

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | «КБИС» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | Д.А. Короченцев |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. | |

**ОТЧЕТ**

по учебной, ознакомительной практике

в АО «Алюминий Металлург Рус»

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Р.А. Карпов

подпись, дата

Обозначение отчета УП.11000.000 Группа ВКБ12

Направление 10.05.01 Информатика и вычислительная техника

Направленность (профиль) Математические методы защиты информации

Руководитель практики:

от предприятия заместитель генерального

директора по информационной безопасности   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.А. Кириченко

подпись, дата

от кафедры         ст. преподаватель                  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Э.Р. Типаева

подпись, дата

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата подпись преподавателя

Ростов-на-Дону

2023 г.



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

**ЗАДАНИЕ**

по учебной, ознакомительной практике

в АО «Алюминий Металлург Рус»

в период с «5» июня 2023 г. по «1» июля 2023 г.

Обучающийся Карпов Роман Александрович

Обозначение отчета УП.11000.000 Группа ВКБ12

Срок представления отчета на кафедру «\_\_» \_\_\_\_\_\_\_\_ 2023 г.

Содержание индивидуального задания

Реализация криптографического алгоритма DES. Создание скрипта, реализующего алгоритм DES.

|  |  |  |
| --- | --- | --- |
| Руководитель практики от  кафедры | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | Э.Р. Типаева |
| Руководитель практики от  предприятия | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | С.А. Кириченко |
| Задание принял к исполнению | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | Р.А. Карпов |



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника

Кафедра «Кибербезопасность информационных систем»

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | «КБИС» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | Д.А. Короченцев |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. | |

**Рабочий график (план) проведения практики**

|  |  |  |
| --- | --- | --- |
| **№** | **Мероприятие** | **Срок выполнения** |
| 1 | Прохождение вводного и первичного инструктажа по охране труда на рабочем месте, и инструктажа по пожарной безопасности на объекте. | 02.06.2023 |
| 2 | Получение индивидуального задания. | 05.06.2023 |
| 3 | Ознакомление с теоретическим материалом. | 06.06.2023 – 12.06.2023 |
| 4 | Разработка криптографического алгоритма DES | 13.06.2023 – 20.06.2023 |
| 5 | Окончательная проверка работоспособности алгоритма | 21.06.2023 |
| 6 | Оформление отчёта по практике. | 22.06.2023 – 24.06.2023 |
| 7 | Защита отчёта по практике на предприятии. | 26.06.2023 |
| 8 | Защита отчёта по практике на кафедре. | 27.06.2023 |

Руководитель практики:

от предприятия заместитель генерального

директора по информационной безопасности \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.А. Кириченко

подпись, дата

от кафедры   старший преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Э.Р. Типаева

подпись, дата

Ростов-на-Дону

2023 г.

ДНЕВНИК ПРОХОЖДЕНИЯ ПРАКТИКИ

В данном разделе ежедневно, кратко и четко записываются выполняемые работы, и в конце каждой недели журнал представляется для проверки руководителю (от предприятия и университета) практики. При выполнении одной и той же работы несколько дней, в графе «дата» сделать запись «с \_\_\_по\_\_\_».

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Место работы | Выполняемые работы | Оценка руководителя |
| 02.06.2023 | ДГТУ | Прохождение вводного и  первичного инструктажа по охране  труда на рабочем месте, и  инструктажа по пожарной  безопасности на объекте. |  |
| 05.06.2023 | АО «Алюминий Металлург Рус» | Ознакомление с территорией предприятия, прохождение первичного инструктажа по ТБ, ПБ |  |
| 05.06.2023 | АО «Алюминий Металлург Рус» | Получение индивидуального задания. |  |
| с 06.06.2023 по 12.06.2023 | АО «Алюминий Металлург Рус» | Ознакомление с теоретическим материалом |  |
| с 12.06.2023 по 15.06.2023 | АО «Алюминий Металлург Рус» | Создание основного алгоритма генерации ключей и шифрования DES |  |
| с 16.06.2023 по 17.06.2023 | АО «Алюминий Металлург Рус» | Реализация алгоритма дешифрования DES |  |
| 18.06.2023 | АО «Алюминий Металлург Рус» | Поиск и исправление сбоев программы |  |
| с 19.06.2023 по 20.06.2023 | АО «Алюминий Металлург Рус» | Реализация работы алгоритма с кодировкой UTF-8 для удобства пользователей |  |
| 21.06.2023 | АО «Алюминий Металлург Рус» | Окончательная проверка работоспособности алгоритма |  |
| с 22.06.2023 по 24.06.2023 | АО «Алюминий Металлург Рус» | Оформление отчёта по практике. |  |
| 26.06.2023 | АО «Алюминий Металлург Рус» | Защита отчёта по практике на предприятии. |  |
| 27.06.2023 | каф. «КБИС» | Защита отчёта по практике на кафедре. |  |

от предприятия  заместитель генерального

директора по информационной безопасности    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.А. Кириченко

подпись, дата

от кафедры   ст. преподаватель   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Э.Р. Типаева

подпись, дата

**Содержание**

Введение 6

1 Алгоритм DES: основа работы, генерация ключей, схема шифрования и дешифрования, шифрование более одного блока 10

1.1 Основа работы алгоритма DES 10

1.1.1 Сеть Фейстеля 10

1.1.2 Простое описание сети Фейстеля 11

1.2 Генерация ключей 12

1.3 Схема шифрования и дешифрования 14

1.4 Шифрование более одного блока 20

2 Криптоанализ шифра DES 21

2.1 Лавинный эффект 21

2.2 Проблемы ключей DES 22

2.3 Линейный и дифференциальный криптоанализ 23

3 Программная реализация алгоритма шифрования DES на языке программирования Python 25

3.1 Реализация функции get\_pass для получения ключей 25

3.2 Реализация функции encrypted для шифрования или дешифрования текста 26

Заключение 27

Перечень используемых источников 29

ПРИЛОЖЕНИЕ А 30

ПРИЛОЖЕНИЕ Б 34

Введение

Учебная практика является составной частью учебного процесса подготовки. Во время практики происходит закрепление и конкретизация результатов теоретического обучения, приобретение студентами умения и навыков практической работы по избранной специальности.

Место прохождения практики: АО «Алюминий Металлург Рус».

История Белокалитвинсого металлургического завода началась в 1939 году, в те годы по всей стране шло большое строительство предприятий, которые смогли бы обеспечить металлом новую авиационную отрасль промышленности.

В 1941 году строительство нового завода было остановлено из-за начавшейся Великой Отечественной войны.

3 мая 1948 года по решению правительства СССР вновь приступили к возведению Белокалитвинского металлургического завода. Одновременно со строительством корпусов заводских зданий шли активные работы по созданию города Белая Калитва: строились жилые дома, школы, больничный городок, Дворец культуры и другие объекты.

6 июля 1953 года Государственная комиссия приняла от строителей первый промышленный объект: блок ремонтных цехов.

5 августа 1954 года коллектив трубопрессового цеха отпрессовал первую продукцию из слитков, полученных с одного из ленинградских заводов.

12 октября 1954 года литейный цех объединения выполнил первое литье, и, по существующей традиции, этот день считается днем рождения предприятия.

Среди крупных объектов, вступивших в строй и действующих в последующие годы, были два прокатных, прутково-профильный, кузнечно-прессовый цеха, цех по производству товаров народного потребления и другие объекты.

