МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведущий кафедрой №____

должность, уч. степень, звание	подпись, дата	инициалы, фамилия

БАКАЛАВРСКАЯ РАБОТА

на тему _____

выполнена _____
фамилия, имя, отчество студента в творительном падеже

по направлению подготовки .03. _____
код наименование направления

_____
наименование направления

направленности _____ _____
код наименование направленности

_____
наименование направленности

Студент группы № _____ _____ _____
подпись, дата инициалы, фамилия

Руководитель

должность, уч. степень, звание	подпись, дата	инициалы, фамилия

В работе не содержится информация с ограниченным доступом и отсутствует сведения, представляющие коммерческую ценность.

подпись руководителя _____ подпись студента

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

УТВЕРЖДАЮ
Заведущий кафедрой №____

_____	_____	_____
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ЗАДАНИЕ НА ВЫПОЛНЕНИЕ БАКАЛАВРСКОЙ РАБОТЫ

студенту группы № _____ _____
фамилия, имя, отчество

на тему _____

_____

_____

утвержденную приказом ГУАП от _____ № _____

Цель работы: _____

_____

_____

Задачи, подлежащие решению: _____

_____

_____

Содержание работы (основные разделы): _____

_____

_____

Срок сдачи работы « ____ » _____ 2016

Руководитель

_____	_____	_____
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

Задание принял к исполнению
Студент группы № _____

_____	_____
подпись, дата	инициалы, фамилия

Содержание

Список сокращений

GnuPG, GPG – GNU Privacy Guard.

ECB – Electronic Codebook (режим электронной кодовой книги , режим простой замены).

CBC – Cipher Block Chaining (режим сцепления блоков шифротекста).

CFB – Cipher Feedback (режим обратной связи по шифротексту).

OFB – Output Feedback (режим обратной связи по выходу).

GPL – GNU General Public License.

MPL – Mozilla Public License.

Введение

На современном этапе развития IT-технологий имеются большие возможности для несанкционированного доступа к хранимой информации. Существенную роль играет вопрос о защите данных пользователя.

Для защиты данных от несанкционированного доступа используется множество криптографических методов:

- Симметричное шифрование;
- Асимметричное шифрование;
- Цифровые подписи;
- Хеширование.

Существует большое количество средств, использующих криптографические методы для защиты данных пользователя, шифрования сообщений, подписания документов и т.д.

Сегодня у человека существует потребность в хранении большого объема конфиденциальной информации. К этому понятию можно отнести персональные данные, телефонные номера, контакты, пароли, личные заметки. Для решения этой задачи разрабатываются пакеты прикладных программ, облачные сервисы и клиентские приложения.

Целью работы является проектирование и разработка программного средства с графическим интерфейсом для создания, редактирования и просмотра зашифрованных текстовых документов “на лету”.

Для достижения этой цели были поставлены следующие задачи:

1. Провести исследование и составить характеристику существующих пакетов и клиентских приложений, предоставляющих аналогичный функционал;
2. На основании проведенного исследования и обзора определить задачи по разработке программного средства.

1 Анализ предметной области

1.1 Обзор аналогичных программ

1.1.1 GNU Privacy Guard

GNU Privacy Guard (GnuPG, GPG) – программа для шифрования информации и создания электронных цифровых подписей [1]. GNU Privacy Guard является продуктом с открытым кодом, созданным сообществом разработчиков. Распространяется под лицензией General Public License (GPL). Пакет изначально присутсвует в любом дистрибутиве Linux и используется во многих приложениях.

Основные функции GnuPG:

- Шифрование текста и файлов (используется несколько алгоритмов);
- Подписывание документов электронной цифровой подписью и проверка чужих подписей;
- Создание и управление списками открытых ключей респондентов.

Следует отметить, что GnuPG является утилитой, которая управляется только из командной строки. Конечно, для упрощения работы с GPG существуют несколько графических оболочек, а также множество компонент для разных языков.

В GnuPG используются разные криптографические алгоритмы: симметричные шифры, шифрование с открытым ключом и смешанные алгоритмы. Основной особенностью является система ключей. В GnuPG вы создаете себе несколько ключей, причем каждый служит для отдельного действия (и использует разные алгоритмы). Длина ключа может быть в пределах от 1024 до 4096 бит.

Для работы с документами доступны несколько команд:

- Подписать документ;
- Зашифровать документ;
- Подписать и зашифровать документ;
- Проверить подпись.

Для шифрования файлов GnuPG использует не чистый алгоритм с открытым ключом, а смешанный – для документа формируется уникальный сеансовый ключ, который шифруется шифром с открытым ключом, текст документа шифруется симметричным шифром на основе сеансового ключа. Получатель с помощью своего секретного ключа расшифровывает сначала сеансовый ключ, а потом с его помощью сам документ. Такая схема работает быстрее, чем шифрование с открытым ключом, а по надежности (криптостойкости) равна надежности самого слабого из алгоритмов.

Теоретически, это позволяет упростить задачу дешифровки – вместо взлома системы шифрования с открытым ключом надо взломать только симметричный шифр, то есть

подобрать сеансовый ключ. Но такой метод даст возможность расшифровать только одно сообщение, ведь для каждого документа сеансовый ключ генерируется отдельно. Для симметричного шифрования по умолчанию применяется алгоритм CAST5.

GPG является многофункциональным инструментом для шифрования. Он не дает возможности редактировать зашифрованные файлы без предварительной расшифровки. Поэтому в дополнение к самому пакету необходимо иметь оболочку, которая предоставляет такой функционал.

1.1.2 Vi/Vim

Vim – улучшенная версия консольного текстового редактора Vi для операционных систем семейства UNIX. Редактор имеет множество функций, одной из которых является возможность работать с зашифрованными текстовыми файлами. Для шифрования используется алгоритм Blowfish.

При создании файла можно указать флаг `-x`:

```
vim -x filename.txt
```

После чего редактор просит пользователя ввести пароль:

```
Enter encryption key: *****
Enter same key again: *****
```

Дальнейшая работа с зашифрованными файлами в редакторе vim ничем не отличается от редактирования обычных текстовых файлов. Алгоритм Blowfish является симметричным, поэтому при открытии файла необходимо ввести тот же самый пароль.

Особенностью Vim является отсутствие необходимости сохранять расшифрованную копию файла на жесткий диск перед его изменением.

1.1.3 CryptoTE

CryptoTE – текстовый редактор со встроенной функцией шифрования. Данная программа позволяет создавать зашифрованные контейнеры, которые могут содержать несколько файлов. Файлы могут быть как текстовыми, так и бинарными (изображения, архивы и т. п.).

Программа позволяет редактировать текстовые файлы внутри контейнера (рисунок 1.1), просматривать бинарные файлы в шестнадцатиричном формате. Имеется возможность импортировать и экспортировать файлы из контейнера.

CryptoTE по умолчанию использует симметричный алгоритм Serpent и библиотеку zlib для сжатия данных. Имеется встроенный генератор паролей.

