Энигмы (три ротора с выбором из 5 роторов, отражатель В и 10 штекерных кабелей для коммутационной панели) может быть настроена на 1,07 x 1023 различных состояний, что сопоставимо с 77-битным криптографическим ключом.

Добавление четвертого ротора (например, для Naval Enigma M4) для повышения его криптостойкости было практически бесполезным: неподвижный четвертый ротор «усложнил машину» только в 26 раз и вместе с тонким отражателем мог рассматриваться как настраиваемый отражатель с 26 положениями. Внедрение общего числа роторов в 8 единиц (на Kriegsmarine M3), а затем – на четырехроторной версии (M4) было гораздо более эффективным шагом. Они увеличили комбинации роторов с 60 до 336.

Проблема криптоанализа шифров Энигмы была экстраординарной (с учетом электромеханических конструкций устройств для криптоанализа, применяемых в то время). Исчерпывающий поиск всех возможных 1,07 x 1023 настроек (атака brute force) был невозможен в 1940-х годах, а его сопоставимый 77-битный ключ огромен даже для современных электронных систем. Чтобы дать представление о размере этого числа, представим, что у нас есть 1,07 x 1023 листов бумаги толщиной около 1 мм. Из этих листов можно сложить примерно 70 000 000 стопок бумаги, каждая из которых простирается от Земли до Солнца. Кроме того, 1,07 x 1023 дюйма равно 288 500 световых лет.

**2 Практическая часть**

В данной лабораторной работе необходимо было разработать приложение-симулятор шифровальной машины, состоящей из клавиатуры, трех роторов и отражателя.