В 1959 году впервые в мире была освоена серийная отливка круглых слитков в электромагнитный кристаллизатор, в 1970 году - плоских слитков. Это изобретение было запатентовано в 17 странах мира.

В 1969 году, впервые в отечественной практике, для получения металла повышенной чистоты были использованы усовершенствованные работниками объединения индукционные канальные печи.

В 1973 году в трубном производстве был смонтирован первый в Европе стан бухтового волочения ТБ-2800.

В 1977 году прокатный цех освоил производство особо крупногабаритной и длинномерной продукции.

БКМЗ остается единственным поставщиком спортивных копий и другого спортивного инвентаря на рынки СНГ. Спортивная продукция соответствует требованиям конгресса ИААФ.

С 1980 года на заводе впервые в стране стали выпускать, наряду с алюминиевой, и тефлонированную (с антипригарным покрытием) посуду.

С середины 60-х годов предприятие стало активно строить объекты соцкультбыта и жилья, так как имеющаяся в городе стройбаза не выполняла возросшие потребности. В 1976 году был создан свой строительный цех. Его создание, а затем и развитие строй индустрии позволило предприятию строить жилье, объекты соцкультбыта хозяйственным способом.

На протяжении всех лет своего существования БКМЗ было градообразующим предприятием. Им было построено большое количество объектов: поликлиника на 150 мест, больница на 124 койко-места, шахматный клуб, санаторий-профилакторий "Каяла", 10 детских садов, база отдыха "Долина Роз" в Крыму, гостиница. "Интурист", киноконцертный зал на 1100 мест, Дворец спорта, Молодежный парк, Дворец бракосочетаний и другое. Жилой фонд предприятия насчитывал более 360 000 кв. м жилья.

За свою историю БКМПО (ныне – АО «Алюминий Металлург Рус») имело многократные правительственные награды, а также областные и городские.

Достаточно только назвать такие даты, как, 1955 год - в этом году завод в первый раз получил переходящее Красное Знамя за трудовые успехи, в 1964 году коллективу БКМЗ было присвоено звание «Предприятие Коммунистического труда». А в 1981 году «За успешно проведенную реконструкцию и ввод в действие дополнительных мощностей» Президиум ВС СССР наградил предприятие орденом Трудового Красного Знамени.

Около 350 работников объединения в разные годы были награждены орденами и медалями за трудовые успехи, а более 130 человек - золотыми, серебряными и бронзовыми медалями ВДНХ (ныне - ВВЦ). БКМЗ гордится своими героями: Быковым Б. И. - Героем Советского Союза и Бермасом В. К. - Героем Социалистического Труда.

На предприятия успешно трудились и продолжают работать 29 человек, награжденных одними из самых высоких наград России - лауреаты Государственной премии и премии Совета Министров. Причем, два человека - бывший Генеральный директор Сафаров Г. С. и главный металлург Березин Л. Г.- имеют оба эти звания.

Семь человек в объединении стали кандидатами технических наук.

Тринадцать работников объединения имеют почетное звание «Заслуженный рационализатор России» и один – «Заслуженный изобретатель России».

На сегодняшний день АО «АМР» является одним из ведущих многопрофильных предприятий России, специализируется на выпуске полуфабрикатов и готовых изделий, алюминия и его сплавов. Изделия предприятия поставлялись для отделки зданий Кремлевского Дворца, Останкинской телевизионной башни, павильонов Всероссийского выставочного центра в городе Москва.

Предприятие поставляет свою продукцию более чем в 20 стран мира. Изделия, выпускаемые объединением, отличает повышенная чистота сплавов точность геометрических размеров, выполнение требований потребителя по уровни механических свойств и ресурсных характеристик материала.

Офис находится в г. Белая Калитва, ул. Заводская, д. 1. На предприятии от университета было 3 студента, которые также проходили практику. В первый день провели вводный инструктаж для студентов, проходящих практику, рассказали о ПБ и ТБ. Каждому студенту было выдано своё индивидуальное задание.

1. Алгоритм DES: основа работы, генерация ключей, схема шифрования и дешифрования, шифрование более одного блока
   1. Основа работы алгоритма DES

В основе алгоритма лежит [сеть Фейстеля](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D1%8C_%D0%A4%D0%B5%D0%B9%D1%81%D1%82%D0%B5%D0%BB%D1%8F) с 16 циклами ([раундами](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%83%D0%BD%D0%B4_(%D0%B2_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%B8))) и [ключом](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)), имеющим длину 56 [бит](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82). Алгоритм использует комбинацию нелинейных (S-блоки) и линейных (перестановки E, IP, IP-1) преобразований.

1.1.1 **Сеть Фейстеля**

Сеть Фейстеля, или конструкция [Фейстеля](https://ru.wikipedia.org/wiki/%D0%A4%D0%B5%D0%B9%D1%81%D1%82%D0%B5%D0%BB%D1%8C,_%D0%A5%D0%BE%D1%80%D1%81%D1%82" \o "Фейстель, Хорст) — один из методов построения [блочных шифров](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80). Сеть состоит из ячеек, называемых ячейками Фейстеля. На вход каждой ячейки поступают данные и ключ. На выходе каждой ячейки получают изменённые данные и изменённый ключ. Все ячейки однотипны, и говорят, что сеть представляет собой определённую многократно повторяющуюся ([итерированную](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))) структуру. [Ключ](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)) выбирается в зависимости от алгоритма шифрования/расшифрования и меняется при переходе от одной ячейки к другой. При [шифровании](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и расшифровании выполняются одни и те же операции; отличается только порядок [ключей](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)).

Ввиду простоты операций сеть Фейстеля легко реализовать как программно, так и аппаратно.

Ряд [блочных шифров](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80) ([DES](https://ru.wikipedia.org/wiki/DES), [RC2](https://ru.wikipedia.org/wiki/RC2), [RC5](https://ru.wikipedia.org/wiki/RC5), [RC6](https://ru.wikipedia.org/wiki/RC6), [Blowfish](https://ru.wikipedia.org/wiki/Blowfish), [FEAL](https://ru.wikipedia.org/wiki/FEAL), [CAST-128](https://ru.wikipedia.org/wiki/CAST-128), [TEA](https://ru.wikipedia.org/wiki/TEA), [XTEA](https://ru.wikipedia.org/wiki/XTEA), [XXTEA](https://ru.wikipedia.org/wiki/XXTEA) и др.) использует сеть Фейстеля в качестве основы. Альтернативой сети Фейстеля является [подстановочно-перестановочная сеть](https://ru.wikipedia.org/wiki/SP-%D1%81%D0%B5%D1%82%D1%8C) ([AES](https://ru.wikipedia.org/wiki/Advanced_Encryption_Standard) и др.).

1.1.2 Простое описание сети Фейстеля

*Шифрование*

Пусть требуется зашифровать некоторую [информацию](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D1%8F), представленную [в двоичном виде](https://ru.wikipedia.org/wiki/%D0%95%D0%B4%D0%B8%D0%BD%D0%B8%D1%86%D1%8B_%D0%B8%D0%B7%D0%BC%D0%B5%D1%80%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8) (в виде последовательности [нулей и единиц](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F)) и находящуюся в [памяти](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C) компьютера или иного устройства (например, в [файле](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB)).

