Учреждение образования

«Белорусский государственный технологический университет»

**Отчет по практическим работам**

По дисциплине: «Защита информации и надежность информационных систем»

Студента 3 курса 1 группы ФИТ

Халалеенко Андрей Николаевич

**Элементы теории информации. Информативность данных в различных кодировках**

**Цель:** приобретение практических навыков трансформации данных и сопоставление энтропийных свойств используемых при этом алфавитов.

**Задачи:**

1. Закрепить теоретические знания по взаимной конвертации данных, представленных в кодах ASCII и base64.

2. Разработать приложение для конвертации произвольного документа в формат base64 и обратно.

3. Исследовать энтропийные характеристики используемых в конвертерах алфавитов.

4. Изучить особенности практической реализации операции XOR над данными, представленными в разных форматах.

5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**Теоретические сведения**

Из энтропийных оценок (алфавитов и сообщений), полученных в ходе выполнения лабораторной работы № 2, мы выяснили, что энтропия зависит от статических характеристик самих алфавитов и сообщений (вспомним энтропию по Шеннону и по Хартли).

Энтропия максимальна при равномерном появлении букв на любом месте сообщения. Для характеристики источника сообщений с различным алфавитом представляет интерес сравнение фактической энтропии источника с максимально возможной. В этом смысле введено понятие избыточности источника сообщений, или избыточности алфавита.

Избыточностью алфавита называют уменьшение информационной нагрузки на один символ вследствие разной вероятности и взаимозависимости появления его символов в сообщениях.

В наиболее общем виде избыточность алфавита R можно оценить отношением энтропии по Хартли и по Шеннону; при этом первая рассчитывается по выражению, вторая – по формуле.

При выполнении предыдущей работы мы убедились, что формально одно и то же сообщение, но представленное на основе алфавита русского (белорусского, английского или иного) языка – с одной стороны, и представленное в кодах ASCII – с другой, будут характеризоваться различным количеством содержащейся в них информации. Эта дополнительная избыточность обусловлена переносом сообщения из одной среды в другую или, иначе говоря, кодированием символов исходного алфавита.

Утверждение восьмибитных кодировок (ASCII) как стандарта принесло некоторые проблемы. К этому моменту уже существовала определенная инфраструктура, использующая семибитные кодировки. Известны проблемы с «обрезанием восьмого бита» в системе электронной почты. Утверждение восьмибитного символа дало 256 различных значений, что позволило уместить в одной кодовой таблице и общепринятые символы (цифры, знаки препинания, латиницу), и символы кириллицы.

Уже созданное к тому времени и работающее программное обеспечение зачастую было приспособлено для семибитных кодировок, что приводило, например, к тому, что почтовый сервер при передаче письма обнулял старшие биты в каждом байте сообщения. Одним из решений проблемы стала кодировка (а точнее – алгоритм) base64. В PGP алгоритм base64 используется для кодирования бинарных данных.

Кодирование base64 разработано для представления произвольных последовательностей октетов в форме, позволяющей использовать строчные и прописные буквы. Используется 65-символьное подмножество набора символов US-ASCII, обеспечивающее представление одним печатным символом 6 битов данных (дополнительный 65-й символ используется для обозначения функции специальной обработки).

Процесс кодирования представляет группу из 24 последовательных битов в форме строки из 4 символов. Обработка выполняется слева направо, а 24-битная исходная группа образуется конкатенацией трех 8-битных групп (байтов). Данные 24 бита после этого трактуются как 4 сцепленных группы по 6 битов, каждая из которых транслируется в один символ алфавита base64.

Каждая 6-битная группа используется в качестве индекса массива из 64 печатных символов. Символы алфавита, соответствующие индексу, помещаются в выходную строку. Кодирование base64 с безопасным алфавитом используется для представления URL и имен файлов. В табл. 3.1 перечислен алфавит, используемый для base64- кодировки. Значения представлены в различных системах счисления: десятичной (10), двоичной (2), восьмеричной (8) и шестнадцатеричной (16 или hex).

Поскольку входная информация base64 всегда включает целое число октетов, возможны лишь перечисленные ниже случаи:

• размер финального блока кодирования на входе кратен 24 битам, кодированный результат будет содержать целое число 4-символьных групп без заполнения символами «=»;

• размер финального блока кодирования на входе составляет 8 битов, выходной блок будет представлять 2 символа, дополненные последовательностью из двух символов заполнения «==»;

• размер финального блока кодирования на входе составляет 16 битов, выходной блок будет представлять 3 символа, дополненные символом заполнения «=».

**Практическое задание**

1) создать собственное приложение (приветствуется!) или воспользоваться Base64-онлайн кодировщиком, с помощью которого конвертировать произвольный документ (а) на латинице (можно использовать документ из ЛР № 2) в документ (б) формата base64.