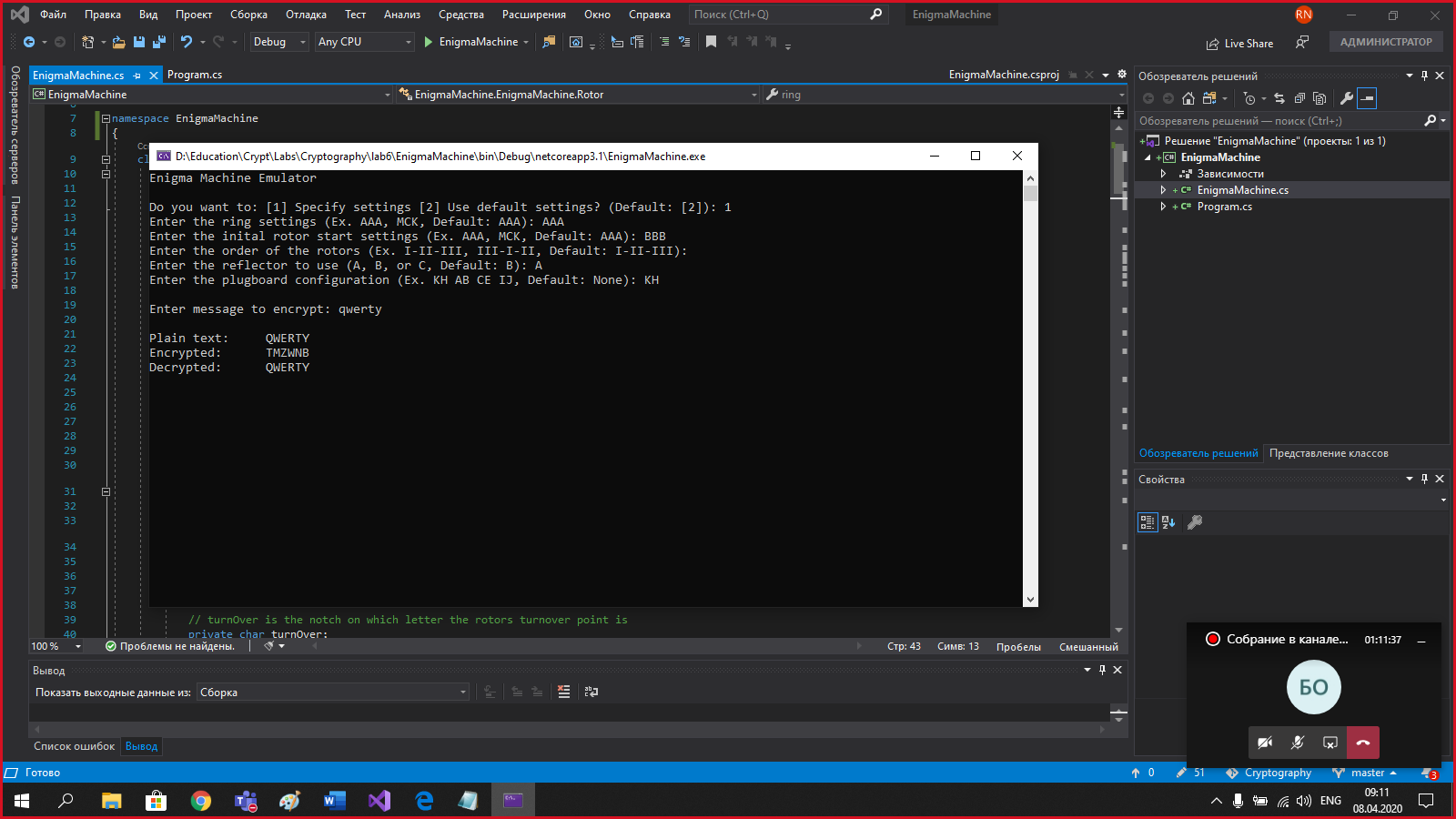


Рисунок 1 *–* Общий интерфейс

С помощью разработанного приложения зашифровать сообщение в соответствии с п.1 практического задания, применив не менее 5 вариантов начальных установок роторов.

Приложение

class EnigmaMachine {

/\* Enigma Machine

Modelled after Enigma I, from ~1930

\*/

private Dictionary<Char, Char> plugBoard;

// The machine has three rotors and a reflector

private Rotor[] rotors;

private Rotor reflector;

private const string alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

// Rotor and reflectors. These configurations are constant and the same on every Enigma machine

private const string rotorIconf = "EKMFLGDQVZNTOWYHXUSPAIBRCJ";

private const string rotorIIconf = "AJDKSIRUXBLHWTMCQGZNPYFVOE";

private const string rotorIIIconf = "BDFHJLCPRTXVZNYEIWGAKMUSQO";

private const string reflectorAconf = "EJMZALYXVBWFCRQUONTSPIKHGD";

private const string reflectorBconf = "YRUHQSLDPXNGOKMIEBFZCWVJAT";

private const string reflectorCconf = "FVPJIAOYEDRZXWGCTKUQSBNMHL";

// Rotor class representing one rotor

private class Rotor {

// The current char of the alphabet, and position of it. This char is visible outside the machine

private int outerPosition;

public char outerChar { get; set; }

// The fixed alphabet of the rotor

private string wiring;

// turnOver is the notch on which letter the rotors turnover point is

private char turnOver;

public string name { get; }

// Ring is the wiring setting relative to the turnover notch and position

// Basically part of the initialization vector

public char ring { get; set; }

public int[] map { get; }

public int[] revMap { get; }

public Rotor(string w, char to, string n) {

turnOver = to;

outerPosition = 0;

ring = 'A'; // A default ring setting

name = n;

map = new int[26];

revMap = new int[26];

setWiring(w);

}

public void setWiring(string newW) {

wiring = newW;

outerChar = wiring.ToCharArray()[outerPosition];

// Fill the mapping arrays

for (int i = 0; i < 26; i++) {

int match = ((int)wiring.ToCharArray()[i]) - 65;

map[i] = (26 + match - i) % 26;

revMap[match] = (26 + i - match) % 26;

}

}

public void setOuterPosition(int i) {

outerPosition = i;

outerChar = alphabet.ToCharArray()[outerPosition];

}

public int getOuterPosition() {

return outerPosition;

}

public void setOuterChar(char c) {

outerChar = c;

outerPosition = alphabet.IndexOf(outerChar);

}

public void step() {

outerPosition = (outerPosition + 1) % 26;

outerChar = alphabet.ToCharArray()[outerPosition];

}

public bool isInTurnOver() {

return outerChar == turnOver;

}

}

private void rotateRotors(Rotor[] r) {

if (r.Length == 3) {

if (r[1].isInTurnOver()) {

// If rotor II is on turnOver, all rotors step

r[0].step();

r[1].step();

} else if (r[2].isInTurnOver()) {

// If rotor III is on turnOver, the two rotors to the right step

r[1].step();

}

// Rotor III always steps

r[2].step();

}

}

// Apply the rotor scramble to character using all three rotors

// Argumentent reverse decides which direction we are scrambling

private char rotorMap(char c, bool reverse) {

int cPos = (int)c - 65;

if (!reverse) {

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

cPos = rotorValue(rotors[i], cPos, reverse);

