

Содержание

Об авторе	20
О технических рецензентах	20
ВВЕДЕНИЕ	21
Для кого предназначена эта книга	22
Структура книги	23
Как пользоваться книгой	25
Ввод исходного кода	25
Исправление опечаток	26
Правила оформления листингов, принятые в книге	26
Ресурсы в Интернете	26
Загрузка и установка Python	27
Windows	27
macOS	27
Загрузка файла <i>pyperclip.py</i>	28
Запуск IDLE	28
Резюме	29
Ждем ваших отзывов!	30
ГЛАВА 1	
ИНСТРУМЕНТЫ “БУМАЖНОЙ” КРИПТОГРАФИИ	31
Что такое криптография	32
Коды и шифры	32
Шифр Цезаря	34
Шифровальный диск	35
Шифрование сообщения с помощью шифровального диска	36
Дешифрование сообщения с помощью шифровального диска	37
Шифрование и дешифрование средствами арифметики	37
Почему не работает двойное шифрование	39
Резюме	39
ГЛАВА 2	
ПРОГРАММИРОВАНИЕ В ИНТЕРАКТИВНОЙ ОБОЛОЧКЕ	41
Простые арифметические выражения	42
Целые и вещественные числа	43
Выражения	43
Порядок операций	44

Вычисление выражений.....	44
Сохранение значений в переменных.....	46
Изменение значений переменных.....	48
Имена переменных.....	49
Резюме.....	50

ГЛАВА 3

СТРОКОВЫЙ ТИП ДАННЫХ И НАПИСАНИЕ ПРОГРАММ 53

Использование строковых значений для работы с текстом.....	54
Конкатенация строк с помощью оператора +.....	55
Репликация строк с помощью оператора *.....	56
Получение символов строки с помощью индексов.....	57
Вывод значений с помощью функции print ().....	60
Вывод экранированных символов.....	61
Одинарные и двойные кавычки.....	63
Написание программ в редакторе файлов IDLE.....	64
Исходный код программы “Hello, world!”.....	64
Проверка правильности исходного кода с помощью онлайн-утилиты Diff.....	65
Использование IDLE для последующего доступа к программе.....	66
Сохранение программы.....	66
Выполнение программы.....	67
Открытие программы, которую вы сохранили ранее.....	68
Как работает программа “Hello, world!”.....	68
Комментарии.....	69
Вывод сообщений для пользователя.....	69
Ввод данных пользователем.....	70
Завершение программы.....	70
Резюме.....	71

ГЛАВА 4

ОБРАТНЫЙ ШИФР 73

Исходный код программы Reverse Cipher.....	74
Пробный запуск программы Reverse Cipher.....	74
Ввод комментариев и установка переменных.....	75
Определение длины строки.....	76
Знакомство с циклом while.....	77
Булев тип данных.....	77
Операторы сравнения.....	78
Блоки.....	81

Инструкция <code>while</code>	82
“Нарращивание” строки	83
Усовершенствование программы за счет использования функции <code>input()</code>	86
Резюме	86

ГЛАВА 5

ШИФР ЦЕЗАРЯ

89

Исходный код программы <code>Caesar Cipher</code>	90
Пример выполнения программы <code>Caesar Cipher</code>	91
Импорт модулей и установка переменных	92
Константы и переменные	93
Цикл <code>for</code>	94
Пример цикла <code>for</code>	95
Цикл <code>while</code> , эквивалентный циклу <code>for</code>	95
Инструкция <code>if</code>	96
Пример инструкции <code>if</code>	96
Инструкция <code>else</code>	97
Инструкция <code>elif</code>	98
Операторы <code>in</code> и <code>not in</code>	99
Строковый метод <code>find()</code>	100
Шифрование и дешифрование символов	101
Обработка “завертывания” символьного набора	102
Обработка символов, не включенных в символьный набор	103
Вывод и копирование преобразованной строки	103
Шифрование других символов	104
Резюме	105

ГЛАВА 6

ВЗЛОМ ШИФРА ЦЕЗАРЯ МЕТОДОМ ГРУБОЙ СИЛЫ

109

Исходный код программы <code>Caesar Hacker</code>	110
Пример выполнения программы <code>Caesar Hacker</code>	111
Установка переменных	112
Организация цикла с помощью функции <code>range()</code>	112
Дешифрование сообщения	114
Использование строкового форматирования для отображения ключа и дешифрованных сообщений	115
Резюме	116

ГЛАВА 7

ШИФРОВАНИЕ С ПОМОЩЬЮ ПЕРЕСТАНОВОЧНОГО ШИФРА

119

Как работает перестановочный шифр.....	120
Шифрование сообщения вручную.....	121
Создание программы шифрования.....	122
Исходный код программы <code>Transposition Encrypt</code>	123
Пример выполнения программы <code>Transposition Encrypt</code>	125
Создание собственных функций с помощью инструкции <code>def</code>	125
Определение функции с параметрами.....	126
Изменение параметров сказывается лишь внутри функции.....	127
Определение функции <code>main()</code>	128
Передача ключа и сообщения в качестве аргументов.....	129
Списковый тип данных.....	130
Изменение элементов списка.....	131
Вложенные списки.....	132
Применение функции <code>len()</code> и оператора <code>in</code> к спискам.....	133
Конкатенация и репликация списков с помощью операторов <code>+</code> и <code>*</code>	134
Алгоритм шифрования с помощью перестановочного шифра.....	134
Составные операторы присваивания.....	136
Перемещение текущего индекса по строке сообщения.....	137
Строковый метод <code>join()</code>	139
Возвращаемые значения и инструкция <code>return</code>	139
Пример инструкции <code>return</code>	140
Возврат зашифрованного шифротекста.....	140
Переменная <code>__name__</code>	141
Резюме.....	142