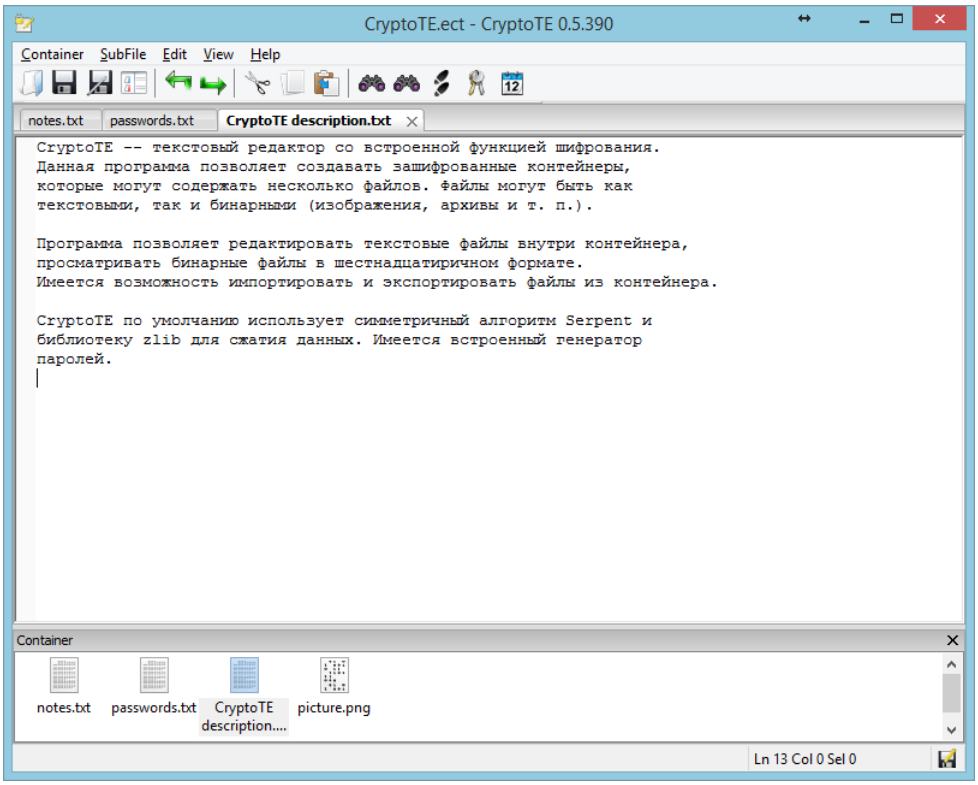

Рисунок 1.1 – Внешний вид редактора CryptoTE

1.1.4 KeyNote NF

KeyNote NF – структурный редактор (англ. outliner), позволяющий структурированно хранить и редактировать заметки.

Файл с заметками может содержать несколько вкладок, а каждая вкладка является либо деревом заметок, либо одной заметкой с форматированием (рисунок 1.2). Сами заметки хранятся в формате RTF (Rich Text Format). Программа позволяет шифровать файл с заметками с помощью пароля.

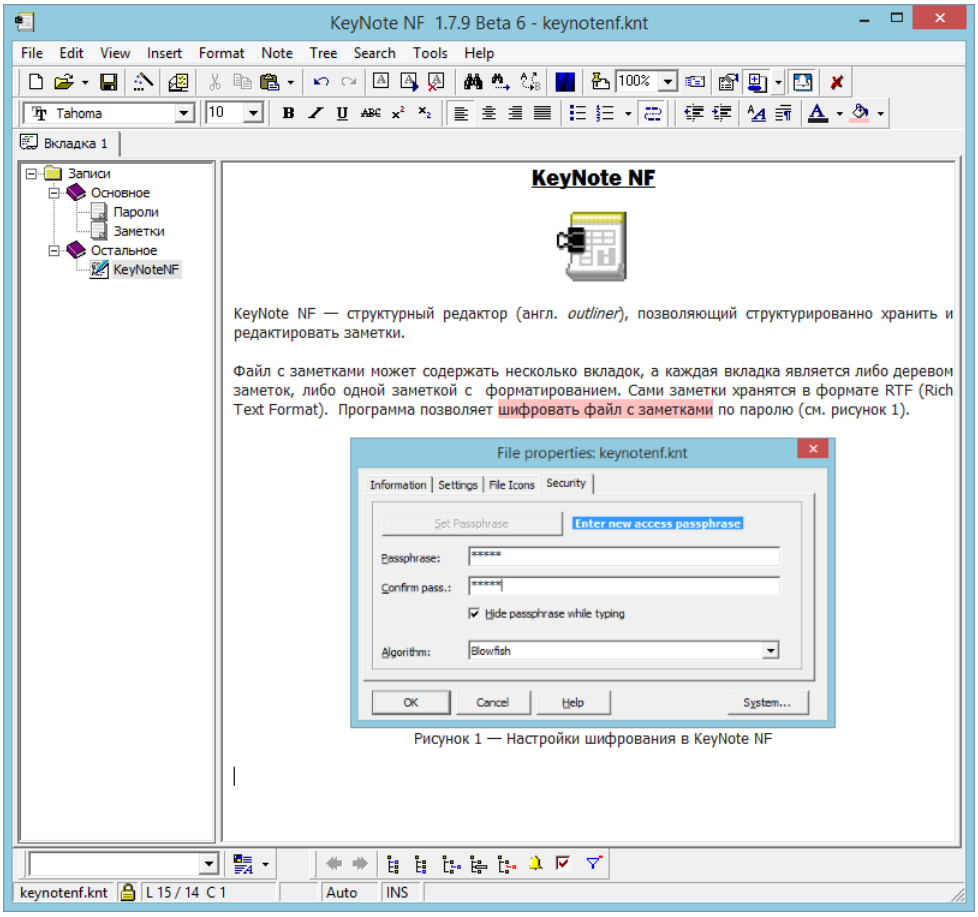

Рисунок 1 — Настройки шифрования в KeyNote NF

Рисунок 1.2 – Внешний вид редактора KeyNote NF

1.1.5 ProtectedText

Приложение представляет из себя веб-сайт [1], на котором любой пользователь имеет возможность бесплатно создать одну или несколько страниц для хранения своих записей в текстовом формате. Доступ к страницам осуществляется по паролю, который вводит пользователь при первом сохранении текста на странице. Страница может может содержать несколько вкладок с текстовыми файлами (рисунок 1.3).

По заверениям разработчиков, на сервере хранится только зашифрованный текст и шифрование/дешифрование осуществляется на стороне пользователя. С записками можно работать используя как персональный компьютер, так и планшеты, смартфоны и остальные устройства с доступом в интернет. При этом не требуется регистрация на сайте.

Стоит заметить, что пользователь может потерять доступ к своим записям не только забыв пароль, но и потеряв ссылку на свою страницу с записями.

Пользователи устройств с ОС Anroid могут воспользоваться приложением Safe Notes от разработчиков ProtectedText. Приложение позволяет создавать, редактировать и шифровать заметки на мобильном устройстве, а так же имеет возможность синхронизировать записи с ProtectedText.