}

} else {

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

cPos = rotorValue(rotors[i], cPos, reverse);

}

}

return alphabet.ToCharArray()[cPos];

}

private int rotorValue(Rotor r, int cPos, bool reverse) {

int rPos = (int)r.ring - 65;

int d;

if (!reverse)

d = r.map[(26 + cPos + r.getOuterPosition() - rPos) % 26];

else

d = r.revMap[(26 + cPos + r.getOuterPosition() - rPos) % 26];

return (cPos + d) % 26;

}

// Apply the reflector, the part that comes after the rotors

private char reflectorMap(char c) {

int cPos = (int)c - 65;

cPos = (cPos + reflector.map[cPos]) % 26;

return alphabet.ToCharArray()[cPos];

}

// Constructor

public EnigmaMachine() {

plugBoard = new Dictionary<char, char>();

// Notch and alphabet are fixed on the rotor

// First argument is alphabet, second is the turnover notch

Rotor rI = new Rotor(rotorIconf, 'Q', "I");

Rotor rII = new Rotor(rotorIIconf, 'E', "II");

Rotor rIII = new Rotor(rotorIIIconf, 'V', "III");

rotors = new Rotor[] { rI, rII, rIII }; // Default ordering of rotors

reflector = new Rotor(reflectorAconf, ' ', "");

}

public void setReflector(char conf) {

if (conf != 'A' && conf != 'B' && conf != 'C') {

throw new ArgumentException("Invalid argument");

}

string wiring = "";

switch (conf) {

case 'A':

wiring = reflectorAconf;

break;

case 'B':

wiring = reflectorBconf;

break;

case 'C':

wiring = reflectorCconf;

break;

}

reflector.setWiring(wiring);

}

// Enter the ring settings and initial rotor positions

public void setSettings(char[] rings, char[] grund) {

if (rings.Length != rotors.Length || grund.Length != rotors.Length) {

throw new ArgumentException("Invalid argument lengths");

}

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

rotors[i].ring = Char.ToUpper(rings[i]);

rotors[i].setOuterChar(Char.ToUpper(grund[i]));

}

}

public void setSettings(char[] rings, char[] grund, string rotorOrder) {

Rotor rI = null;

Rotor rII = null;

Rotor rIII = null;

// Get the current ordering

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

if (rotors[i].name == "I")

rI = rotors[i];

if (rotors[i].name == "II")

rII = rotors[i];

if (rotors[i].name == "III")

rIII = rotors[i];

}

string[] order = rotorOrder.Split('-');

// Set the new ordering

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

if (order[i] == "I")

rotors[i] = rI;

if (order[i] == "II")

rotors[i] = rII;

if (order[i] == "III")

rotors[i] = rIII;

}

setSettings(rings, grund);

}

public void setSettings(char[] rings, char[] grund, string rotorOrder, char reflectorConf) {

setReflector(reflectorConf);

setSettings(rings, grund, rotorOrder);

}

// Encrypts or decrypts a message

public string runEnigma(string msg) {

StringBuilder encryptedMessage = new StringBuilder();

msg = msg.ToUpper();

foreach (char c in msg) {

encryptedMessage.Append(encryptChar(c));

}

return encryptedMessage.ToString();

}

// Encrypts (or decrypts) a single character

private char encryptChar(char c) {

// Rotate the rotors before scrambling

rotateRotors(rotors);

// Into plugboard from keyboard <--

if (plugBoard.ContainsKey(c)) {

c = plugBoard[c];

}

// Scramble with rotors

// First we go all the way through the rotors <--

c = rotorMap(c, false);

// Reflect at the end so we don't just unscramble it again when we go back

// If the line below is commented out, the cipher will be equal to the message

c = reflectorMap(c);

// Go back through all the rotors the other way -->

c = rotorMap(c, true);

// Plugboard again, from other direction -->

if (plugBoard.ContainsKey(c)) {

c = plugBoard[c];

}

// Character is now encrypted

return c;

}

// Add a character pair into the plugboard

public void addPlug(char c, char cc) {

if (Char.IsLetter(c) && Char.IsLetter(cc)) {

c = Char.ToUpper(c);

cc = Char.ToUpper(cc);

if (c != cc && !plugBoard.ContainsKey(c)) {

plugBoard.Add(c, cc);

plugBoard.Add(cc, c);

}

} else {

throw new ArgumentException("Invalid character");

}

}

}

Криптостойкость моего варианта машины отображена на рисунке 2.