ГЛАВА 8

ДЕШИФРОВАНИЕ ПЕРЕСТАНОВОЧНОГО ШИФРА

145

Как дешифровать на бумаге текст, зашифрованный с помощью перестановочного шифра.....	146
Исходный код программы <code>Transposition Decrypt</code>	147
Пример выполнения программы <code>Transposition Decrypt</code>	149
Импорт модулей и функция <code>main()</code>	149
Дешифрование сообщения с помощью ключа.....	150
Функции <code>round()</code> , <code>math.ceil()</code> и <code>math.floor()</code>	150
Функция <code>decryptMessage()</code>	151
Булевы операторы.....	153
Настройка переменных <code>column</code> и <code>row</code>	157

Вызов функции <code>main()</code>	159
Резюме	159

ГЛАВА 9

НАПИСАНИЕ ТЕСТОВ

161

Исходный код программы <code>Transposition Test</code>	162
Пример выполнения программы <code>Transposition Test</code>	163
Импорт модулей.....	164
Создание псевдослучайных чисел.....	164
Создание случайной строки.....	166
Дублирование строки произвольное число раз	167
Списковые переменные используют ссылки	168
Передача ссылок	170
Использование функции <code>copy.deepcopy()</code> для дублирования списка.....	171
Функция <code>random.shuffle()</code>	171
Случайное перемешивание строки.....	172
Тестирование каждого сообщения.....	172
Проверка корректности результата и завершение программы.....	174
Вызов функции <code>main()</code>	174
Тестирование программы-тестера	175
Резюме	175

ГЛАВА 10

ШИФРОВАНИЕ И ДЕШИФРОВАНИЕ ФАЙЛОВ

177

Простые текстовые файлы	178
Исходный код программы <code>Transposition File Cipher</code>	178
Пример выполнения программы <code>Transposition File Cipher</code>	180
Работа с файлами	181
Открытие файлов	181
Запись и закрытие файлов.....	182
Чтение из файла	183
Функция <code>main()</code>	183
Проверка существования файла	184
Функция <code>os.path.exists()</code>	184
Проверка существования файла с помощью функции <code>os.path.exists()</code>	185
Строковые методы, используемые для повышения гибкости пользовательского ввода	185
Строковые методы <code>upper()</code> , <code>lower()</code> и <code>title()</code>	186
Строковые методы <code>startswith()</code> и <code>endswith()</code>	186

Использование строковых методов в программе	187
Чтение входного файла	188
Измерение затрат времени на шифрование и дешифрование.....	188
Модуль <code>time</code> и функция <code>time.time()</code>	188
Использование функции <code>time.time()</code> в программе.....	189
Запись в выходной файл.....	190
Вызов функции <code>main()</code>	190
Резюме	191

ГЛАВА 11

ПРОГРАММНОЕ РАСПОЗНАВАНИЕ АНГЛИЙСКИХ СЛОВ

193

Может ли компьютер понимать английский язык?	194
Исходный код модуля <code>Detect English</code>	196
Применение модуля <code>Detect English</code>	197
Указания по использованию модуля и установка констант.....	198
Словарный тип данных	198
Различие между словарями и списками.....	200
Добавление и изменение элементов словаря	200
Применение функции <code>len()</code> к словарям	201
Применение оператора <code>in</code> к словарям	202
Поиск элементов в словарях выполняется быстрее, чем в списках	202
Использование циклов <code>for</code> со словарями.....	203
Реализация файла словаря	203
Метод <code>split()</code>	204
Разбивка файла словаря на отдельные слова	204
Возврат данных в виде словаря	205
Подсчет количества английских слов в сообщении	206
Ошибка деления на нуль.....	206
Считаем английские слова.....	207
Функции <code>float()</code> , <code>int()</code> и <code>str()</code> и целочисленное деление.....	208
Нахождение доли английских слов в сообщении	209
Удаление небуквенных символов	209
Метод <code>append()</code> списка.....	210
Создание строки, объединяющей буквы.....	211
Распознавание английских слов	211
Использование аргументов по умолчанию.....	212
Вычисление процентной доли	213
Резюме	215

ГЛАВА 12

ВЗЛОМ ПЕРЕСТАНОВОЧНОГО ШИФРА 217

Исходный код программы Transposition Hacker.....	218
Пример выполнения программы Transposition Hacker.....	219
Импорт модулей.....	220
Создание многострочного текста с помощью тройных кавычек.....	221
Отображение результатов взлома сообщения	222
Получение взломанного сообщения.....	222
Строковый метод strip()	224
Применение строкового метода strip()	225
Невозможность взлома сообщения.....	226
Вызов функции main()	226
Резюме	226

ГЛАВА 13

АФФИННОЕ ШИФРОВАНИЕ С ПОМОЩЬЮ МОДУЛЬНОЙ АРИФМЕТИКИ 229

Модульная арифметика.....	230
Оператор деления с остатком	231
Нахождение множителей для вычисления наибольшего общего делителя	232
Групповое присваивание	234
Алгоритм Евклида для нахождения НОД.....	235
Как работают мультипликативный и аффинный шифры.....	236
Выбор допустимых мультипликативных ключей.....	237
Шифрование с помощью аффинного шифра.....	238
Дешифрование с помощью аффинного шифра	239
Вычисление модульных обращений.....	240
Оператор целочисленного деления.....	241
Исходный код модуля Cryptomath.....	242
Резюме	243