* Информация разбивается на блоки одинаковой (фиксированной) длины. Полученные блоки называются входными, так как поступают на вход алгоритма. В случае если длина входного блока меньше, чем размер, который выбранный алгоритм шифрования способен зашифровать единовременно (размер блока), то блок удлиняется каким-либо способом. Как правило, длина блока является [степенью](https://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%B7%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2_%D1%81%D1%82%D0%B5%D0%BF%D0%B5%D0%BD%D1%8C) двойки, например, составляет 64 бита или 128 бит.

Далее будем рассматривать операции, происходящие только с одним блоком, так как в процессе шифрования с другими блоками выполняются те же самые операции.

* Выбранный блок делится на два подблока одинакового размера — «левый» (�0L0) и «правый» (�0R0).
* «Правый подблок» �0R0 изменяется функцией F с использованием [раундового](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%83%D0%BD%D0%B4_(%D0%B2_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%B8)) ключа �0K0



* Результат [складывается по модулю 2 («xor»)](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8E_2) с «левым подблоком» �0L0:



* Результат будет использован в следующем [раунде](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%83%D0%BD%D0%B4_(%D0%B2_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%B8)) в роли «правого подблока» �1R1:



* «Правый подблок» �0R0 текущего раунда (в своем неизмененном на момент начала раунда виде) будет использован в следующем раунде в роли «левого подблока» �1L1:



* По какому-либо математическому правилу вычисляется [раундовый](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%83%D0%BD%D0%B4_(%D0%B2_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%B8)) [ключ](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)) �1K1 — ключ, который будет использоваться в следующем раунде.

Перечисленные операции выполняются N-1 раз, где N — количество [раундов](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%83%D0%BD%D0%B4_(%D0%B2_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%B8)) в выбранном алгоритме шифрования. При этом между переходами от одного раунда (этапа) к другому изменяются ключи: �0K0 заменяется на �1K1, �1K1 — на �2K2 и т. д.).

#### Расшифрование

Расшифровка информации происходит так же, как и шифрование, с тем лишь исключением, что [ключи](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)) следуют в обратном порядке, то есть не от первого к N-му, а от N-го к первому.

* 1. Генерация ключей

DES создает 16 раундовых ключей ik по 48 битов из ключа k шифра на 56 битов. Однако, чтобы задать ключ шифра надо среди 56 битов ключа дополнительно вписать 8 битов в позиции 8,16,...,64 для проверки четности таким образом, чтобы каждый байт содержал нечетное число единиц. С помощью этой операции выявляют ошибки при обмене и хранении ключей. Ключевое расписание состоит из этапов:

* Перестановка сжатия для удаления битов проверки – из 64- битового ключа удаляют биты 8,16,24,32,…,64 и переставляет остальные биты согласно таблице (в ходе перестановки сохраняется нумерация битов расширенного ключа).



Рис. 1 – Таблица удаления битов в ключе

* После перестановки 56 битов ключа делятся на два блока C0 и D0 по 28 бит каждый. Далее для генерации раундовых ключей из блоков C0 и D0 с помощью операции циклического сдвига влево на 1-2 бита строятся блоки Ci и Di , i =1,2,...,16 . В раундах 1,2,9 и 16 смещение – на 1 бит, в других раундах — на 2 бита. После определения блоков Ci и Di биты этих блоков объединяются в один ключ на 56 битов.



Рисунок 2 – Таблица сдвига ключа

* Перестановка сжатия (P-бокс, таблица) изменяет 56 битов на 48 битов, которые образуют раундовый ключ.



Рисунок 3 – Последняя перестановка битов в ключе

При расшифровании – раундовые ключи используются те же, что и при зашифровании, но теперь они идут в обратном порядке

* 1. Схема шифрования и дешифрования

Исходный текст — блок 64 бит.

Процесс шифрования состоит из начальной перестановки, 16 циклов шифрования и конечной перестановки.

### *Начальная перестановка*

Исходный текст� (блок 64 бит) преобразуется c помощью начальной перестановки IPIP которая определяется таблицей 1:



Рисунок 4 – Таблица начальной перестановки IP

По таблице бит исходного текста под номером 58 становится первым, 50-ый бит – вторым и т.д.

*Циклы шифрования*

DES использует 16 раундов. Каждый раунд DES применяет шифр Фейстеля, как это показано на рисунке

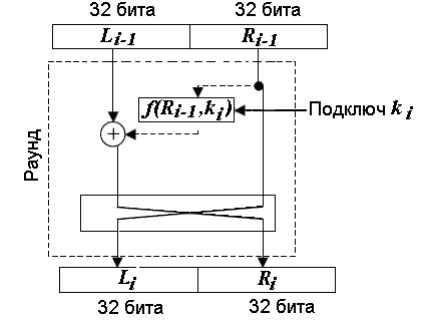


Рисунок 5 – Иллюстрация одного раунда шифрования

Раунд принимает полублоки Li−1 и Ri−1 от предыдущего раунда (или начального блока перестановки) и создает полублоки Li и Ri для входа в следующий раунд (или конечный блок перестановки). Все необратимые элементы сосредоточены в функции f(Ri-1, ki).

*Функция DES*

Функция DES с помощью 48-битового ключа зашифровывает 32 самых правых бит Ri−1, чтобы получить на выходе 32-битовый результат. Эта функция содержит, как это показано на 4 составляющие: операция XOR,

P-бокс расширения, группу S -боксов и прямой P-бокс.

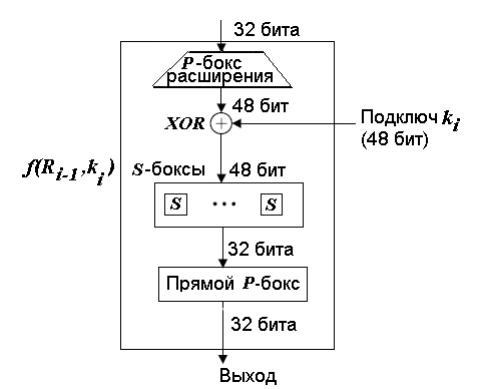


Рисунок 6 – Составляющие алгоритма DES и результат их работы (в кол-ве битов)

P-бокс расширения служит для расширения 32-битового блока Ri−1 до 48 битов, чтобы согласовать его размеров с размерами подключа раунда. Блок Ri−1 делится на 8 секций по 4 бита. Каждая секция расширяется до 6 бит.

(Для секции значения входных битов в позициях 1, 2, 3 и 4 присваиваются битам в позициях 2, 3, 4 и 5 соответственно на выходе. 1-ый выходной бит равен входному 4-му биту предыдущей секции; 6-ой бит выхода равен 1-му биту следующей секции. Если секции 1 и 8 рассматривать как соседние секции, то те же самые правила применяются к битам 1 и 32).

Хотя отношения между входом и выходом могут быть определены математически, P-бокс задают таблицей.



Рисунок 7 – Таблица расширения Ri до 48 бит

(число выходов 48, диапазон значений – от 1 до 32. Некоторые входные биты порождают несколько выходных).

После расширения DES использует операцию XOR над расширенной частью правого полублока Ri−1 и ключом раунда ki.

После суммирования с битами ключа блок из 48 битов делится на 8 последовательных 6-битових векторов b1 ,b2 ,...,b8 , каждый из которых заменяется на 4-битовий вектор b′j с помощью S-боксов.