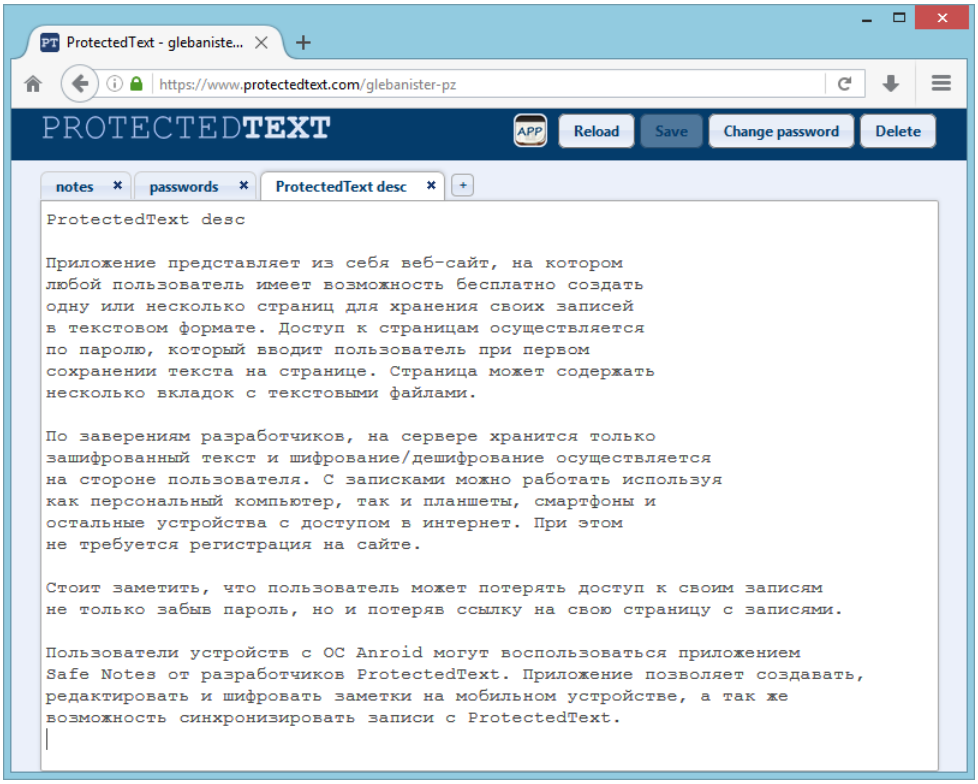

Рисунок 1.3 – Внешний вид редактора ProtectedText

[1] ProtectedText – https://www.protectedtext.com/

1.1.6 NotepadCrypt

NotepadCrypt – простой текстовый редактор, основанный на Notepad2. От Notepad2 редактор отличается наличием опции шифрования редактируемого файла.

Приложение по умолчанию сохраняет текстовые файлы в незашифрованном виде. Поэтому перед сохранением файла необходимо указать пароль, по которому будет производиться шифрование (рисунок 1.4). Программа использует алгоритм шифрования AES-256.

Также стоит отметить возможность указать мастер-пароль (англ. master passphrase). Файлы сохраненные с указанным мастер-паролем могут быть восстановлены в случае утери пароля от зашифрованного файла.

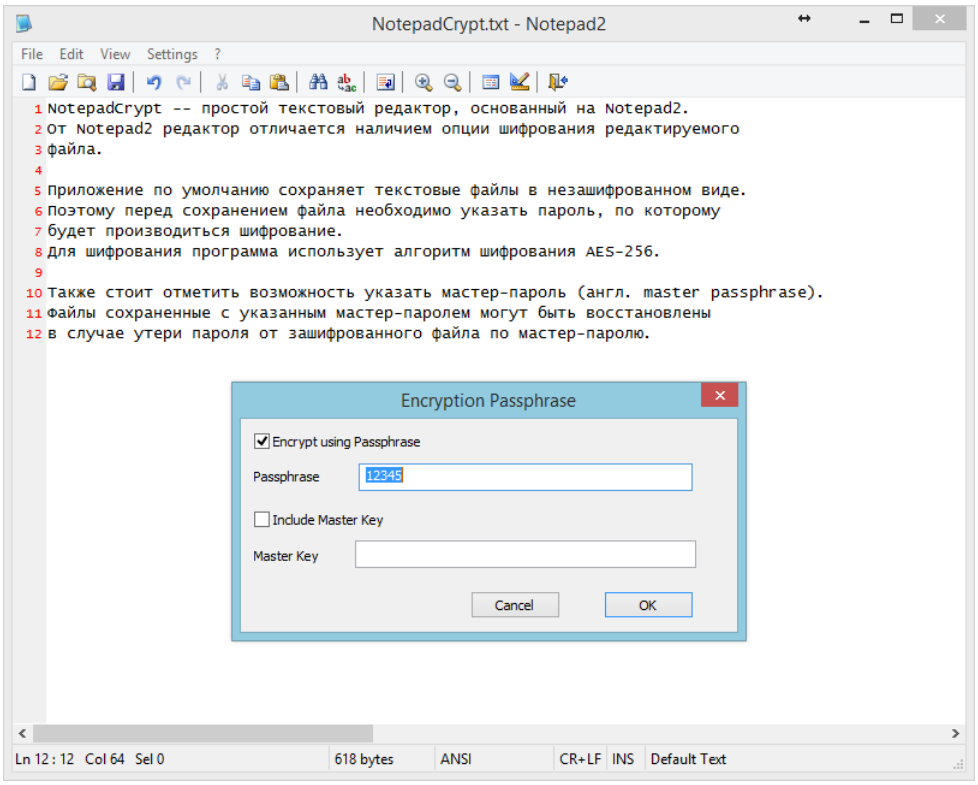

Рисунок 1.4 – Внешний вид редактора NotepadCrypt

1.1.7 Notepad++ (с аддоном NppCrypt)

Notepad++ – текстовый редактор, позволяющий расширить свой функционал с помощью плагинов. Одним из интересных плагинов является NppCrypt. Плагин дает возможность пользователю зашифровать текст.

В отличие от описаных ранее редакторов, плагин лишь изменяет выделенный текст внутри редактора. Таким образом перед сохранением отредактированного текста пользователю необходимо его зашифровать, выбрав соответствующую опцию в меню доступа к функциям плагина NppCrypt (рисунок 1.5). Аналогично после открытия зашифрованного файла необходимо выполнить дешифрование, после чего можно приступить к редактированию текста. Такая работа с файлом не слишком удобна и существенно снижает эффективность пользователя.

В то же время плагин имеет довольно гибкие настройки и множество опций симметричного шифрования. Шифротекст может храниться в бинарном, в шестнадцатиричном виде или в кодировке base64. Текст можно зашифровать одним из множества алгоритмов (IDEA, CAST5, Blowfish, AES-256 и т. д.). Плагин позволяет выбрать один из нескольких режимов шифрования (ECB, CBC, CFB, OFB и т. д.). Также имеются настройки алгоритма генерации ключа по паролю.