ГЛАВА 14

ПРОГРАММИРОВАНИЕ АФФИННОГО ШИФРА 245

Исходный код программы Affine Cipher.....	246
Пример выполнения программы Affine Cipher.....	248
Импорт модулей, настройка констант и функция main()	248
Вычисление и проверка ключей.....	250
Кортежи.....	251
Выявление слабых ключей	251
Сколько ключей может иметь аффинный шифр.....	253

Написание функции шифрования	255
Написание функции дешифрования	256
Генерирование случайных ключей.....	257
Вызов функции <code>main()</code>	258
Резюме	259

ГЛАВА 15

ВЗЛОМ АФФИННОГО ШИФРА 261

Исходный код программы <code>Affine Hacker</code>	262
Пример выполнения программы <code>Affine Hacker</code>	263
Импорт модулей, настройка констант и функция <code>main()</code>	264
Функция взлома аффинного шифра	265
Оператор возведения в степень	265
Вычисление общего количества возможных ключей.....	266
Инструкция <code>continue</code>	267
Использование инструкции <code>continue</code> для пропуска кода.....	268
Вызов функции <code>main()</code>	269
Резюме	270

ГЛАВА 16

ПРОГРАММИРОВАНИЕ ПРОСТОГО ПОДСТАНОВОЧНОГО ШИФРА 271

Как работает простой подстановочный шифр	272
Исходный код программы <code>Simple Substitution Cipher</code>	273
Пример выполнения программы <code>Simple Substitution Cipher</code>	275
Импорт модулей, настройка констант и функция <code>main()</code>	276
Списковый метод <code>sort()</code>	277
Функции-обертки	279
Функция <code>translateMessage()</code>	280
Строковые методы <code>isupper()</code> и <code>islower()</code>	282
Сохранение регистра букв с помощью метода <code>isupper()</code>	283
Генерирование случайных ключей.....	284
Вызов функции <code>main()</code>	285
Резюме	285

ГЛАВА 17

ВЗЛОМ ПРОСТОГО ПОДСТАНОВОЧНОГО ШИФРА 287

Дешифрование с использованием словарных шаблонов.....	288
Поиск шаблонов слов.....	288
Поиск возможных вариантов дешифрования букв	289
Обзор процесса взлома	291
Модуль <code>Word Patterns</code>	292

Исходный код программы Simple Substitution Hacker	293
Пример выполнения программы Simple Substitution Hacker.....	296
Импорт модулей и настройка констант.....	297
Поиск символов с помощью регулярных выражений.....	298
Функция main()	298
Вывод результатов взлома	299
Создание словарей шифробукв.....	300
Создание пустого словаря шифробукв	300
Добавление букв в дешифровальный словарь	300
Пересечение двух словарей.....	302
Как работают вспомогательные функции	303
Выявление достоверно установленных букв в словарях.....	307
Тестирование функции removeSolvedLetterFromMapping()	309
Функция hackSimpleSub()	310
Строковый метод replace()	312
Дешифрование сообщения	313
Дешифрование сообщения в интерактивной оболочке	315
Вызов функции main()	316
Резюме	316

ГЛАВА 18

ПРОГРАММИРОВАНИЕ ШИФРА ВИЖЕНЕРА

319

Использование многобуквенных ключей в шифре Виженера.....	320
Чем длиннее ключ шифра Виженера, тем он надежнее.....	322
Выбор ключа, предотвращающего словарные атаки.....	323
Исходный код программы Vigenere Cipher	323
Пример выполнения программы Vigenere Cipher.....	325
Импорт модулей, настройка констант и функция main()	325
Создание строк с помощью списковых методов append() и join()	326
Шифрование и дешифрование сообщения.....	328
Вызов функции main()	331
Резюме	331

ГЛАВА 19

ЧАСТОТНЫЙ АНАЛИЗ

333

Анализ частотности букв в тексте	334
Частотное соответствие букв	336
Вычисление оценки частотного соответствия букв для простого подстановочного шифра.....	337
Вычисление оценки частотного соответствия букв для перестановочного шифра.....	338

Использование частотного анализа в случае шифра Виженера	339
Исходный код программы <code>Frequency Analysis</code>	340
Сохранение букв алфавита в порядке <code>ETAOIN</code>	341
Подсчет букв в сообщении	342
Получение первого элемента кортежа.....	344
Упорядочение букв, встречающихся в сообщении, в соответствии с частотностью	344
Подсчет букв с помощью функции <code>getLetterCount()</code>	345
Создание словаря счетчиков частотности со списками букв.....	345
Сортировка списков букв в порядке, обратном порядку <code>ETAOIN</code>	346
Сортировка списков словаря по частотности	351
Создание списка отсортированных букв	353
Вычисление оценки частотного соответствия букв в сообщении	353
Резюме	355