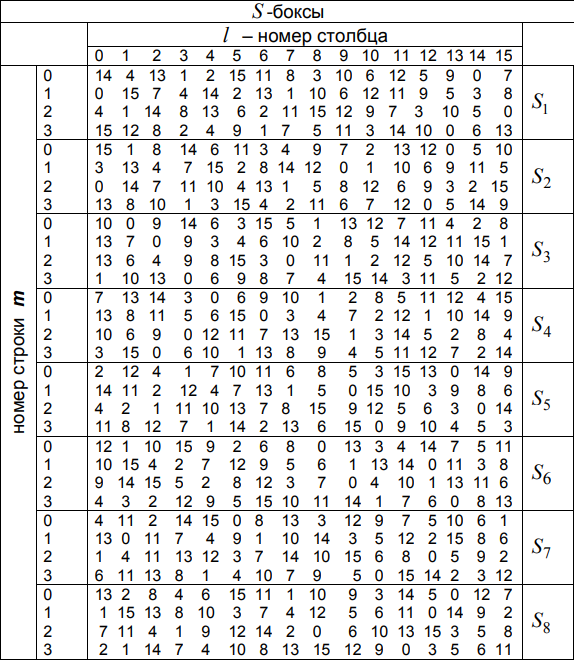


Рисунок 8 – Таблица S-боексов для формирования новых векторов

І вектор b1 попадает в бокс S1, ІІ вектор b2 – в бокс S2. S -бокс – это таблица из 4 строк и 16-ти столбцов. Чтобы в S-боксе найти шифрообозначения вектора, надо:

* Из 1-го и последнего битов вектора образовать двоичное число, перевести его в десятичную систему. Это будет номер строки m (m = 0,1,2,3);
* Из 2-го, 3-го, 4-го и 5-го бита вектора образовать двоичное число, перевести его в десятичную систему. Это будет номер столбца l (l = 0,...,15);
* Число, стоящее в S -боксе на пересечении m-ой строки и l-го столбца, будет шифрообозначение b′j вектора bj;
* Шифрообозначение b′j перевести в двоичную систему.

Таким образом, после S-боксов мы получаем восемь 4-битовых векторов b′1, b′2,...,b′8, которые опять объединяют в 32-битовый блок. Далее биты блока перетасовываются в прямом P-боксе на основе заданной таблицы (правила пользования таблицей перестановки старые: например, 7-ой бит входа станет 2-ым битом выхода).



Рисунок 9 – Перестановка бит в Ri

После перестановки результат функции проходит через XOR с Li-1

После 16-го раунда DES правый и левый блоки уже не меняются местами, а объединяются в блок R16L16 и подвергаются финальной перестановке IP-1.



Рисунок 10 – Конечная перестановка бит в R16L16

При расшифровании данных все действия выполняются в обратном порядке. В 16 циклах расшифрования, в отличие от шифрования c помощью прямого преобразования [сетью Фейстеля](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D1%8C_%D0%A4%D0%B5%D0%B9%D1%81%D1%82%D0%B5%D0%BB%D1%8F), здесь используется обратное преобразование сетью Фейстеля.



Ключ ��Ki, i=16,…,1, функция f, перестановка IP и ��−1IP-1 такие же, как и в процессе шифрования. Алгоритм генерации ключей зависит только от ключа пользователя, поэтому при расшифровании они идентичны.

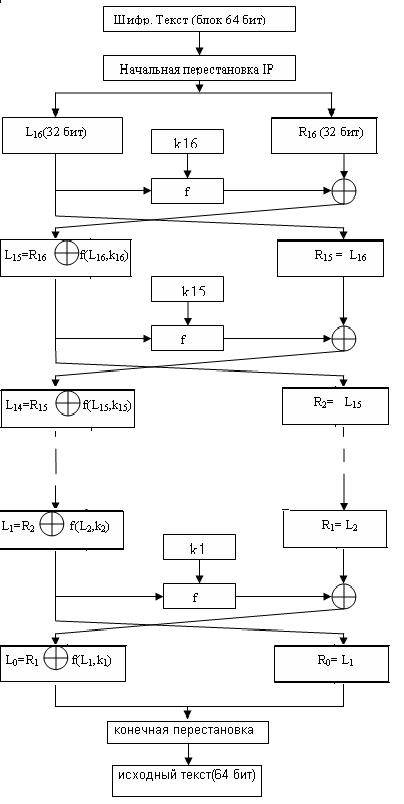


Рисунок 11 – Иллюстрация алгоритма дешифрования

* 1. Шифрование более одного блока

В [1981 году](https://ru.wikipedia.org/wiki/1981_%D0%B3%D0%BE%D0%B4) был принят стандарт [FIPS 81](https://web.archive.org/web/20090604131329/http:/www.itl.nist.gov/fipspubs/fip81.htm). В стандарте были описаны первые режимы работы блочных шифров: ECB, CBC, OFB и CFB. В [2001 году](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) институт [NIST](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BE%D0%B2_%D0%B8_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B9) (национальный институт стандартов и технологий [США](https://ru.wikipedia.org/wiki/%D0%A1%D0%A8%D0%90)) пересмотрел список режимов и добавил в него описание работы [блочного шифра](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80) [AES](https://ru.wikipedia.org/wiki/Advanced_Encryption_Standard) в режиме CTR ([SP800-38A](http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf)). В январе [2010 года](https://ru.wikipedia.org/wiki/2010_%D0%B3%D0%BE%D0%B4) NIST добавил в стандарт описание работы шифра [AES](https://ru.wikipedia.org/wiki/Advanced_Encryption_Standard) в режиме XTS (SP800-38E).

В стандарте описаны не все режимы, а только режимы, одобренные институтом [NIST](https://ru.wikipedia.org/wiki/NIST). Например, режим [CTS](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B8%D0%BC%D1%81%D1%82%D0%B2%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D1%88%D0%B8%D1%84%D1%80%D0%BE%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%B0) в стандарте не описан, но реализован во многих популярных криптографических [библиотеках](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)).

Режимы шифрования определяются рядом национально и международно признанных организаций. Наиболее влиятельной из них является [NIST](https://ru.wikipedia.org/wiki/NIST).

1. Криптоанализ шифра DES
   1. Лавинный эффект

*Лавинный эффект* – проявление зависимости всех выходных битов шифротекста от каждого входного бита открытого текста (в криптографии такой анализ проводят для блочных шифров и хэшфункций). Лавинный эффект проявляется в зависимости всех выходных битов от каждого входного бита. Термин введен Фейстелем, хотя концептуальное понятие использовалось еще Шенноном. Если криптографический алгоритм не обладает лавинным эффектом в достаточной степени, противник может сделать предположение о входной информации, основываясь на выходной информации. Таким образом, достижение лавинного эффекта является важной целью при разработке криптографического алгоритма. Криптоалгоритм удовлетворяет лавинному критерию, если при изменении одного бита на входе алгоритма изменяется в среднем половина битов на выходе алгоритма. Если же при изменении одного бита на входе каждый бит на выходе изменяется с вероятностью ½, то криптоалгоритм удовлетворяет строгому лавинному критерию. В DES лавинный эффект проявляется уже на 4-5 раунде. Так, если зашифровать на одном ключе с помощью DES, 2 блока открытого текста, отличающиеся одним битом, то блоки шифротекстов будут отличаться на 29 бит, т.е. изменение открытого текста на 1,5% вызывает 45% изменений шифротекста. Изменение битов по раундам:

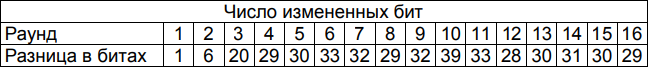


Рисунок 12 – Таблица количества изменения бит в каждом раунде

Доказано, что после того как текст зашифрован в 8 раундах, каждый бит шифрованного текста — функция каждого бита открытого текста и ключа.