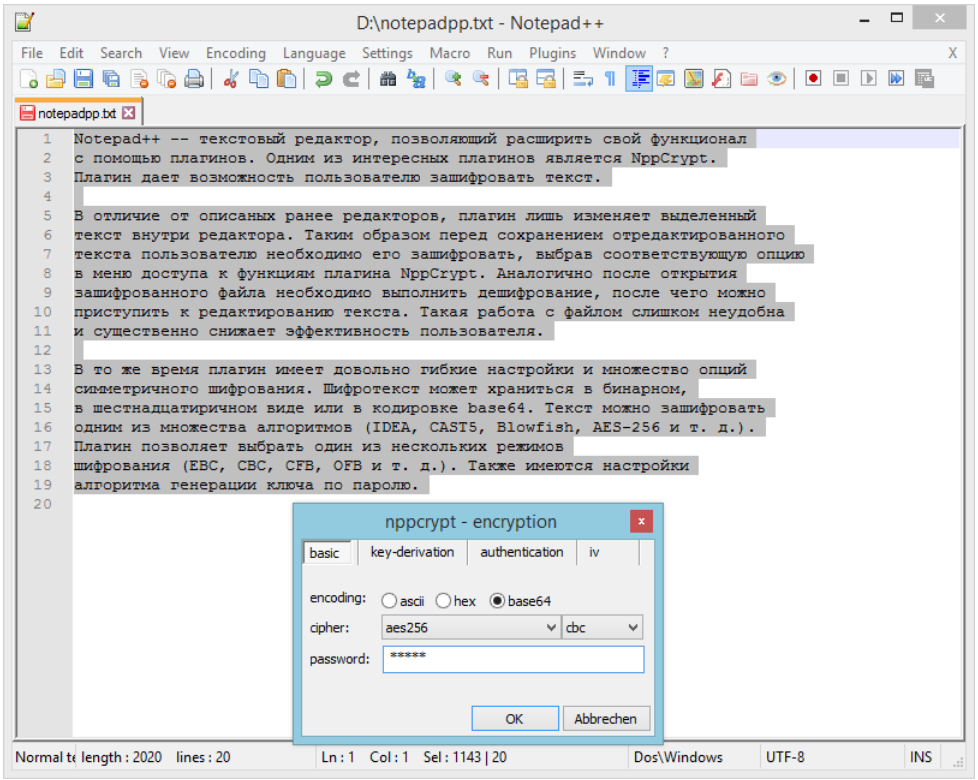

Рисунок 1.5 – Внешний вид редактора Notepad++

1.2 Сравнительный анализ

В таблице 1 приведен сравнительный анализ описанных в пункте 1.1 программных средств для шифрования текстовых документов.

Таблица 1 – Сравнение приложений для шифрования текстовых записей

Название	Лицензия	Операционная система	Криптографические методы	Графический интерфейс
GNU Privacy Guard	GNU GPLv3	Linux, Windows, Mac OS X	AES, DES, Blowfish, CAST5, MD5, SHA1, SHA256, SHA512, RSA, ...	–
Vim	GPL-совместимая	Linux, Windows, OS X	Blowfish	–
CryptoTE	GNU GPLv2	Linux, Windows	Serpent	+
KeyNote NF	MPLv2.0	Windows	Blowfish, IDEA	+
ProtectedText	–	Веб-приложение	AES, SHA256	+
NotepadCrypt	GNU GPLv3	Windows	AES, SHA256	+
Notepad++ (NppCrypt)	GNU GPLv3	Linux, Windows	AES, DES, CAST5, IDEA, Blowfish, MD5, SHA1, SHA256, SHA512, ...	+

1.3 Постановка задачи

Хранение и быстрый доступ к конфиденциальной информации – важная необходимость в жизни современного человека. Сравнительный анализ показал, что актуально и целесообразно спроектировать и реализовать программный продукт в соответствии со следующими требованиями:

– Наличие графического интерфейса;

– Работа с зашифрованными файлами с использованием прозрачного шифрования;

– Использование различных алгоритмов шифрования;

– Мультиплатформенность.

2 Теоретические сведения

2.1 Языки программирования и библиотеки

В качестве основного языка был выбран Python – высокоуровневый интерпретируемый язык программирования общего назначения с динамической типизацией данных [**?**].

К достоинствам Python можно отнести:

– Язык является интерпретируемым (не требует компиляции), что повышает производительность разработчика при написании и отладке программного кода.

– Кроссплатформенность. Основной интерпретатор языка (CPython) реализован для большинства активно используемых платформ, что позволяет использовать один и тот же код на любой системе.

– Имеется большая стандартная библиотека и множество пользовательских модулей, находящихся в открытом доступе.

Основные недостатки языка:

– Низкое быстродействие;

– Не слишком удачная поддержка многопоточности.

Реализуемая программа состоит из двух частей:

1. Графический интерфейс;
2. Подсистема шифрования.

Графический интерфейс программы был реализован с использованием библиотеки Qt5 и ее привязки к Python – PyQt5.

Несмотря на особенности выполняемой задачи (шифрование текстовых документов), в общем случае шифрование должно выполняться эффективно по времени. Из-за низкой скорости выполнения кода использовать Python для реализации криптографических методов, в частности алгоритмов шифрования, является неразумным. Таким образом появляется две альтернативы: использовать язык C (стандарт C99) или язык ассемблера. В разделе 3 приведен краткий анализ реализаций алгоритмов шифрования на языке ассемблера и C.

2.2 Алгоритмы шифрования

В проекте на данный момент используются лишь два алгоритма шифрования: FEAL-4 и Blowfish. Выбор первого обусловлен простотой реализации и довольно высокой скоростью работы, несмотря на невысокую криптографическую стойкость: шифр может быть взломан по пяти подобранным исходным блокам с использованием линейного криптоанализа [5].

Алгоритм Blowfish является безопасным незапатентованным свободным для использования шифром. На данный момент, в отличие от FEAL-4, не существует эффективных методов взлома шифра Blowfish.

Алгоритм FEAL (Fast data Encipherment ALgorithm) был разработан криптографами Акихиро Шимизу (Akihiro Shimizu) и Шоджи Миягучи (Shoji Miyaguchi) из японской компании NTT в 1987 г. [4, стр. 206].

Разработчики алгоритма продвигали FEAL как потенциальную замену стандарта шифрования DES – по их мнению, FEAL-4 (число в названии обозначает количество раундов алгоритма) предлагал более быстрое шифрование без потери его качества. Кроме того, в FEAL отсутствуют табличные замены, поэтому реализация алгоритма не требует дополнительной энергонезависимой памяти для хранения таблиц.

На рисунке 2.1 представлена структура общего варианта алгоритма FEAL – FEAL-NX – вместе с функцией расширения ключа. В упрощенном варианте алгоритма FEAL-4 функция расширения не используется, поэтому рассмотрена не будет.

Алгоритм FEAL-4 имеет структуру сети Фейстеля: в каждом раунде выполняется обработка правого 32-битного подблока функцией $F(R, k_{16})$, где R – правый подблок данных, k_{16} – 16-битное значение ключа раунда. Результат обработки накладывается на левой подблок операцией XOR. Перед первым раундом выполняется операция XOR блока шифруемых данных с ключом, а также аналогичное наложение левого подблока на правый. После финального раунда алгоритма также производятся те же действия: наложение левого блока на правый и сложение с ключом. Расшифрование выполняется аналогично, но 16-битовые подключи подставляются в обратном порядке.