ГЛАВА 20

ВЗЛОМ ШИФРА ВИЖЕНЕРА

357

Использование перебора по словарю для взлома шифра Виженера методом грубой силы	358
Исходный код программы <code>Vigenere Dictionary Hacker</code>	358
Пример выполнения программы <code>Vigenere Dictionary Hacker</code>	359
О структуре программы	360
Использование метода Касиски для определения длины ключа	361
Нахождение повторяющихся сегментов	361
Определение множителей в интервалах повторения	362
Получение каждой <i>n</i> -й буквы строки.....	364
Применение частотного анализа для взлома каждого подключа	365
Перебор возможных подключей методом грубой силы.....	367
Исходный код программы <code>Vigenere Hacker</code>	368
Пример выполнения программы <code>Vigenere Hacker</code>	374
Импорт модулей и функция <code>main()</code>	375
Нахождение повторяющихся последовательностей.....	375
Вычисление множителей интервалов повторения	378
Удаление дубликатов с помощью функции <code>set()</code>	379
Удаление дублирующихся множителей и сортировка списка	380
Нахождение наиболее часто встречающихся множителей	381
Нахождение наиболее вероятной длины ключа	383
Списковый метод <code>extend()</code>	383
Расширение словаря <code>repeatedSeqSpacings</code>	384
Извлечение множителей из списка <code>factorsByCount</code>	385

Получение букв, зашифрованных одним и тем же подключом	386
Попытки дешифрования с помощью ключей вероятной длины	387
Аргумент <code>end</code> функции <code>print()</code>	390
Запуск программы в “тихом” режиме или с выводом информации для пользователя	390
Нахождение возможных комбинаций подключей	391
Вывод дешифрованного текста с сохранением корректного регистра букв	395
Получение взломанного сообщения	396
Выход из цикла, если найден потенциальный ключ	397
Тестирование всех остальных вариантов длины ключа методом грубой силы	398
Вызов функции <code>main()</code>	399
Изменение значений констант, используемых в программе	399
Резюме	400

ГЛАВА 21

ОДНОРАЗОВЫЙ ШИФРОБЛОКНОТ

403

Не поддающийся взлому одноразовый шифроблокнот	404
Выравнивание длины ключа по размеру сообщения	404
Создание истинно случайного ключа	406
Избегайте двухразовых шифроблокнотов	407
Почему двухразовый шифроблокнот эквивалентен шифру Виженера	408
Резюме	409

ГЛАВА 22

НАХОЖДЕНИЕ И ГЕНЕРИРОВАНИЕ ПРОСТЫХ ЧИСЕЛ

411

Что такое простое число	412
Исходный код модуля <code>primeNum</code>	414
Пример работы модуля <code>primeNum</code>	417
Как работает алгоритм перебора делителей	417
Реализация алгоритма перебора делителей	419
Решето Эратосфена	420
Генерирование простых чисел с помощью решета Эратосфена	422
Тест Миллера – Рабина для проверки простоты числа	423
Проверка больших простых чисел	424
Генерирование больших простых чисел	426
Резюме	426

ГЛАВА 23
ГЕНЕРИРОВАНИЕ КЛЮЧЕЙ ДЛЯ КРИПТОСИСТЕМ
С ОТКРЫТЫМ КЛЮЧОМ **429**

Криптосистемы с открытым ключом.....	430
Проблема аутентификации.....	432
Цифровые подписи	432
Остерегайтесь атак MITM.....	433
Порядок генерирования открытых и закрытых ключей	434
Исходный код программы Make Public Private Keys.....	435
Пример выполнения программы Make Public Private Keys	437
Создание функции main ()	438
Генерирование ключей с помощью функции generateKey ()	439
Вычисление значения e	439
Вычисление значения d	440
Возврат ключей	440
Создание файлов ключей с помощью функции makeKeyFiles ()	441
Вызов функции main ()	443
Гибридные криптосистемы.....	443
Резюме	444

ГЛАВА 24
ПРОГРАММА ШИФРОВАНИЯ С ОТКРЫТЫМ КЛЮЧОМ **445**

Как работают криптосистемы с открытым ключом.....	446
Создание блоков	446
Преобразование строки в блок.....	447
Арифметика шифрования и дешифрования с открытым ключом.....	449
Преобразование блока в строку	451
Почему нельзя взломать сообщение, зашифрованное с помощью открытого ключа	452
Исходный код программы Public Key Cipher.....	455
Пример выполнения программы Public Key Cipher	458
Начало программы.....	460
Как программа определяет, что ей делать: шифровать или дешифровать.....	460
Преобразование строк в блоки с помощью функции getBlocksFromText ()	462
Функции min () и max ()	463
Сохранение блоков в переменной blockInt.....	463
Дешифрование блоков с помощью функции getTextFromBlocks ()	465
Использование спискового метода insert ()	466

Объединение элементов списка <code>message</code> в одну строку	467
Функция <code>encryptMessage()</code>	467
Функция <code>decryptMessage()</code>	468
Чтение открытого и закрытого ключей из соответствующих файлов	469
Запись зашифрованного сообщения в файл	469
Дешифрование содержимого файла	472
Вызов функции <code>main()</code>	474
Резюме	474

ПРИЛОЖЕНИЕ А

ОТЛАДКА КОДА PYTHON

477

Как работает отладчик	477
Кнопка <code>Go</code>	478
Кнопка <code>Step</code>	479
Кнопка <code>Over</code>	479
Кнопка <code>Out</code>	479
Кнопка <code>Quit</code>	479
Отладка программы <code>Reverse Cipher</code>	480
Задание точек останова	482
Резюме	483

ПРИЛОЖЕНИЕ Б

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

485

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

503

1

ИНСТРУМЕНТЫ “БУМАЖНОЙ” КРИПТОГРАФИИ

“Джин шифрования выпущен из бутылки”.

Ян Кум, основатель WhatsApp



Прежде чем приступать к написанию шифровальных программ, давайте рассмотрим, как реализуется процесс шифрования и дешифрования с помощью карандаша и бумаги. Это позволит вам лучше понять природу шифров и тех математических инструментов, которые применяются для создания секретных сообщений. В данной главе вы узнаете о том, что такое криптография и чем коды отличаются от шифров. Затем мы воспользуемся простым шифром, известным как *шифр Цезаря*, для шифрования и расшифровки бумажных сообщений.