Но эксперименты показывают, что DES с менее чем 16 раундами более уязвимы к атакам на основе открытого текста, чем к атаке грубой силы, требующей использования 16 раундов.

* 1. Проблемы ключей DES

Самая серьезная проблема DES –размер ключа (56 битов). Чтобы предпринять атаку грубой силы, надо проверить ключей.

Если проверять 1.000.000 ключей/сек., то потребуется ≈ 2000 лет, чтобы выполнить атаку грубой силы на DES на одном процессоре.

Если сделать компьютер с миллионом крипточипов, то все множество ключей проверится за 20 часов. Когда был введен DES, стоимость такого компьютера была немного более миллиона долларов, но она быстро снизилась. Такой специальный компьютер был создан в 1998 году и нашел ключ за 56 часов. С помощью компьютерных сетей моделируют параллельный поиск ключей, что также ускоряет вскрытие. Ключ 56 битов не обеспечивает достаточной безопасности. (решение проблемы – использование тройного DES(3DES) с двумя ключами (112 битов) или тройного DES с тремя ключами).

Ключевое расписание ключей для раундов допускает слабые ключи. При генерировании раундовых ключей 56-битный ключ шифра делится на две половины и каждая из них сдвигается независимо. Если ключ шифра состоит только из 0 или 1 или, если одна его половина из 0, а другая – из 1, то в этом случае раундовые ключи оказываются попарно одинаковыми, т.е. K1 = K16, K2 = K15 и т.д., и процедуры шифрования/дешифрования оказываются идентичными.

Формально слабым ключом DES называется такой 56-битный ключ K, при котором EK(EK(X)) = X , где X – 64-битный блок открытого текста. Среди всех возможных ключей DES имеется 4 слабых.

Для алгоритма DES кроме слабых ключей существуют еще полуслабые ключи – это пара 56-битных ключей K1, K2, при которой DESK1(DESK2(X)) = X. Полуслабые ключи дают одинаковый результат при шифровании текстов. Это тоже связано с генерацией раундовых ключей – вместо 16 различных ключей генерируются только 2 различных, которые затем используются 8 раз в алгоритмах. Таких ключей у DES 6 пар.

Оценим вероятность случайного выбора слабого, полуслабого или возможно слабого ключа. Общее число вышеупомянутых ключей: 4 + 12 + 48 = 64. Вероятность выбора одного из этих ключей исключительно мала: . Поэтому слабость ключей не является существенным недостатком, поскольку в программной или аппаратной реализации DES достаточно просто запретить использование проблемных ключей.

2.3 **Линейный и дифференциальный криптоанализ**

С самого начала использования алгоритма криптоаналитики всего мира прилагали множество усилий для взлома DES. Фактически DES дал невиданный доселе толчок развитию криптоанализа. Вышли сотни трудов, посвященных различным методам криптоанализа именно в приложении к алгоритму DES, а также деталям самого алгоритма и их влиянию на криптостойкость. Можно утверждать, что именно благодаря DES появились целые направления криптоанализа, такие как:

- *линейный криптоанализ* – анализ зависимостей между открытым текстом и шифротекстом;

- *дифференциальный криптоанализ* – анализ зависимостей между соотношениями двух или более открытых текстов и соответствующих им шифротекстов;

- *криптоанализ на связанных ключах* – поиск и анализ зависимостей между шифротекстами, полученными на искомом ключе и ключах, связанных предполагаемым соотношением с искомым ключом; однако DES оказался неуязвим к данному виду атак, так как по ключевому расписанию циклический сдвиг битов ключа выполняется на различное число позиций в разных раундах. DES стойко выдержал 20 лет массового всемирного криптоанализа – десятилетия криптоанализа не привели к обнаружению серьезных 14 уязвимостей в алгоритме. Основными результатами усилий по взлому DES можно считать следующие:

* Японский специалист Мицуру Мацуи изобретатель линейного криптоанализа, в 1993 году показал, что вычислить ключ шифра можно методом линейного криптоанализа при наличии у атакующего пар «открытый текст – шифротекст».
* Криптологи из Израиля, изобретатели дифференциального криптоанализа Эли Бихам и Ади Шамирв 1991 году представили атаку, в которой ключ шифрования вычислялся методом дифференциального криптоанализа при условии, что атакующий имеет специально выбранных пар «открытый текст – шифротекст».

В дальнейшем эти атаки были несколько усилены (например, атака линейным криптоанализом при наличии пар вместо ), появлялись также новые виды атак на DES (например, атака, позволяющая вычислить ключ высокоточным облучением аппаратного шифратора и последующим анализом ошибок шифрования). Однако, все эти атаки требуют наличия огромного количества пар «открытый текст – шифротекст», получение которых на практике является настолько трудоемкой операцией, что наиболее простой атакой на DES все еще можно считать полный перебор ключей.

1. Программная реализация алгоритма шифрования DES на языке программирования Python
   1. Реализация функции get\_pass для получения ключей

В исходной функции main, после ввода пароль он переводится в HEX, а после в BIN формат. Ключ проходит первую перестановку через заранее заготовленную таблицу и разделяется на две части.

На входе функция get\_pass получает разделённый на две части пароль, заранее переведённый в BIN формат. Создаётся пустой массив pwrd, который заполнится ключами для каждого раунда шифрования/дешифрования. Запускается цикл из 16 раундов, в каждом из который левая и правая часть смещается на n бит влево (зависит от раунда), после чего происходит конкатенация двух частей пароля и вторая перестановка бит в ключе. Ключ делится на части по 4 бита и каждый блок переводится в HEX символ, конкатенируется и добавляется в массив pwrd.

После прохождения всех 16 раундов и записи в массив всех ключей функция возвращает массив ключей.

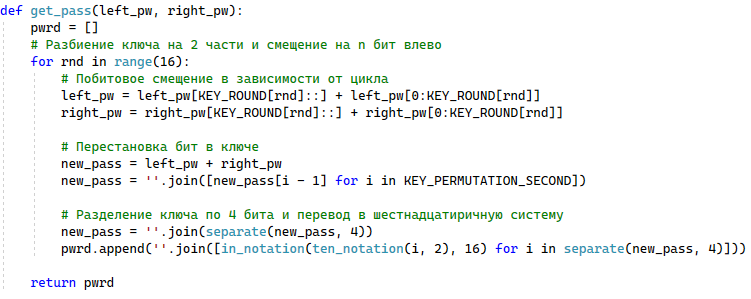


Рисунок 13 – Реализации функции get\_pass

* 1. Реализация функции encrypted для шифрования или дешифрования текста

Функция main получается на вход текст, который переводит на HEX формат и разбивает на блоки по 16 символов. Дальше цикл проходит по каждому блоку и отправляет его в функцию encrypted

Функция encrypted получается на вход два аргумента: HEX блок текста и массив паролей из пункта 3.1. Далее в функции encrypted текст переходит в BIN формат, проходит первую перестановку и разделяется на две части (R0 и L0 по 32 бита). Запускается цикл в 16 раундов, в начале правая часть снова проходит через перестановку и складывается по модулю два с двоичной записью ключа i-го раунда. Полученная запись разделяется на 8 блоков по 6 бит, проходит по S-боксу и формируется новый S-блок. Далее этот S-блок проходит через ещё одну перестановку и складывается по модулю два с левой частью (L0). Если i ≠ 15, то L0 и R0 меняются местами.