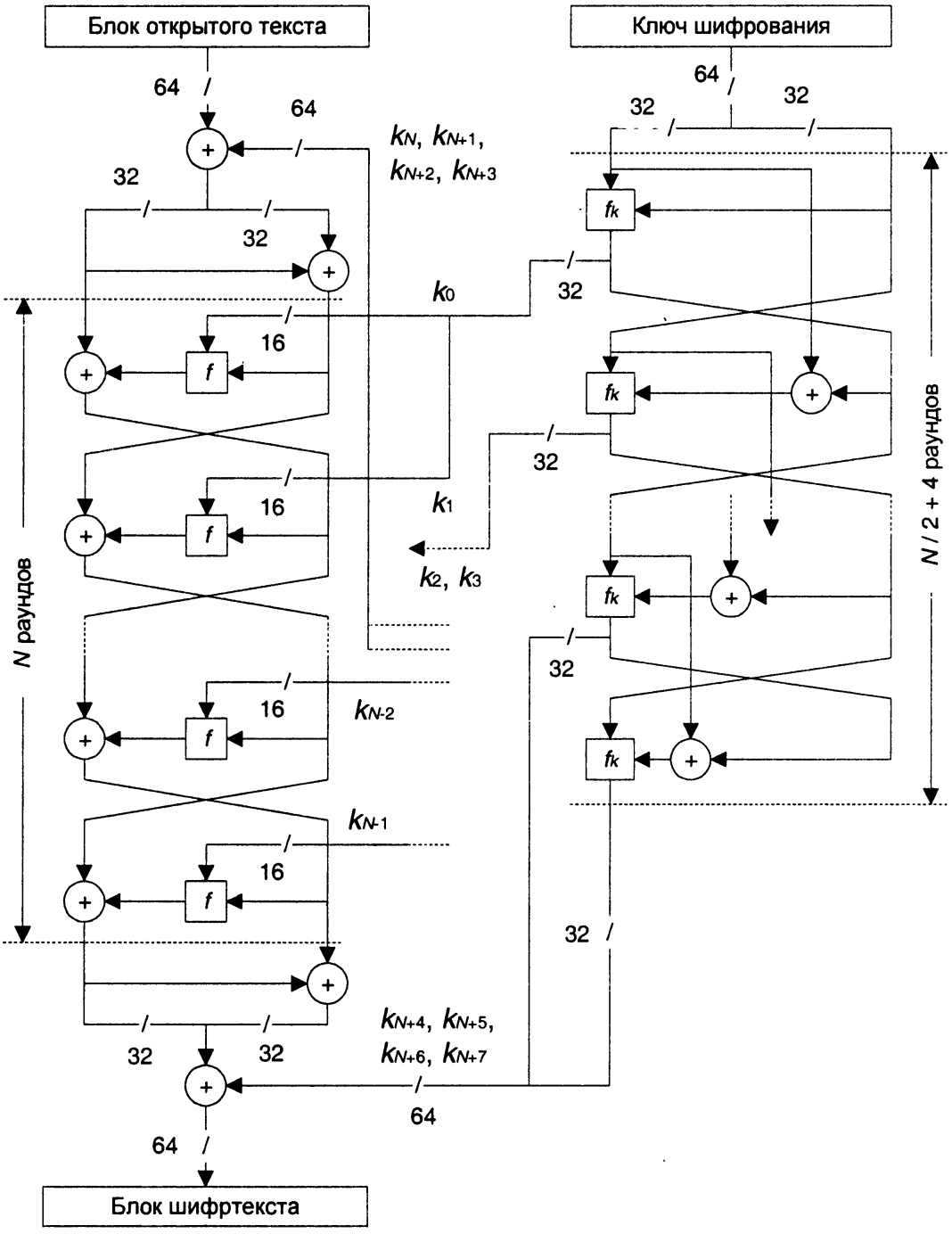

Рисунок 2.1 – Структура алгоритма FEAL-NX.

Структура функции F представлена на рисунке 2.2. В ней используются лишь три вида преобразований: операция XOR, функции S_0 и S_1, которые можно описать следующим образом:

$$S_0 = \mathrm{ROL}_2((x + y) \mathrm{mod} 2^8),$$
$$S_1 = \mathrm{ROL}_2((x + y + 1) \mathrm{mod} 2^8),$$

где ROL_2 – операция циклического сдвига влево на два разряда.

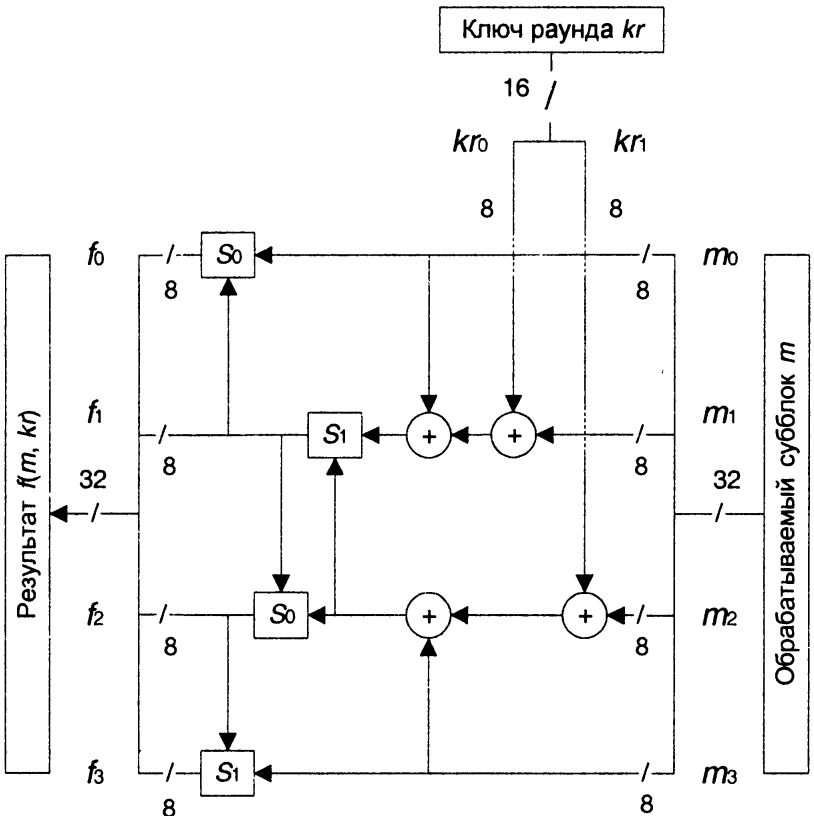

Рисунок 2.2 – Функция F алгоритма FEAL-4.

Алгоритм FEAL-4, изначально предлагавшийся в качестве замены стандарта DES, оказался весьма нестоек к различным видам криптоанализа. Та же участь постигла и алгоритм FEAL-8. Алгоритмы FEAL-16, FEAL-32 и т. д. оказались существенно более стойкими за счет большего количества раундов. Однако скорость их работы оказалась ниже скорости алгоритма DES. Поэтому, как стандарт шифрования, алгоритм FEAL не состоялся.

Алгоритм Blowfish разработан Брюсом Шнайером в 1994 г. также в качестве замены DES [4, стр. 118]. Алгоритм оказался весьма “удачным”. Он очень широко реализован в различных шифровальных средствах.

Blowfish, как и FEAL-4, шифрует данные 64-битными блоками. Однако размер ключа может варьироваться от 32 до 448 бит. Алгоритм также представляет собой сеть Фейстеля. Его структура приведена на рисунке 2.3.

