Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет систем управления и робототехники

Лабораторная работа №1

Кодирование и шифрование

Студент: Зелепугин А. Ю.

Группа: R3242

Преподаватель: Перегудин А.А.

Санкт-Петербург

2023

Добро пожаловать в первый в мире интерактивный отчёт о выполнении лабораторной работы. Он тесно связан с кодом, которым я её выполнял, поэтому код – и есть отчёт, который генерирует сам себя, в связи с этим рекомендую сходить, скачать и запустить исходный код на python, который лежит на Git Hub и обозван «Зелепугин_R3242_Лаб№1.py», чтобы читать в тёмной теме в PyCharm, а не тут на белом фоне.

Тем не менее, в этом документе тоже будет представлен отчёт, но с ним уже не удастся повзаимодействовать.

Оглавление:

Задание 1

Я взял русский алфавит из строчных, заглавных и добил символов до 63, потому что это наиболее близкое простое число к 64 как 2 в 6 степени, в 31 символ не интересно укладываться, ведь тогда не будет даже пробелов, а в 127 тупо лень.

Алфавит получился: абвгдежзиклмнопрстуфхцчшыэюяАБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЫЭЮЯ/., -01

Ценным сообщением послужило Капец Ценное.

Используя матрицу
[3 2]
[1 2]
С определителем 4.0
шифрован: ХКЫыЯоГ -ИЦш
дешифрован: Капец Ценное
испорчен: жЖ-ыЯоГ -ИЦш
дешифрован испорченный: Цыугц Ценное
испорчен снова и иначе: жКЫыЖоГ--ИЦш
дешифрован испорченный: сППеПттЭнное

Используя матрицу
[3 2 1]
[1 2 3]
[1 1 2]
С определителем 4.0
шифрован: 1свЭЗуРДрдЭЗ
дешифрован: Капец Ценное
испорчен: жЖ-ЭЗуРДрдЭЗ
дешифрован испорченный: ИЧЧец Ценное
испорчен снова и иначе: жсвЭЖуР-рдЭЗ
дешифрован испорченный: шЦАЭднАНеное

Используя матрицу
[2 4 1 2]
[1 2 3 1]
[2 1 4 3]
[2 3 2 3]
С определителем 10.0
шифрован: ЗцфЮчМ-тДтНЛ
дешифрован: Капец Ценное
испорчен: жЖ-ЮчМ-тДтНЛ
дешифрован испорченный: чэТц Ценное
испорчен снова и иначе: жцфЮЖМ--ДтНЛ
дешифрован испорченный: ПГТЖДЖЖюнное

Следует обратить внимание на то, что в результате внешнего вмешательства менялся не один символ, а комбинация из символов, по длине равная стороне шифрующей матрицы.
Это связано с матричным умножением, которое лежит в основе метода.
Таким образом, если бы я, шифруя матрицей 4х4, портил 1-й, 5-й, 9-й символы, потенциально оказались бы

не подверженными восстановлению группы 1-4, 5-8, 9-12 символы, то есть всё сообщение.

На 5 строке я определил функцию, превращающую слово в число.
На 22 - нахождение обратной матрицы по модулю, воспользовавшись тем,
что от обычного нахождения обратной она отличается только определителем
На 48 - само шифрование Хилла, основанное на матричном умножении

Задание 2

Представил, что у меня на руках два зашифрованных сообщения
СсОИ,ХмчДНЕЕШ Н
и
тГпаХЕЧШЦСТЗиХЮ

Известно, что в них использовался шифр Хилла с одним и тем же ключом,
который мне неизвестен и генерируется функций на строке 110.
Использовал по назначению алфавит из предыдущего задания.
Но, чтобы было сильно интереснее, познавательнее, применимее к жизни,
Ключ-то мой размера 3 на 3!
Тем не менее, в статье Википедии напрямую сказано, что была механическая машина для
ключа 6 на 6, но мне как будто делать нечего.
Соответственно, и сообщения я придумал не из 12, а из 15 символов, так снова веселее:
Это уже длиннее
и
х1ЦяВнГЮпЮ/АпбЮ
Второе сообщение генерируется случайным образом в строке 143-145
и используется только для задания сообщения и вывода в самом конце (ну и в этом тексте).
В этом можно убедиться поиском по коду с помощью ctrl+f для __cs2
Таким образом я имитировал потерю одного из исходных сообщений.