После прохождения 16-ти раундов L0 конкатенируется в R0, проходит через перестановку IP-1 и возвращается в виде HEX блоков по 16 символов

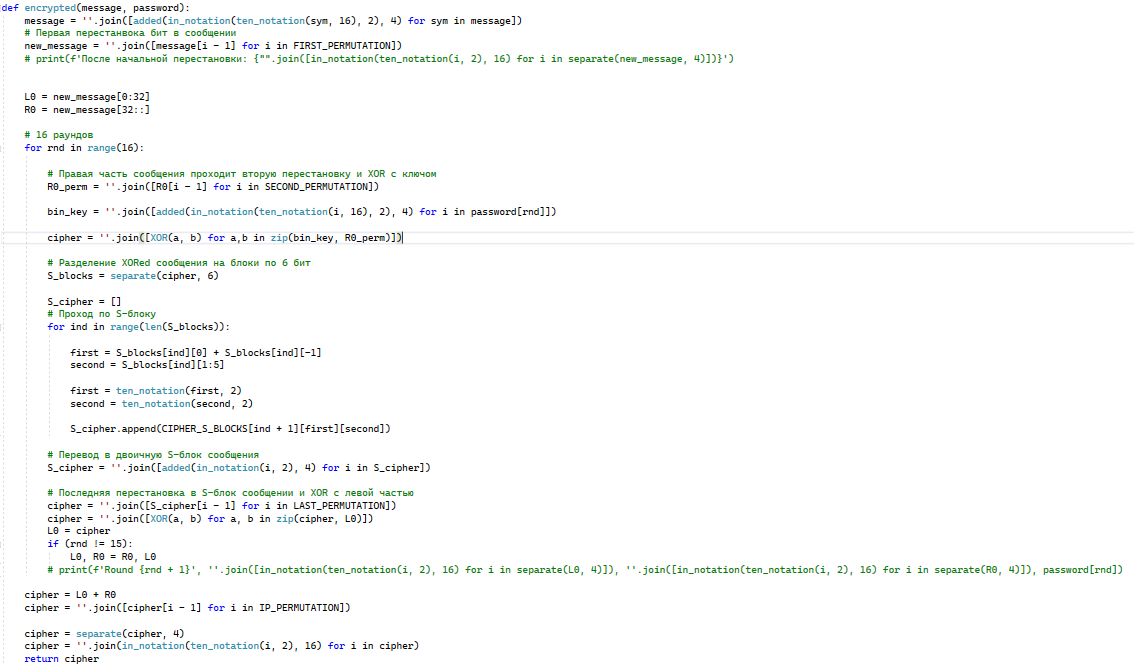


Рисунок 14 – Реализация функции encrypted

Заключение

Стандарт шифрования данных (DES) — блочный шифр с симметричными ключами, изданный национальным Институтом Стандартов и Технологии (NIST) как FIPS 46 в Федеральном Регистре.

На стороне шифрования DES принимает исходный текст на 64 бита и создает зашифрованный текст на 64 бита. На стороне дешифрования DES принимает зашифрованный текст на 64 бита и создает блок на 64 бита исходного текста. Ключ шифра на 56 битов одного типа используется и для шифрования, и для дешифрования.

Процесс шифрования состоит из двух перестановок (P -блоки), которые называются начальными и конечными перестановками, и шестнадцати раундов Фейстеля. Каждый раунд DES — шифр Фейстеля с двумя элементами (смеситель и устройство замены). Каждый из этих элементов является обратимым.

Основой DES является функция DES. Функция DES применяет ключ на 48 битов к самым правым 32 битам, чтобы получить на выходе 32 бита. Эта функция составлена из четырех операций: перестановки расширения, отбеливателя (который добавляет ключ), группы S -блоков и прямой перестановки.

Генератор ключей раунда создает из ключа шифра на 56 битов шестнадцать ключей по 48 битов. Однако ключ шифра обычно представляется как ключ на 64 бита, в котором 8 дополнительных битов являются проверочными битами — они отбрасываются перед фактическим процессом генерирования ключей.

DES показывает хорошие рабочие характеристики по отношению к эффектам полноты (законченности) и лавины. К числу слабостей DES относятся: построение шифра (S -блоки и P -блоки) и ключ шифра (длина), слабые ключи, полуслабые ключи, возможно слабые ключи и дополнение ключей.

Так как DES — не группа, одно из решений по улучшению безопасности DES состоит в том, чтобы пользоваться многократными DES, которые используют кратное число применения ключей (двукратный или трехкратный DES).

Двукратный DES уязвим к атаке сведения к середине, поэтому в обычных приложениях используется трехкратный DES с двумя ключами или с тремя ключами.

Разработка S -блоков и число раундов сделали DES почти обладающим иммунитетом от дифференциального криптоанализа. Однако DES уязвим при применении линейного криптоанализа, если злоумышленники смогут собрать достаточно много исходных текстов.

Теоретически, шифртекст может быть расшифрован только при помощи ключа, которым он был зашифрован. DES использует 64-битный ключ, но 8 бит используется для проверки четности, так что эффективная длина составляет всего 56 бит.

Длина ключа всегда казалась небезопасной, а в 1998 Electronic Frontier Foundation (EFF, Фонд Электронных Рубежей) смог расшифровать сообщение, зашифрованное с помощью DES, за 56 часов.

Перечень используемых источников

* 1. https://encyclopedia.kaspersky.ru/glossary/des-data-encryption-standard/
  2. <https://heritage-offshore.com/informacionnoj-bezopasnosti/chto-takoe-shifrovanie-3des-i-kak-rabotaet-des/>
  3. <https://ru.wikipedia.org/wiki/DES>
  4. <https://ru.wikipedia.org/wiki/Triple_DES>
  5. <https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80#%D0%93%D0%9E%D0%A1%D0%A2_28147-89>
  6. <https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D1%8C_%D0%A4%D0%B5%D0%B9%D1%81%D1%82%D0%B5%D0%BB%D1%8F>
  7. <https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B6%D0%B8%D0%BC_%D1%88%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F>

ПРИЛОЖЕНИЕ А

**Техническое задание на программное средство**

|  |  |
| --- | --- |
| «СОГЛАСОВАННО» | «УТВЕРЖДЕНО» |
| Руководитель проекта | Зав. кафедрой «КБИС» |
| \_ / Э.Р. Типаева / | \_\_\_\_\_\_\_\_\_/ Д.А. Короченцев / |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023г. | «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023г. |

**А.1 Введение**

Наименование разрабатываемого программного средства:

“Программная реализация криптографического алгоритма DES”

Область применения:

* лабораторные работы по предмету алгоритмы и структуры данных;
* личное пользование приложением.

**А.2 Основания для разработки**

Разработка ведётся на основании документа “Положение о практике обучающихся, осваивающих основные профессиональные образовательные программы высшего образования”, утверждённого министерством образования и науки российской федерации 4.05.2018.

Предметным основанием является задание на учебную практику.

**А.3 Назначение разработки**

Функциональное назначение программного средства заключается в изучении криптографического алгоритма DES.

**А.4 Требование к программе или программному изделию**

**А.4.1 Требование к функциональным характеристикам**

Программа корректно использует криптографический алгоритм DES.