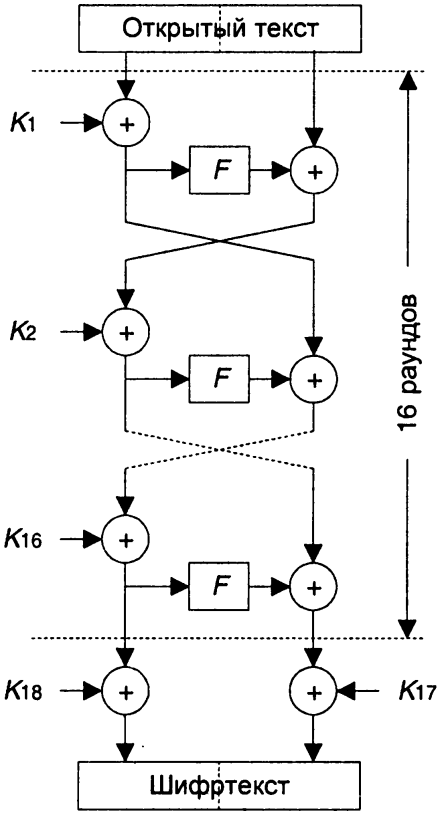

Рисунок 2.3 – Структура алгоритма Blowfish.

Шифрование данных выполняется в 16 раундов, в каждом из которых над левым 32-битным подблоком данных производятся следующие действия [6]:

1. Значение подблока складывается с ключом i-го раунда K_i операцией XOR, результат операции становится новым значением подблока.
2. Подблок обрабатывается функцией F. Результат обработки накладывается на правый подблок операцией XOR.
3. Подблоки меняются местами во всех раундах, кроме последнего.

После 16 раундов выполняется наложение на подблоки еще двух подключей: K_{17} и K_{18} складываются операцией XOR с правым и левым подблоками соответственно.

Функция F (рисунок 2.4) обрабатывает подблок следующим образом:

1. 32-битное входное значение делится на четыре фрагмента по восемь битов, каждый из которых является индексом соответствующих таблиц замен $S_1..S_4$.
2. Значения из таблиц S_1 и S_2 складываются по модулю 2^{32}.
3. Полученное значение складывается операцией XOR со значением из S_3.
4. Полученное значение складывается по модулю 2^{32} со значением из S_4.

Расшифрование выполняется аналогично шифрованию, но ключи $K_1..K_{18}$ используются в обратном порядке.

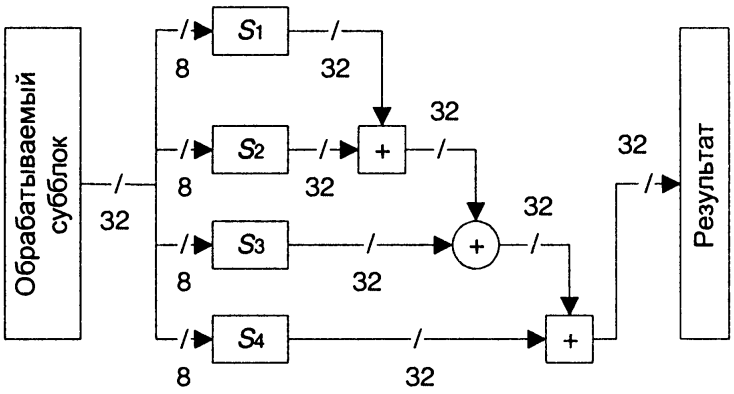

Рисунок 2.4 – Функция F алгоритма Blowfish.

В отличие от FEAL-4, перед шифрованием алгоритмом Blowfish необходимо сгенерировать раундовые 32-битные ключи $K_1..K_{18}$ и четыре таблицы замены $S_1..S_4$. Каждая таблица замены содержит 256 32-битных значений.

Процедура расширения ключа состоит из следующих шагов:

1. Исходные значения ключей раундов и табилиц замен инициализируются псевдослучайно строкой, в качестве которой используется шестнадцатеричная запись дробной части числа π. Исходные значения ключей и таблиц можно найти в приложении В.
2. Операцией XOR на K_1 накладываются первые 32-бита ключа шифрования, на K_2 – следующие 32 бита и т.д. до K_{18}. Ключ шифрования накладывается циклически, если используется более короткий ключ шифрования, чем необходимо для наложения на $K_1..K_{18}$.
3. С использованием полученных ключей раундом и таблиц замен выполняется шифрование алгоритмом Blowfish блока данных, состоящего из 64 нулевых битов. Результат становится новым значением ключей K_1 и K_2.
4. Используя измененные значения ключей K_1 и K_2, шифруется результат, полученный на предыдущем этапе. В результате получаются новые значения ключей K_3 и K_4.
5. Шифрование выполняется до тех пор, пока новыми значениями не будут заполнены все ключи раундов и таблицы замен.

Особенностью алгоритма Blowfish является то, что он не годится для применения в случаях, где требуется частая смена ключей. Процедура расширения ключа является достаточно ресурсоемкой, поэтому одно из достоинств алгоритма Blowfish – достаточно высокая скорость шифрования – проявляется только в тех случаях, если на одном ключе шифруется достаточно большой объем информации. И наоборот, если менять ключ после каждого из шифруемых блоков, скорость алгоритма становится катастрофически низкой именно из-за необходимости каждый раз выполнять расширения ключа.

Явные достоинства и отсутствие критичных недостатков предопределили широкое использование алгоритма Blowfish.

2.3 Хранение данных

Для хранения зашифрованных данных необходимо обеспечить:

– Целостность данных;

– Поддержка нескольких алгоритмов шифрования и хэширования;

Для хранения зашифрованных данных была спроектирована структура файла, включающая в себя:

1. Заголовок.
    а) Магическое число (сигнатура) – 4 байта : `'tfe', 0x42`.
    б) Используемый алгоритм – 1 байт.
    в) Используемая функция хэширования – 1 байт.
    г) Отступ до зашифрованных данных от начала заголовка – 2 байта.
    д) Размер исходных зашифрованных данных – 8 байт.
2. Преамбула.
    а) Хэш первых 512 байт исходных данных – размер зависит от используемой функции хэширования.
    б) Соль для пароля – 16 байт.
3. Зашифрованные данные.

Используемый алгоритм и функция хэширования являются номерами в таблицах функций алгоритмов и функций хэширования. На данный момент реализована поддержка следующих функций:

– Функции шифрования:

    1. FEAL-4;
    2. Blowfish;

– Функции хэширования:

    1. MD5;

При использовании блочного шифра в режиме EBC необходимо дополнить исходные данные, чтобы их размер стал кратен размеру блока. В реализации последний блок данных дополняется необходимым количеством байтов со значением 0x42. При дешифровании лишние байты отрезаются, так как известен размер зашифрованных данных.

Формат позовляет хранить зашифрованные данные любого типа, есть возможность проверить полученные после расшифрования данные на соответствие исходным данным (то есть проверить был ли использован правильный ключ при расшифровании).

Однако такой формат может быть контейнером только для одного файла. Отсутствует поддержка алгоритмов сжатия и возможность восстановить тип исходного файла.

Несмотря на недостатки, спроектированный формат зашифрованного файла подойдет для использования в разрабатываемом программном средстве.

2.4 Проектирование интерфейса

Графический пользовательский интерфейс программы должен выглядеть как обычный текстовый редактор (рисунок 2.5). В рамках рассматриваемой задачи интерфейс должен предоставлять пользователю следующие функции:

– Редактирование текстовых файлов;

– Шифрование текстовых файлов;

– Базовые настройки текстового редактора (внешний вид, шрифт);

– Основные настройки шифрования.