В этой главе...

- Что такое криптография
- Коды и шифры
- Шифр Цезаря
- Шифровальные диски
- Криптография и арифметика
- Двойное шифрование

Что такое криптография

С незапамятных времен все, кому требовалось обмениваться тайными сведениями, например шпионы, военные, пираты, торговцы и дипломаты, прибегали к засекречиванию своих сообщений, чтобы тайны оставались надежно скрыты от посторонних глаз. *Криптография* — это наука, которая изучает способы создания и применения секретных кодов. Чтобы понять, как работают криптографические методы, рассмотрим следующие два фрагмента текста.

ngr N.vNwz5uNz5Ns6620Nz0N3z2v	!NN2 Nuwv,N9,vNN!vNrBN3zyN4vN
N yvNwz9vNz5N6!9Nyvr9	N6 Qvw0z6nvN.7N0yv4N 4 zzvNN
y0QNnvNwv tyNz	vyN,NN99z0zz6wz0y3vv26 9
Nw964N6!9N5vzxs690,N.vN2z5u-	w296vyNNrrNyQst.560N94Nu5y
3vNz Nr Ny64v,N.vNt644!5ztr vNz	rN5nz5vv5t6v63zNr5.
N 6N6 yv90,Nr5uNz Nsvt64v0N	N75sz6966NNvw6 zu0 wfNxs6t
yvN7967v9 BN6wNr33Q N-m63 rz9v	49NrN3Ny9Nvzy!

Текст слева — секретное сообщение, которое было *зашифровано*, т.е. преобразовано в секретный код. Любому человеку, не знающему способа его *дешифрования*, или *расшифровки*, оно кажется полной абракадаброй. Сообщение справа — случайный набор символов, не имеющий никакого скрытого смысла. Шифрование позволяет сохранить смысл сообщения в тайне от тех, кому не известен способ его расшифровки, даже если сообщение попадет им в руки. *Шифрованное сообщение воспринимается посторонними как случайный набор букв, не несущий в себе никакого смысла.*

Криптограф, или *шифровальщик*, использует и изучает секретные коды. Разумеется, секретные сообщения не всегда остаются секретными. *Криптоаналитик*, т.е. *взломщик кодов*, или *хакер*, способен взламывать и читать сообщения, зашифрованные другими людьми. Цель книги — научить вас зашифровывать и расшифровывать сообщения с помощью различных методов. К счастью, все те методы взлома, которые вы изучите, не настолько опасны, чтобы у вас из-за них могли возникнуть проблемы с законом.

Коды и шифры

В отличие от шифров *коды* изначально создаются такими, чтобы они были понятны и общедоступны. Коды заменяют буквы символами, которые любой человек может использовать для перевода в форму сообщения.

В начале XIX века развитие электрического телеграфа привело к созданию известного кода, обеспечивавшего почти мгновенный обмен сообщениями между континентами по проводам. Сообщения, отправляемые по телеграфу, достигали своих адресатов гораздо быстрее, чем прежняя лошадиная почта, перевозившая мешки с письмами. Однако телеграф не позволял отправлять сообщения в том же виде, в каком они были написаны на бумаге, т.е. в виде последовательностей букв. По нему могли пересылаться только два типа электрических импульсов: короткий, который называли “точка”, и длинный, который называли “тире”.

Для преобразования букв алфавита в электрические импульсы необходимо располагать системой кодов, с помощью которой можно было бы переводить привычные для нас буквы на язык точек и тире. Процесс преобразования букв алфавита в последовательности точек и тире для отправки по телеграфу называется *кодированием*, а обратный процесс преобразования электрических импульсов в буквы при получении сообщения — *декодированием*. Способ, применяемый для кодирования и декодирования телеграфных сообщений (а впоследствии и сообщений, передаваемых по радио), был изобретен Сэмюэлем Морзе и Альфредом Вейлем и получил название *код Морзе*, или *азбука Морзе* (табл. 1.1).

Таблица 1.1. Международная азбука Морзе

Латинский символ	Код	Латинский символ	Код	Цифра	Код
A	•—	N	—•	1	•-----
B	—•••	O	----	2	••-----
C	—•—•	Q	•—••	3	•••---
D	—••	R	—•—	4	••••-
E	•	S	•—•	5	•••••
F	••—•	T	•••	6	—••••
G	—••	U	—	7	---•••
H	••••	V	••—	8	-----••
I	••	W	•••—	9	-----•
J	•----	X	•—•	0	-----
K	—•—	Y	—•—		
L	•—••	Z	—•••		
M	---				

Используя телеграфный ключ для передачи точек и тире, телеграфист отправлял текстовые сообщения, способные почти мгновенно достигать адресата, находящегося на другом конце земного шара¹!

В отличие от знакового кодирования *шифр* — специфический тип кодов, обеспечивающих сохранение содержания сообщения в тайне. Шифр можно использовать для того, чтобы преобразовать исходный текст, написанный на понятном языке (так называемый *простой текст*), в бессвязный набор символов, называемый *шифротекстом*, который скрывает смысл секретного сообщения. Шифр — это набор правил преобразования между простым и зашифрованным текстом. В правилах для шифрования и дешифрования часто используется один и тот же ключ, который называется *секретным* и известен только отправителю и получателю сообщения. В книге вы изучите несколько видов шифров и напишете программы для шифрования и дешифрования текста с их помощью. Но сначала следует научиться шифровать сообщения вручную, используя простые средства “бумажной” криптографии.