Состав выполняемых функций:

* консольный интерфейс взаимодействия с пользователем;

Входные данные:

* Режим работы (шифрование/дешифрование)
* Текст
* Пароль

Выходные данные:

* Зашифрованный/дешифрованный текст

Данные программой хранятся до следующего использования программы или выхода из нее. Здесь кэширование не применяется.

**А.4.2 Требование надежности**

Разрабатываемая программа должна удовлетворять следующим требования к надежности:

Отказ программы вследствие внутренних ошибок не допустим;

Отказ программы вследствие неверно введенных данных не допустим;

Отказ программы вследствие неверного клонирования не допустим;

**А.4.3 Условия эксплуатации**

Для функционирования программного продукта необходимо соблюдение всех требований и правил эксплуатации компьютерной техники. Программное средство обслуживается одним пользователем.

**А.4.4 Требование к составу и параметрам технических средств**

Состав технических средств:

* 13 Кб свободного места на диске;
* 32 или 64-разрядный процессор с частотой не ниже 1 ГГц;
* Монитор VGA или HDMI;
* Клавиатура и мышь;
* Оперативная память не меньше 256 Мб;
* Интерпретатор Python 3.10.

**А.4.5 Требования к информационной и программной совместимости**

Программное средство требует семейство операционных систем Windows или др.

**А.4.6 Требования к маркировке и упаковке**

Требования к маркировке не предъявляются. Требования к упаковке определяются требованиями к хранению носителей информации, содержащих ПО.

**А.4.7 Требования к транспортировке и хранению**

Требования к транспортировке и хранению определяются требованиями к транспортировке и хранению носителей информации, содержащих ПО. Данную программу можно скачать с репозитория «*GitHub».*

**А.5 Стадии и этапы разработки**

Изучение предметной области (от 06.06.23 до 12.06.23);

Разбор технического задания (от 08.06.23 до 11.06.23);

Анализ работы криптографического алгоритма DES (06.06.23 до 12.06.23);

Разработка криптографического алгоритма DES (от 13.06.23 до 20.06.23);

Исправление ошибок (21.06.2023);

Испытание программного средства (от 17.06.23 до 23.06.23).

**А.6 Порядок и контроль приемки**

Порядок и контроль приемки определяется руководством кафедры КБИС Донского государственного технического университета.

Разработчик технического задания: Карпов Роман Александрович

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2023г \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

ПРИЛОЖЕНИЕ Б

Листинг 1 – Вспомогательные таблицы перестановок:

# Побитовая перестановка текста

FIRST\_PERMUTATION = [

58, 50, 42, 34, 26, 18, 10, 2,

60, 52, 44, 36, 28, 20, 12, 4,

62, 54, 46, 38, 30, 22, 14, 6,

64, 56, 48, 40, 32, 24, 16, 8,

57, 49, 41, 33, 25, 17, 9, 1,

59, 51, 43, 35, 27, 19, 11, 3,

61, 53, 45, 37, 29, 21, 13, 5,

63, 55, 47, 39, 31, 23, 15, 7

]

# Сжатие ключа до 56 бит

KEY\_PERMUTATION\_FIRST = [

57, 49, 41, 33, 25, 17, 9,

1, 58, 50, 42, 34, 26, 18,

10, 2, 59, 51, 43, 35, 27,

19, 11, 3, 60, 52, 44, 36,

63, 55, 47, 39, 31, 23, 15,

7, 62, 54, 46, 38, 30, 22,

14, 6, 61, 53, 45, 37, 29,

21, 13, 5, 28, 20, 12, 4

]

# Сжатие ключа до 48 бит

KEY\_PERMUTATION\_SECOND = [

14, 17, 11, 24, 1, 5,

3, 28, 15, 6, 21, 10,

23, 19, 12, 4, 26, 8,

16, 7, 27, 20, 13, 2,

41, 52, 31, 37, 47, 55,

30, 40, 51, 45, 33, 48,

44, 49, 39, 56, 34, 53,

46, 42, 50, 36, 29, 32

]

# Расширение правой части с 32 до 48 бит

SECOND\_PERMUTATION = [

32, 1, 2, 3, 4, 5,

4, 5, 6, 7, 8, 9,

8, 9, 10, 11, 12, 13,

12, 13, 14, 15, 16, 17,

16, 17, 18, 19, 20, 21,

20, 21, 22, 23, 24, 25,

24, 25, 26, 27, 28, 29,

28, 29, 30, 31, 32, 1

]

LAST\_PERMUTATION = [

16, 7, 20, 21,

29, 12, 28, 17,

1, 15, 23, 26,

5, 18, 31, 10,

2, 8, 24, 14,

32, 27, 3, 9,

19, 13, 30, 6,

22, 11, 4, 25

]

IP\_PERMUTATION = [

40, 8, 48, 16, 56, 24, 64, 32,

39, 7, 47, 15, 55, 23, 63, 31,

38, 6, 46, 14, 54, 22, 62, 30,

37, 5, 45, 13, 53, 21, 61, 29,

36, 4, 44, 12, 52, 20, 60, 28,

35, 3, 43, 11, 51, 19, 59, 27,

34, 2, 42, 10, 50, 18, 58, 26,

33, 1, 41, 9, 49, 17, 57, 25,

]

CIPHER\_S\_BLOCKS = {

1: [

[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7],

[0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8],

[4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0],

[15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13]

],

2: [

[15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10],

[3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5],

[0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15],

[13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9]

],

3: [

[10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8],

[13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1],

[13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7],

[1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12]

],

4: [

[7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15],

[13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9],

[10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4],

[3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14]

],

5: [

[2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9],

[14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6],

[4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14],

[11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3]

],

6: [

[12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11],

[10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8],

[9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6],

[4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13]

],

7: [

[4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1],

[13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6],

[1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2],

[6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12]

],

8: [

[13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7],

[1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2],

[7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8],

[2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11]

]

}

# Индекс сдвига бит влево в ключе в зависимости от номера раунда

KEY\_ROUND = [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1]

Листинг 2 – Реализация функций in\_notation и ten\_notation для перевода в другие СС:

# Перевод в другую (до 16-ти) систему счисления из десятичной

def in\_notation(String: int, base: int):

try: (String // base)

except TypeError: return 'Вводимые данные должны быть int'

if base < 2 or base > 16:

return 'Неверная система счисления'

else:

extra = {10: 'A', 11: 'B', 12: 'C', 13: 'D', 14: 'E', 15: 'F'}

ans = ''

while String > 0:

if String % base < 10:

ans += str(String % base)

else:

ans += extra[String % base]

String //= base

if ans != '': return ans[::-1]

else: return '0'

# Перевод из любой (до 16ти) в десятиричную систему счисления

def ten\_notation(String: str, base: int):

extra = {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9,

'A': 10, 'B': 11, 'C': 12, 'D': 13, 'E': 14, 'F': 15}

try: all(extra[i] for i, j in zip(String, str(base)))

except TypeError: return 'Неверные входные данные'

if max([extra[i] for i in str(String)]) < int(base):

ans = 0

for num, ind in zip(str(String)[::-1], range(0, len(str(String)))):

if num.isdigit():

ans += int(int(num) \* base \*\* ind)

else:

ans += int(int(extra[num]) \* base \*\* ind)

return ans

else:

return 'Неверная система счисления'