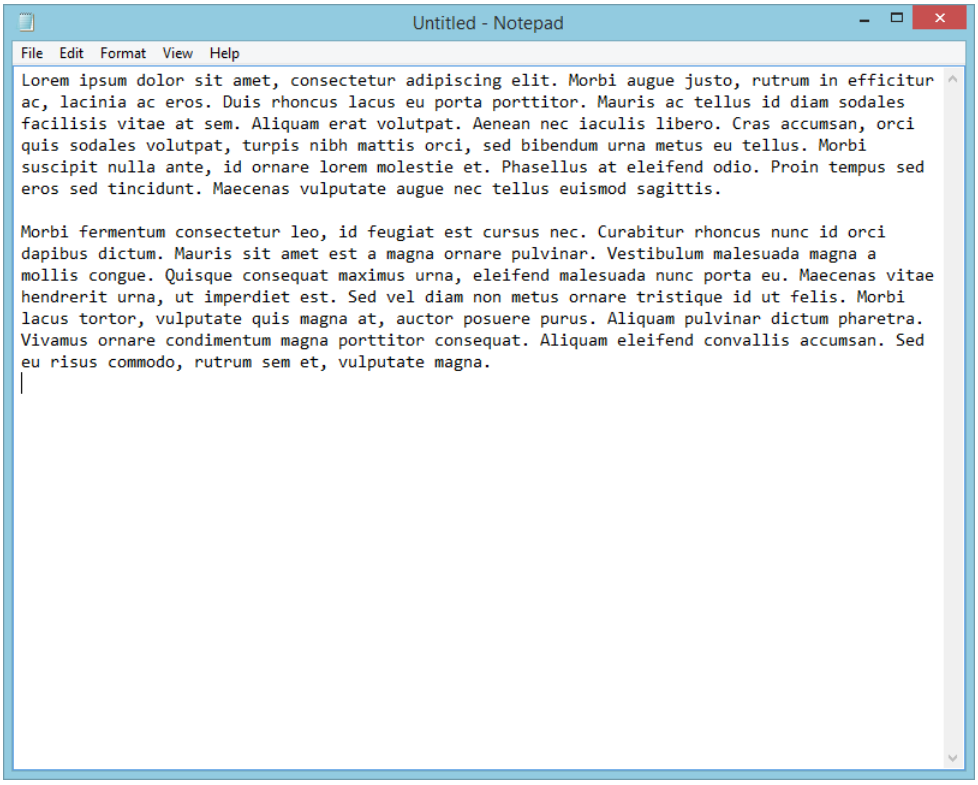

Рисунок 2.5 – Интерфейс обычного текстового редактора.

3 Реализация

3.1 Общее описание проекта

Согласно поставленным задачам, проект должен содержать:

– Реализацию алгоритмов шифрования;

– Модуль для работы с зашифрованными файлами;

– Описание пользовательского интерфейса;

– Файлы локализаций.

Для удобного взаимодействия разработчика с проектом, была использована система контроля версий Git. Структура репозитория была определена следующим образом:

```
└── src                 - Исходный код проекта
    ├── algs            - Код алгоритмов шифрования
    └── app             - Основная программа
        ├── mlibs       - Модули программы (в т.ч. динамические библиотеки)
        │   └── win32   - Динамические библиотеки для платформы Windows
        └── tr          - Файлы локализаций
```

Алгоритмы шифрования подключаются к программе как динамические библиотеки. Реализация каждого алгоритма шифрования содержится в своей директории отдельно от остальных исходников.

Модули программы поделены на несколько частей:

– Привязка динамических библиотек с алгоритмами шифрования к Python;

– Модуль для работы с разработанным форматом файла;

– Описание логики работы графического пользовательского интерфейса.

Файлы локализаций содержат перевод строк, используемых в приложении: ‘Файл’, ‘правка’, ‘сохранить’, ‘копировать’, ‘вставить’ и т. д.

Весь исходный код проекта содержится в репозитории на GitHub. Веб-адрес репозитория: `http://github.com/MrP4p3r/PyTFE`.

Далее представлено содержание проекта на финальной стадии разработки:

```
└── src
    ├── algs
    │   ├── blowfish
    │   │   ├── blowfish.asm        Реализация Blowfish на языке ассемблера
    │   │   ├── blowfishPS.asm      Начальные значения таблиц
    │   │   ├── blowfish.c          Реализация Blowfish на C
    │   │   └── blowfishPS.c        Начальные значения таблиц
    │   └── feal4
    │       ├── feal4.asm           Реализация FEAL-4 на языке ассемблера
    │       └── feal4.c             Реализация FEAL-4 на C
    └── app
        ├── mlibs
        │   ├── __init__.py         *
        │   ├── blowfish.py         Связка библиотеки Blowfish с Python
        │   ├── commonfunctions.py  Общие функции
        │   ├── defaultvalues.py    Константы
        │   ├── feal4.py            Связка библиотеки FEAL-4 с Python
        │   ├── interte.py          Описание интерфейса текстового редактора
        │   ├── mwidgetste.py       Описание дополнительных элентов интерфейса
        │   ├── pbkdf2.py           Реализация алгоритма PBKDF2
        │   ├── tfe.py              Модуль для шифрования/дешифрования
        │   └── win32
        │       ├── blowfish.dll    *
        │       └── feal4.dll       *
        ├── pytfe.pro               *
        ├── pytfete.py              Главный исполняемый файл программы
        ├── setup.py                Скрипт для сборки проекта
        └── tr
            ├── com.ts              Общая таблица переводов
            ├── qt_ru.ts            Русский переводов строк Qt5
            ├── tr_en.ts            Английский перевод
            ├── tr_ru.ts            Русский перевод
            ├── com.qm              *
            ├── qt_ru.qm            *
            ├── tr_en.qm            *
            └── tr_ru.qm            *
```

3.2 Реализация алгоритмов шифрования

Существуют различные режимы шифрования исходного текста блочным шифром: режим простой замены (ECB), режим сцепления блоков (CBC), режим обратной связи по шифротексту (CFB), режим обратной связи по выходу (OFB) и др. Несмотря на явные недостатки режима ECB, из-за простоты реализации был выбран именно он.

При выполнении работы были написаны реализации алгоритмов шифрования FEAL-4 и Blowfish на языке ассемблера и на языке высокого уровня C. Исходный код всех реализаций представлен в приложении А.

Для компиляции исходного кода на языке C был использован компилятор GNU C Compiler (`gcc`), для компиляции ассемблерного кода – Flat Assembler (`fasm`). Был проведен анализ скорости шифрования каждой из реализаций. Результаты приведены в таблице 2.

Таблица 2 – Результаты сравнения скорости работы алгоритмов

Шифр	Скорость шифрования/дешифрования
Blowfish (C)	151.7 Мб/с
Blowfish (ASM)	108.4 Мб/с
FEAL-4 (C)	234.3 Мб/с
FEAL-4 (ASM)	382.4 Мб/с

Скорость выполнения кода на языке C зависит от качества оптимизаций, которые проводит компилятор. А скорость выполнения ассемблерного кода прямо зависит от исходного кода, поскольку выполняется лишь трансляция в машинный код.