Шифр Цезаря

Первый из шифров, который мы изучим, — шифр Цезаря, названный так в честь Юлия Цезаря, который пользовался им 2000 лет тому назад. Хорошая новость состоит в том, что он прост и несложен для изучения. Но есть и плохая новость: в силу простоты этого шифра криптоаналитику не составит большого труда взломать его. Тем не менее ознакомление с ним даст вам полезный опыт.

Шифр Цезаря основан на замене одной буквы другой после предварительного смещения всего алфавита на определенное число позиций. Юлий Цезарь заменял буквы в своих сообщениях путем смещения алфавита на три позиции и последующей замены каждой буквы соответствующей буквой из смещенного алфавита.

Например, вместо каждой буквы ‘А’ он подставлял букву ‘D’, вместо каждой буквы ‘В’ — букву ‘Е’ и т.д. Если Цезарю нужно было сдвинуть букву, находящуюся в конце алфавита, скажем, ‘У’, то он возвращался в начало алфавита, смещаясь в целом на три позиции и подставляя букву ‘В’. В этом разделе мы будем шифровать сообщения вручную, применяя шифр Цезаря.

¹ Более подробную информацию об азбуке Морзе можно найти в Википедии: https://ru.wikipedia.org/wiki/Азбука_Морзе.

Шифровальный диск

Чтобы упростить преобразование простого текста в зашифрованный с помощью шифра Цезаря, мы будем использовать *шифровальный диск*. Он состоит из двух колец, каждое из которых разбито на 26 ячеек (по числу букв английского алфавита). Внешнее кольцо представляет алфавит исходного текста, а внутреннее – соответствующие буквы зашифрованного текста. Буквы на внутреннем кольце пронумерованы числами от 0 до 25. Эти числа определяют ключ шифрования, в данном случае – количество позиций, на которое нужно перейти от буквы 'А' к соответствующей букве на внутреннем кольце. Поскольку сдвиг выполняется по кругу, смещение с ключом, значение которого превышает 25, продолжается с начала алфавита, соответственно, смещение на 26 позиций равносильно отсутствию сдвига, смещение на 27 позиций – сдвигу на 1 позицию и т.д.

Виртуальный шифровальный диск доступен по адресу <https://inventwithpython.com/cipherwheel/> (рис. 1.1). Чтобы проверить диск, щелкните на нем один раз и перемещайте указатель мыши по кругу, пока не будет достигнута нужная конфигурация. Повторный щелчок останавливает дальнейшее вращение диска.

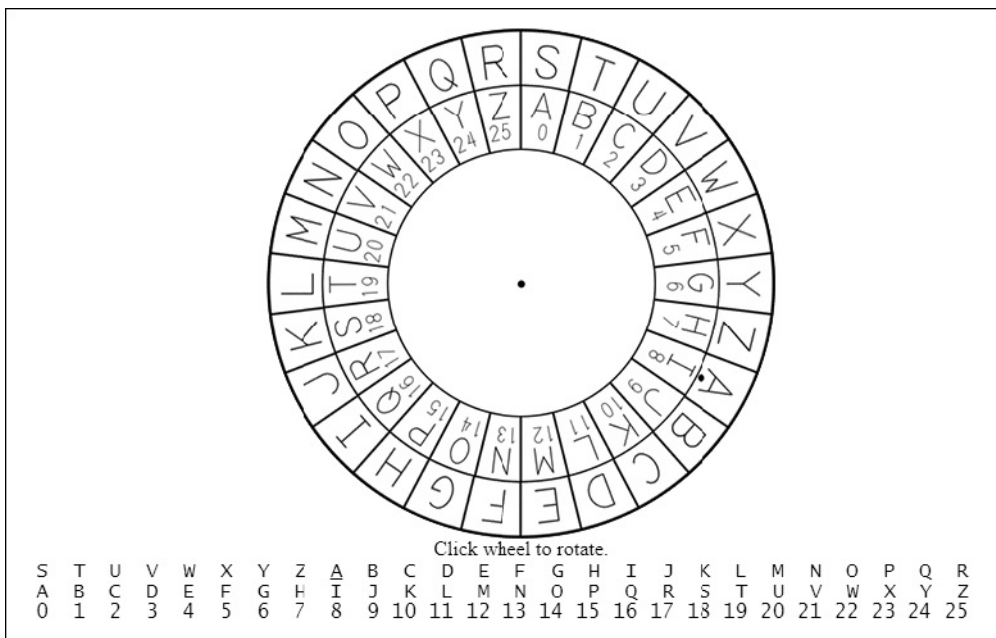


Рис. 1.1. Онлайн-шифровальный диск

Вы сможете получить бумажную версию шифровального диска, распечатав изображение, доступное по указанному адресу. Вырежьте два круга,

вложите меньший из них в больший и закрепите по центру булавкой, чтобы их можно было вращать.

Используя либо бумажную, либо виртуальную модель, вы сможете вручную зашифровать секретное сообщение.

Шифрование сообщения с помощью шифровального диска

В качестве примера зашифруем сообщение “THE SECRET PASSWORD IS ROSEBUD” с помощью онлайн-ового шифровального диска. Проверните диск так, чтобы деления внутреннего и внешнего кругов совпадали. Обратите внимание на изображение точки под буквой ‘А’. Число на внутреннем круге под этой точкой и есть ключ шифрования, который будет применяться при данной конфигурации диска.