Листинг 3 – Реализация вспомогательных функций get\_info, added, separate для получения текста и пароля, добавления незначащих нулей и разделения текста:

# Ввод текста и ключа

def get\_info():

text\_message = input('Введите текст: \n').rstrip()

password = input('Введите пароль: ').rstrip()

return text\_message, password

# Добавление незначащих нулей

def added(text: str, num: int):

while len(text) % num != 0:

text = '0' + text

if len(text) == 0:

text = '0000'

return text

# разделение текста на блоки по 64 бита

def separate(text: str, num: int):

temp = []

length = 0

line = ''

for i in text: # text = аааааааааааа

if length == num:

temp.append(line)

length = 0

line = i

else:

line += i

length += 1

if line != '': temp.append(line)

return temp

Листинг 3 – Реализация функции get\_pass для получения ключей

def get\_pass(left\_pw, right\_pw):

pwrd = []

# Разбиение ключа на 2 части и смещение на n бит влево

for rnd in range(16):

# Побитовое смещение в зависимости от цикла

left\_pw = left\_pw[KEY\_ROUND[rnd]::] + left\_pw[0:KEY\_ROUND[rnd]]

right\_pw = right\_pw[KEY\_ROUND[rnd]::] + right\_pw[0:KEY\_ROUND[rnd]]

# Перестановка бит в ключе

new\_pass = left\_pw + right\_pw

new\_pass = ''.join([new\_pass[i - 1] for i in KEY\_PERMUTATION\_SECOND])

# Разделение ключа по 4 бита и перевод в шестнадцатиричную систему

new\_pass = ''.join(separate(new\_pass, 4))

pwrd.append(''.join([in\_notation(ten\_notation(i, 2), 16) for i in separate(new\_pass, 4)]))

return pwrd

Листинг 4 – Реализация функции XOR:

def XOR(arg1, arg2) -> str: return str(int(arg1) ^ int(arg2))

Листинг 5 – Реализация функции encrypted для шифрования/дешифрования:

def encrypted(message, password):

message = ''.join([added(in\_notation(ten\_notation(sym, 16), 2), 4) for sym in message])

# Первая перестанвока бит в сообщении

new\_message = ''.join([message[i - 1] for i in FIRST\_PERMUTATION])

# print(f'После начальной перестановки: {"".join([in\_notation(ten\_notation(i, 2), 16) for i in separate(new\_message, 4)])}')

L0 = new\_message[0:32]

R0 = new\_message[32::]

# 16 раундов

for rnd in range(16):

# Правая часть сообщения проходит вторую перестановку и XOR с ключом

R0\_perm = ''.join([R0[i - 1] for i in SECOND\_PERMUTATION])

bin\_key = ''.join([added(in\_notation(ten\_notation(i, 16), 2), 4) for i in password[rnd]])

cipher = ''.join([XOR(a, b) for a,b in zip(bin\_key, R0\_perm)])

# Разделение XORed сообщения на блоки по 6 бит

S\_blocks = separate(cipher, 6)

S\_cipher = []

# Проход по S-блоку

for ind in range(len(S\_blocks)):

first = S\_blocks[ind][0] + S\_blocks[ind][-1]

second = S\_blocks[ind][1:5]

first = ten\_notation(first, 2)

second = ten\_notation(second, 2)

S\_cipher.append(CIPHER\_S\_BLOCKS[ind + 1][first][second])

# Перевод в двоичную S-блок сообщения

S\_cipher = ''.join([added(in\_notation(i, 2), 4) for i in S\_cipher])

# Последняя перестановка в S-блок сообщении и XOR с левой частью

cipher = ''.join([S\_cipher[i - 1] for i in LAST\_PERMUTATION])

cipher = ''.join([XOR(a, b) for a, b in zip(cipher, L0)])

L0 = cipher

if (rnd != 15):

L0, R0 = R0, L0

# print(f'Round {rnd + 1}', ''.join([in\_notation(ten\_notation(i, 2), 16) for i in separate(L0, 4)]), ''.join([in\_notation(ten\_notation(i, 2), 16) for i in separate(R0, 4)]), password[rnd])

cipher = L0 + R0

cipher = ''.join([cipher[i - 1] for i in IP\_PERMUTATION])

cipher = separate(cipher, 4)

cipher = ''.join(in\_notation(ten\_notation(i, 2), 16) for i in cipher)

return cipher

Листинг 6 – Реализация функции main для взаимодействия с пользователем

def main():

try: x = int(input('Вы хотите кодировать(0) или декодировать(1) текст?: '))

except ValueError: return print('Либо 0, либо 1, а не буковки')

if x == 0:

message, password = get\_info()

message = list(message)

for i in range(len(message)):

message[i] = separate(message[i].encode('UTF-8').hex().upper(), 16)

ans = []

password = added(password.encode('UTF-8').hex().upper(), 16)

password = ''.join([added(in\_notation(ten\_notation(sym, 16), 2), 4) for sym in password])

# Создание 56-битного ключа перестановкой P-блока

password = ''.join([password[i - 1] for i in KEY\_PERMUTATION\_FIRST])

left\_pw = password[:len(password) // 2]

right\_pw = password[(len(password) // 2)::]

pwrd = get\_pass(left\_pw, right\_pw)

# return print(f'Зашифрованное сообщение: {encrypted(message, pwrd)}')

for msg in message:

temp = []

for i in msg:

i = added(i, 16)

temp.append(encrypted(i, pwrd))

ans.append(' '.join(temp))

return [print(i, end=' ') for i in ans]

if x == 1:

# Создание 56-битного ключа перестановкой P-блока

message, password = get\_info()

message = message.split(' ')

password = added(password.encode('UTF-8').hex().upper(), 16)

password = ''.join([added(in\_notation(ten\_notation(sym, 16), 2), 4) for sym in password])

# Создание 56-битного ключа перестановкой P-блока

password = ''.join([password[i - 1] for i in KEY\_PERMUTATION\_FIRST])

left\_pw = password[:len(password) // 2]

right\_pw = password[(len(password) // 2)::]

pwrd = get\_pass(left\_pw, right\_pw)

temp = []

ans = []

try:

for msg in message:

a = (bytes.fromhex(encrypted(msg, pwrd[::-1])).decode('UTF-8')).replace('\x00', '')

temp.append(a)

return print(''.join(temp))

except: return print('Ошибка. Вероятно, неправильный пароль')

else:

return print('Вводить надо либо 0, либо 1')

if \_\_name\_\_ == '\_\_main\_\_':

main()

ОТЗЫВ-ХАРАКТЕРИСТИКА

Обучающийся Карпов Роман Александрович

группа ВКБ12

кафедра «Кибербезопасность информационных систем»

вид практики ознакомительная практика

наименование места практики АО «Алюминий Металлург Рус»

(наименование предприятия, структурного подразделения)

Обучающийся выполнил задания программы практики:

Реализация криптографического алгоритма DES. Изучил задание, собрал информацию, разработал структуру, организацию и управление проекта, реализовал код, протестировал его, изготовил готовый продукт.

Дополнительно ознакомился/изучил с

Криптографическим алгоритмом DES, его реализацией, с использованием в реальных проектах.

Встроенными функциями Python повышенного уровня и началами объектно-ориентированного программирования

Заслуживает оценки \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
|  | Руководитель практики  от предприятия  С.А. Кириченко  «\_\_\_» \_\_\_\_\_\_\_\_\_\_2023 г. |