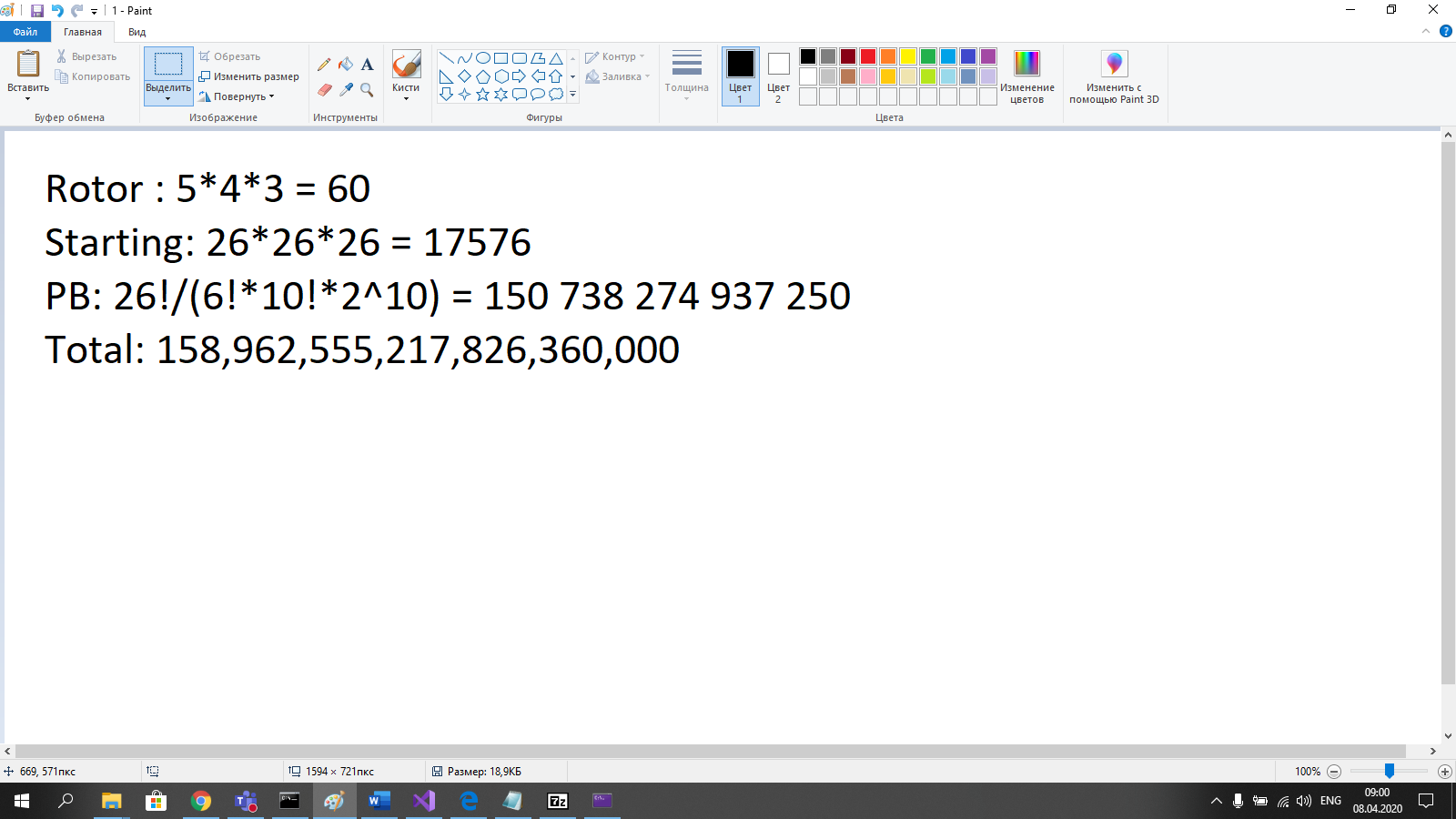


Рисунок 2 *–* Криптостойкость машины

Вывод: В данной лабораторной работе я изучил и приобрел практические навыки разработки и использования приложений для реализации перестановочных шифров. Разработал приложение-симулятор шифровальной машины, состоящей из клавиатуры, трех роторов и отражателя. С помощью разработанного приложения зашифровал сообщение в соответствии с п.1 практического задания, применив не менее 5 вариантов начальных установок роторов.