На рис. 1.1 таким числом является 8. Мы используем его в качестве ключа для того, чтобы зашифровать сообщение (рис. 1.2).

T	H	E		S	E	C	R	E	T	P	A	S	S	W	O	R	D	I	S	R	O	S	E	B	U	D
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
B	P	M		A	M	K	Z	M	B	X	I	A	A	E	W	Z	L	Q	A	Z	W	A	M	J	C	L

Рис. 1.2. Шифрование сообщения с помощью шифра Цезаря по ключу 8

Найдите каждую букву сообщения на внешнем круге и замените ее соответствующей буквой из внутреннего круга. В данном примере сообщение начинается с буквы ‘Т’ (первая буква ‘Т’ во фразе “THE SECRET...”), поэтому находим букву ‘Т’ на внешнем круге, а затем соответствующую ей букву на внутреннем круге, которой является буква ‘В’. Таким образом, в исходном сообщении буква ‘Т’ всегда будет заменяться буквой ‘В’. (Если бы вы использовали другой ключ шифрования, то каждая буква ‘Т’ заменялась бы другой буквой.) Следующая буква сообщения – ‘Н’, которая превращается в букву ‘Р’. Далее буква ‘Е’ превращается в букву ‘М’. Каждая буква внешнего круга всегда шифруется одной и той же буквой внутреннего круга. Как только вы найдете первую букву ‘Т’ в сообщении “THE SECRET...” и увидите, что она шифруется буквой ‘В’, можете заменить буквой ‘В’ все буквы ‘Т’ в сообщении с целью экономии времени, чтобы каждую букву приходилось искать на шифровальном диске всего лишь раз.

Завершив процесс шифрования всего исходного сообщения “THE SECRET PASSWORD IS ROSEBUD”, вы получите результирующее зашифрованное сообщение “BPM AMKZMB XIAAEWZL QA ZWAMJCL”. Обратите внимание на то, что небуквенные символы, в данном случае пробелы, остаются неизменными.

Теперь вы сможете спокойно отправить зашифрованное сообщение нужному получателю (или оставить его при себе), и никто не сможет прочитать его, если вы не сообщите ему ключ шифрования. Убедитесь, что ключ хранится в надежном месте, поскольку любой человек, которому станет известно, что вы использовали ключ шифрования 8, сможет прочитать зашифрованное сообщение.

Дешифрование сообщения с помощью шифровального диска

Чтобы дешифровать зашифрованный текст, начните с внутреннего круга шифровального диска. Предположим, вы получили зашифрованный текст “IWT CTL EPHHLDGS XH HLDGSUXHW”. Вы не сможете расшифровать его, пока не узнаете ключ (если только вы не криптоаналитик). К счастью, отправитель уже сообщил вам, что в своих сообщениях он использует ключ 15. Конфигурация шифровального диска для этого ключа приведена на рис. 1.3.

Установите букву ‘A’ на внешнем круге (помечена точкой) напротив буквы на внутреннем круге с номером 15 (буква ‘P’). Затем найдите первую букву секретного сообщения на внутреннем круге (буква ‘I’) и запишите букву, которая соответствует ей на внешнем круге (буква ‘T’). Вторая буква секретного сообщения, ‘W’, дешифруется в букву ‘H’. Расшифровав все буквы зашифрованного текста, вы получите сообщение в виде простого текста: “THE NEW PASSWORD IS SWORDFISH” (рис. 1.4).

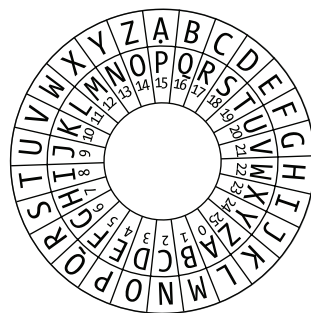


Рис. 1.3. Конфигурация шифровального диска для ключа 15

I	W	T		C	T	L		E	P	H	H	L	D	G	S		X	H		H	L	D	G	S	U	X	H	W
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
T	H	E		N	E	W		P	A	S	S	W	O	R	D		I	S		S	W	O	R	D	F	I	S	H

Рис. 1.4. Дешифрование сообщения с помощью шифра Цезаря по ключу 15

Если использовать некорректный ключ, например 16, то расшифрованное сообщение превратится в бессмысленный набор символов “SGD MDV OZRRVNQC HR”.

Шифрование и дешифрование средствами арифметики

Шифровальный диск — удобный инструмент для применения шифра Цезаря, но аналогичные операции можно выполнять и в арифметическом

виде. Запишите буквы алфавита от 'A' до 'Z' и пронумеруйте их числами от 0 до 25. Начните с нуля под 'A', единицей под 'B' и т.д., пока не поставите номер 25 под 'Z' (рис. 1.5).

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Рис. 1.5. Нумерация алфавита числами от 0 до 25

Полученную таблицу перекодировки букв в цифры можно использовать для представления букв, что позволяет выполнять над буквами арифметические операции. Например, если вы запишете слово "CAT" цифрами 2, 0 и 19, то сможете прибавить к каждой из них 3, получив в результате последовательность 5, 3 и 22. Эти новые числа представляют буквы "FDW". Позже мы сможем написать компьютерную программу, которая все сделает за нас.