Далее остаётся только понять, что шифрование - это Ax=y,
что x и y нам известны, а найти нужно A.
Матрица находится через транспонирование одной, что становится практически очевидным, если вручную провести процесс шифровки.
Это вызвано тем, что обычно в СЛАУ Ax=y находится x, а у нас A.
Учитывая то, что длина сообщения, которое я шифровал больше, чем квадрат размерности матрицы, то есть больше 9 в моём случае, данных у меня даже с избытком.
Также упомяну, что матрицу я нахожу построчно (повекторно)
в силу того, что превращаю наборы Ax=y в наборы X*a1=y,
а решением этой штуки станет вектор a1, у меня строка матрицы A.
Так я обнаружил прямую матрицу, шифрующую исходное сообщение. Функция нахождения обратной у меня уже есть, так что остаётся только ее применить:

Получил, что исходно было х1ЦяВнГЮпЮ/АпбЮ
А на самом деле было х1ЦяВнГЮпЮ/АпбЮ
(Повторюсь, эта переменная взята из скрытых и не использовалась при подсчёте матриц - см строки 168-171).
То есть успешно проведены оперативно-розыскные работы, обнаружена ключ-матрица
и дешифровано неизвестное сообщение.

Задание 3

Алфавит из 32 букв: абвгдежзиклмнопрстуфхцчшыэюя .,-

Матрица G - на самом деле её строки должны быть базисом пространства кодовых слов.

То есть линейные комбинации строк G дают все возможные правильные (без ошибок) слова Хэмминга.
Таким образом, мной была выбрана такая порождающая матрица
[[1 0 0 0 0 1 1]
[0 1 0 0 1 0 1]
[0 0 1 0 1 1 0]
[0 0 0 1 1 1 1]]

Замечу, что на самом деле левая часть матрицы представляет все возможные позиции 1 на 4 разрядах.
Соответственно, последние три бита - соответствующие этим единицам проверочные биты.
Я решил писать их в конец, потому что могу.
Если бы писал в начало, то правая часть 4 на 4 могла бы быть единичной матрицей.
Могла БЫ потому, что

Можно было бы заняться извращениями и избрать другой базис для всех 4-буквенных комбинаций.
И тогда я бы посчитал для них проверочные биты, и приписал бы каждой строке соответствующую комбинацию
Причём не обязательно в начало, можно разместить как угодно, но удобно - как удобно.

Матрица H - в случае Хэмминга 7, 4 - 7 на 3. У меня такая:
[[0 1 1 1 1 0 0]
[1 0 1 1 0 1 0]
[1 1 0 1 0 0 1]]

Снова обратим внимание на то, что справа (то есть там, где я пишу проверочные биты) образовалась единичная матрица 3 на 3, что легко трактуется как базис всех возможных комбинаций проверочных битов
Ну а слева я по приколу раскидал какие-то циферки.)
На самом деле, конечно, это символы, соответствующие сообщениям, которым, в свою очередь, соответствуют данные проверочные символы.

Опять же, если бы писал проверочные биты не в конец, то единичная матрица не стояла бы в конце,
Например, матрица H запросто могла бы выглядеть так, если бы моим сообщением
Были 2, 4, 6 и 7 биты, а пров. символами 1, 3 и 5, причем в качестве базиса проверочных битов я бы выбрал
[[1 1 1]
[0 1 1]
[0 0 1]]
Тогда моя матрица H
[[1 0 1 0 1 0 1]
[0 0 1 1 1 1 0]
[0 1 0 1 1 0 1]]
Аналогичный пример можно придумать для G, но я не буду, мне лень: в ней больше цифр.

Итого в качестве интересного 4-х буквенного слова я выбрал: мило
Закодированное при помощи G слово с проверочными битами:
[0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 1 1 1 1 0 1 0 0 1 0 1 1 1 0 1 0 0 1]
На строке 336 успешно вредоносно вмешался в пятый символ, теперь слово такое:
[0 1 0 1 1 1 0 1 0 1 0 1 0 0 0 1 1 1 1 0 1 0 0 1 0 1 1 1 0 1 0 0 1]
Декодировал это вмешанное сообщение, получил: мило

На строке 341 инвертировал два символа, теперь слово такое:
[0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 1 1 0 1 0 0 1]
Декодировал это вмешанное сообщение, получил мило

А тут (352) инвертировал два подряд, слово такое:
[0 1 0 0 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 1 1 0 1 0 0 1]
Декодировал это вмешанное сообщение, получил эило