2) с помощью приложения, созданного в ЛР № 2, получить распределение частотных свойств алфавитов по документам (а) и (б). Вычислить энтропию Хартли и Шеннона, а также избыточность алфавитов. Объяснить полученный результат.

public static string createReport(){

double base64ES = LW2.EntropyOfAlphabet(LW2.Alphabets.Base64),

latinES = LW2.EntropyOfAlphabet(LW2.Alphabets.Latin),

base64EC = Math.Log2("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/".Length),

latinEC = Math.Log2("qwertyuiopasdfghjklzxcvbnm".Length);

string report = "";

report += "Base64 entropy (Shennon): " + base64ES + "\n";

report += "Latin entropy (Shennon): " + latinES + "\n";

report += "Base64 entropy (Chartly): " + base64EC + "\n";

report += "Latin entropy (Chartly): " + latinEC + "\n";

report += "Redundancy base64: " + (base64EC - base64ES) / base64EC \* 100 + "%\n";

report += "Redundancy latin: " + (latinEC - latinES) / latinEC \* 100 + "%\n";

return report;}

Листринг 1– Вывод результатов программы

public static string toBase64(string path)

{

string text;

using (StreamReader sr = new StreamReader(path))

{

text = sr.ReadToEnd();

}

return Convert.ToBase64String(System.Text.Encoding.Unicode.GetBytes(text));}

Листринг 2– Перевод в Base64

public static void createBase64Doc()

{

using (FileStream fs = new FileStream("base64.txt", FileMode.OpenOrCreate))

{

fs.Write(System.Text.Encoding.Unicode.GetBytes(toBase64("latin.txt")));

}

}

Листринг 3– Создание файла с данными функции toBase64()

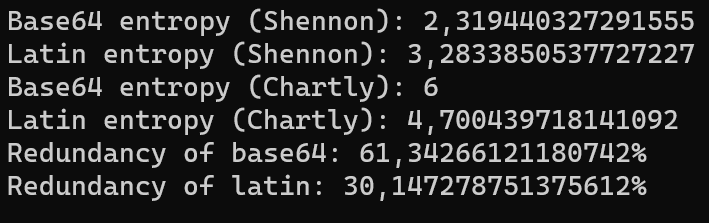


Рисунок 1 – Результат программы

3) написать функцию, которая принимает в качестве аргументов два буфера (а и b) одинакового размера и возвращает XOR (собственная фамилия (а) и имя (b); при разной длине меньшую дополнить нулями). Входные аргументы представлять 1) в кодах ASCII, 2) в кодах base64. Что будет результатом операции аXORbXORb? При написании не использовать стандартные функции языка программирования. Итоговые данные сравните с результатами, используя стандартные функции языка программирования (если они есть).

public static byte[] XOR(byte[] buf1, byte[] buf2)

{

byte[] ans = new byte[buf1.Length];

if (buf1.Length == buf2.Length)

{

for (int i = 0; i < buf1.Length; i++)

{

ans[i] = (byte)(buf1[i] ^ buf2[i]);

}

}

return ans;}

Листринг 4– Функция строгой дизъюнкции

static void Main(string[] args)

{

LAB3.createBase64Doc();

Console.WriteLine(LAB3.createReport());

Console.Write("ASCII XOR: ");

string surname = "Halaleenko",

name = "Andrew",

surnameASCII = "",

nameASCII = "";

foreach (var ch in surname)

surnameASCII += Convert.ToInt32(ch);

foreach (var ch in name)

nameASCII += Convert.ToInt32(ch);

while (surnameASCII.Length != nameASCII.Length)

nameASCII += '0';

foreach (var ch in LAB3.XOR(Encoding.Unicode.GetBytes(surnameASCII), Encoding.Unicode.GetBytes(nameASCII)))

Console.Write(Convert.ToString(ch, 2));

Console.WriteLine();

Console.Write("Base64 XOR: ");

string name64 = Convert.ToBase64String(Encoding.Unicode.GetBytes(name)),

surname64 = Convert.ToBase64String(Encoding.Unicode.GetBytes(surname));

while (surname64.Length != name64.Length)

name64 += '0';

foreach (var ch in LAB3.XOR(Encoding.Unicode.GetBytes(surname64), Encoding.Unicode.GetBytes(name64)))

Console.Write(Convert.ToString(ch, 2));

Console.WriteLine();

Console.WriteLine();

Console.Write("aXORbXORb: \t");

var aXORbXORb = LAB3.XOR(Encoding.Unicode.GetBytes(surname64), LAB3.XOR(Encoding.Unicode.GetBytes(name64), Encoding.Unicode.GetBytes(surname64)));

foreach (var ch in aXORbXORb)

Console.Write(Convert.ToString(ch, 2));

Console.WriteLine(); }

Листинг 5– Точка входа приложения

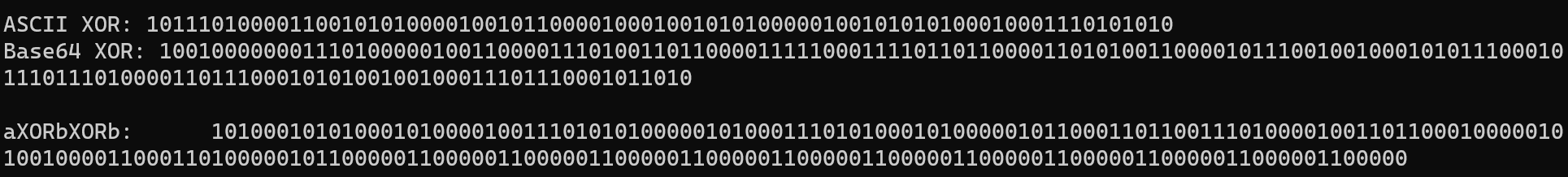


Рисунок 2 – Результат программы

**Вывод:** в данной работе были проверены энтропийные свойства алфавитов. Были рассмотрены энтропии Шеннона и Хартли. Опробованы и реализованы их алгоритмы. С их помощью были оценены энтропийные показатели алфавитов. Было выяснено, что алфавиты с большим количеством символов являются избыточными и информацию, которую несет каждый символ алфавита, уменьшается.