Чтобы использовать арифметику для шифрования с помощью шифра Цезаря, найдите число под буквой, которую хотите зашифровать, и прибавьте к ней номер ключа. Результирующая сумма и есть число, находящееся под зашифрованной буквой. Зашифруем, например, текст "HELLO. HOW ARE YOU?" с помощью ключа 13. (В качестве ключа можно использовать любое другое число от 1 до 25.) Прежде всего найдите число под буквой 'H', т.е. 7. Затем прибавьте 13 к этому числу: $7 + 13 = 20$. Поскольку число 20 находится под буквой 'U', то буква 'H' превращается в букву 'U'.

Точно так же, чтобы зашифровать букву 'E' (4), выполните сложение $4 + 13 = 17$. Буква над 17 — 'R', поэтому 'E' преобразуется в 'R' и т.д.

Этот процесс отлично работает до тех пор, пока мы не достигнем буквы 'O'. Число под 'O' — 14. Но $14 + 13 = 27$, а список чисел доходит лишь до значения 25. Если сумма чисел буквы и ключа равна или превышает 26, придется вычесть из нее 26. В данном случае $27 - 26 = 1$. Буква над цифрой 1 — 'B', поэтому, если используется ключ 13, буква 'O' превращается в 'B'. Зашифровав подобным образом каждую букву сообщения, получим зашифрованный текст "URYUW. UBJ NER LBH?".

Чтобы расшифровать зашифрованный текст, следует вместо прибавления ключа к каждой букве использовать вычитание. Букве 'B' в зашифрованном тексте соответствует число 1. Результатом вычитания 13 из 1 будет отрицательное число -12 . Аналогично нашему правилу "вычитания 26", если результат меньше нуля при дешифровании, то к нему нужно прибавить 26. Поскольку $-12 + 26 = 14$, буква 'B' в зашифрованном тексте дешифруется в букву 'O'.

Как видите, чтобы воспользоваться шифром Цезаря, вовсе не обязательно иметь шифровальный диск. Все, что нам для этого нужно, — карандаш, бумага и минимальные знания арифметики!

Почему не работает двойное шифрование

У кого-то из читателей может возникнуть мысль, что использование двух разных ключей шифрования подряд позволит вдвое увеличить стойкость шифра. Однако в случае шифра Цезаря (и большинства других шифров) это не так. В действительности результат двойного шифрования будет эквивалентен тому, который вы могли бы получить обычным способом с помощью единственного ключа. Попробуем применить двойное шифрование, чтобы понять, почему так происходит.

Если вы зашифруете слово “KITTEN” с помощью ключа 3, то будете прибавлять 3 к кодам букв простого текста, и результирующим текстом будет “NLWWHQ”. Если после этого зашифровать слово “NLWWHQ”, только теперь с ключом 4, то получим слово “RPAALU”. Но аналогичного результата можно достичь, если сразу зашифровать слово “KITTEN”, используя ключ 7.

Для большинства шифров многократное повторное шифрование не приводит к повышению стойкости шифра. Более того, если зашифровать простой текст с использованием двух ключей, сумма которых равна 26, то получим зашифрованный текст, полностью совпадающий с исходным!

Резюме

Шифр Цезаря и другие подобные ему шифры не одно столетие применялись для шифрования секретной информации. Но если вы захотите вручную зашифровать длинное сообщение, например целую книгу, то на это у вас может уйти несколько дней или недель. Тут нам на помощь и приходит программирование. С шифрованием и дешифрованием больших объемов текста компьютер способен справиться менее чем за секунду!

Чтобы использовать компьютер для шифрования информации, следует освоить понятный ему язык и научиться писать программы, т.е. наборы инструкций, следуя которым компьютер будет делать то же самое, что сделали бы мы. К счастью, изучить язык программирования наподобие Python гораздо легче, чем, скажем, японский или испанский. Что касается знания математики, то от вас потребуются лишь умение выполнять операции сложения, вычитания и умножения.

В следующей главе вы узнаете о том, как применять интерактивную оболочку Python для построчного изучения программного кода.

Контрольные вопросы

Ответы на контрольные вопросы приведены в приложении Б.

1. Зашифруйте следующие фразы из книги Амброза Бирса "Словарь Сатаны" ("The Devil's Dictionary"), используя указанные ключи.
 - A. "AMBIDEXTROUS: ABLE TO PICK WITH EQUAL SKILL A RIGHT-HAND POCKET OR A LEFT." (ключ 4).
 - B. "GUILLotine: A MACHINE WHICH MAKES A FRENCHMAN SHRUG HIS SHOULDERS WITH GOOD REASON." (ключ 17).
 - B. "IMPIETY: YOUR IRREVERENCE TOWARD MY DEITY." (ключ 21).
2. Дешифруйте следующие зашифрованные фрагменты текста, используя указанные ключи.
 - A. "ZXAI: P RDHIJBT HDBTIXBTH LDGCQN HRDIRWBTC XC PBTGXRP PCS PBTGXRPCHXC HRDIAPCS." (ключ 15).
 - B. "MQTSWXSU: E VMZEP EWTMVERX XSTYFPMG LRSVW." (ключ 4).
3. Зашифруйте следующее предложение, используя ключ 0:
"THIS IS A SILLY EXAMPLE."
4. Ниже приведены пары исходных слов и их зашифрованных версий. Какие ключи использовались в каждом случае?
 - A. ROSEBUD — LIMYVOX
 - B. YAMAMOTO — PRDRDFKF
 - B. ASTRONOMY — HZAYVUVTF
5. Как будет выглядеть приведенное ниже предложение, зашифрованное с помощью ключа 8, если дешифровать его с помощью ключа 9?
"UMMSVMAA: CVKWUUVV XIBQMVKM QV XTIVVQVO I ZMDMVOMBPIB QA EWZBP EPQTM."