Далее по заданию сменил 3 символа, слово такое:
[0 1 0 0 0 1 0 1 0 1 1 1 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 1 1 0 0 0 0 1]
Декодировал это вмешанное сообщение, получил мило

Естественно, было бы крайне интересно проверить подряд 3 символа (строки 373-375), слово такое:
[1 0 1 0 0 1 0 1 0 1 1 1 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 1 1 0 0 0 0 1]
Декодировал это вмешанное сообщение, получил шило

Было бы странно остановиться, так что 4 символа подряд, слово такое:
[0 1 0 1 0 1 0 1 0 1 1 1 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 1 1 0 0 0 0 1]
Декодировал это вмешанное сообщение, получил мило

Меня сильно удивило то, что сообщение декодировалось правильно, попробую ещё раз
[0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 1 1 0 0 0 0 1]
Декодировал это вмешанное сообщение, получил лало

В результате приведённых выше изысканий выяснил, что "внезапно" если в каждом из блоков кодировки, то есть группах по 7 (для (7,4)),
допущено не более ошибки, код, предназначенный для исправления не более одной ошибки,
исправляет эти ошибки.

При этом если допустить более 1 ошибки на блок, то код перестанет корректно расшифровывать сообщение,
И, если портить два символов в двух блоках (по заданию нельзя портить больше 4), можно испортить
две буквы, хотя задействовать два блока для этого не обязательно, как показано в последнем примере,
где "сломались" две буквы, но корректировался только один блок.
Это вызвано тем, что буквы кодируются 5 знаками, то есть блок может кодировать "половину" одной буквы
и "половину другой". Почему при порче 4 знаков предпоследний раз я получил исходное слово,
мне пока нет времени анализировать, назову это "чудом"

Если вы внимательно смотрели мой код, то выяснили,
что функции для 3-го задания я задал в пространстве его имён.
Так и надо.

Задание 4

ЭссеСеСер

Итак, дано: поле 64x64, ключ, два состояния каждой клетки.
Очевидно, так и хочется принять позицию ключа за позицию ошибки, раз уж код её ищет.
В хорошем случае (что я предварительно обсужу со вторым, раз уж задача допускает)
будет перевёрнуто 0 или 64 монеты. Тогда я инвертирую одну, и это ответ.
Иначе попробуем воспользоваться кодом Хэмминга (63,57).
Такой код может указать на 1 ошибку в блоке из 63 символов.
Соответственно, порождающая матрица, пусть её левая часть снова станет единичной матрицей,
то есть матрицей 57 на 57 - она базис пространства исходного сообщения.
Тогда справа к ней припишем соответствующие проверочные биты (6 штук).
Итого dim(G) = 57 на 63
Так думал молодой Евгений (Андрей), пока не понял, что мы, похоже, не кодируем сообщение,
Нам не важно, что было сначала, это случайность, по сути, сообщение -
это случайное сообщение с ошибкой (ну или без), так что матрица G нам истинно не нужна.

Проверочная матрица будет состоять из двух частей: справа (там, где мы писали проверочные биты)
будет матрица с базисом, с помощью которого получаемы все комбинации слов из 6 нулей и единиц,
Снова упрощая, единичная матрица 6 на 6.
Справа для неё напишем те исходные сообщения, для которых получаются данные наборы проверочных битов.
Тогда dim(H) = 6 на 63
Эту матрицы я согласую с товарищем и далее:
Подозреваю, нужно выяснить, на какую ячейку указывал бы код Хэмминга,
если бы стоял в исходном положении,дальше прикинуть, какое число нужно перевернуть,
чтобы xor полученного с этим числом дал бы то число,

под которым лежит ключ.
ВАЖНО: перед побегом согласовать угол, с которого считается начало координат,
например, левый дальний от входа.
То есть моя задача - перевернуть так, чтобы xor дал координату,
А его задача - умножить вектор значений поля на матрицу H.
Естественно, он умножит все, кроме 0-го бита, то есть с первого по последний.
Если синдром (вектор ошибки = 0 вектору), то и ключ в левой дальний от входа клетке.
Удачного побега!

Вывод

В результате тщательно проделанной работы,
притронулся к кодированию и шифрованию информации, а именно:

1. Воспользовался шифром Хилла;
2. Взломал шифр Хилла;
3. Кодировался кодом Хэмминга;
4. Предположил, как сбежать при помощи кода Хэмминга из 6 битной тюрьмы.