Результаты сравнения скорости работы реализаций алгоритмов получились несколько противоречивыми: FEAL-4, написанный на языке C, работает медленнее неоптимизированной ассемблерной реализации, в то время как Blowfish работает быстрее реализации на языке ассемблера, код которой был написан относительно аккуратно.

В любом случае, переносимость кода все-таки перевешивает несущественное уменьшение производительности одного из алгоритмов. Поэтому в качестве основных реализаций алгоритмов, которые будут использованы в программе, были выбраны реализации на языке C.

Исходный код был скомпилирован компилятором GNU C Compiler в динамические библиотеки feal4.dll и blowfish.dll для использования в программе.

3.3 Связка динамических библиотек с Python

Подключение модулей в Python осуществляется оператором `import`. Однако таким образом могут быть подключены лишь модули, написанные на Python. Для подключения динамических библиотек в Python используется модуль ctypes.

В ctypes имеются функции для работы с динамическими библиотеками (.DLL) в Windows и shared objects (.SO) на UNIX-подобных системах. Однако в рамках работы была добавлена лишь поддержка DLL.

Разработанные динамические библиотеки необходимо связать с функциями на Python. Для удобства обращения к этим функциям, они были оформлены как методы классов. Каждый класс имеет одинаковый интерфейс (набор методов): функции шифрования/дешифрования чанка, шифрования/дешифрования одного блока и конструктора класса. Это позволяет упростить реализацию общей функции шифрования: ее реализация не зависит от определенного алгоритма шифрования.

Связка динамических библиотек с Python состоит из двух модулей: `feal4.py` и `blowfish.py` (код модулей можно найти в приложении Б). В них содержатся описания классов feal4 и blowfish. При инициализации экземпляров класса необходимо задать секретный ключ, по которому будет производиться шифрование. После этого можно начинать использовать использовать объекты через вышеописанный интерфейс.

3.4 Модуль для работы с зашифрованными файлами

Для шифрования и упаковки данных пользователя был разработан отдельный модуль, на языке Python (исходный код имеется в приложении Б). Модуль содержит набор констант, таблицы алгоритмов шифрования и хэширования и набор базовых функций для работы с форматом, таких как формирование/чтение заголовка файла, шифрование/дешифрование буфера [2], шифрование/дешифрование файлов.

Таблица алгоритмов шифрования для каждого алгоритма имеет указатель на класс (пункт 3.3), функцию генерации ключа, размер блока функции шифрования, размер чанка (считываемого блока данных из буфера) и идентификатора алгоритма. Формат предусматривает наличие до 256-ти записей в таблице алгоритмов. Таблица алгоритмов хэширования содержит функцию хэширования, длину хэша и идентификатор соответствующие алгоритму хэширования. Алгоритмов хэширования в таблице также может быть до 256-ти.

Использование модуля в целом сводится к вызову функций шифрования и дешифрования буферов при сохранении и чтении файла соответственно. Функция шифрования принимает на вход указатель на входной буфер, выходной буфер, длину входного буфера, символьный пароль и опционально алгоритм шифрования и алгоритм хэширования в соответствии с вышеописанными таблицами. По умолчанию используются алгоритмы Blowfish и MD5 для шифрования и хэширования соответственно. Функция дешифрования принимает на вход указатель на входной буфер с зашифрованными данными, выходной буфер для расшифрованных данных и символьный пароль.

Для генерации ключей по символьному паролю используется алгоритм PBKDF2. Алгоритм позволяет генерировать ключ произвольной длины по символьому паролю произвольной длины. То есть этот алгоритм можно использовать в связке с любым алгоритмом шифрования. Алгоритм позволяет “разбавлять” символьный пароль т. н. солью. Соль хранится вместе с зашифрованными данными. В работе была использована сторонняя реализация алгоритма, поэтому подробно его работа не рассмотрена.

[2] Буфер – (здесь) объект имеющий интерфейс, позволяющий производить операции чтения, записи, установки курсора и т. д. Буфером может быть открытый файл или строка в памяти.

3.5 Реализация графического интерфейса

Пакет Qt5 предлагает утилиту QtDesigner для проектирования графического интерфейса. Но в качестве альтернативы интерфейс может быть описан в коде программы без использования QtDesigner. Именно так и было сделано, поскольку в этом есть ряд преимуществ для автора работы.

Описание графического интерфейса представляет собой несколько Python-скриптов с переназначением некоторых базовых классов библиотеки Qt5 и набора методов:

1. `interte.py`

   Скрипт содержит переопредление класса виджета QMainWindow – главного окна программы (рисунок 3.1), и набор методов, реализующих основные функции типичного текстового редактора.

2. `mwidgetste.py`

   Скрипт содержит дополнительные переопределенные виджеты, а также описание окна настроек программы (рисунки 3.2-3.3).

3. `defaultvalues.py`

   Скрипт содержит определения констант.

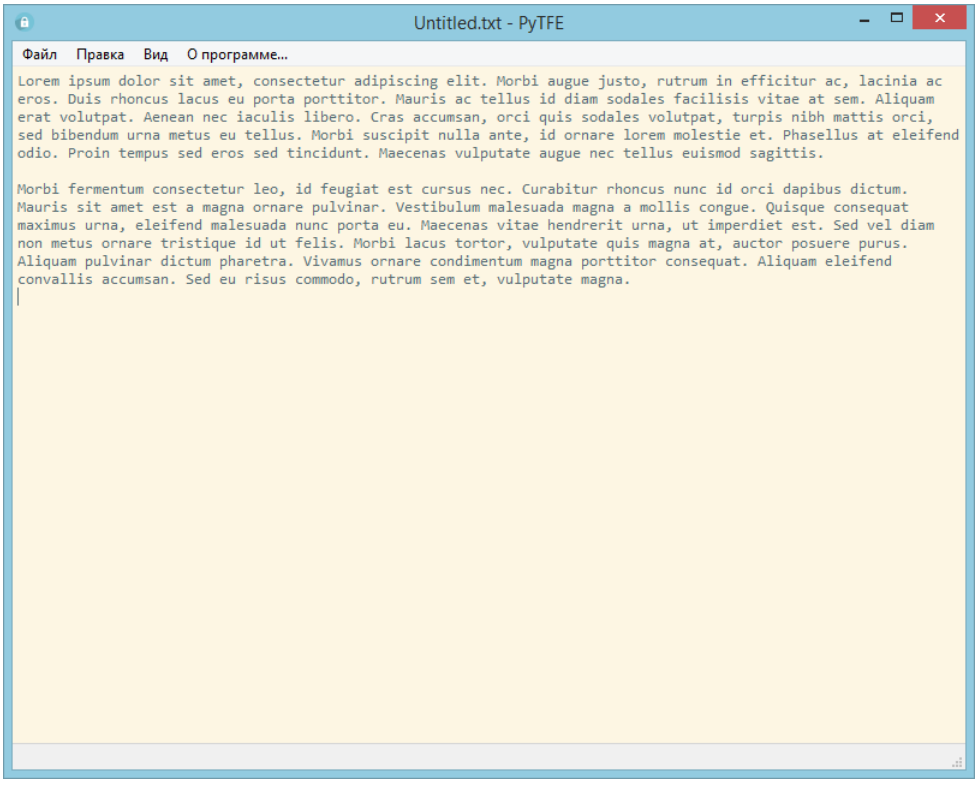

Рисунок 3.1 – Вид главного окна программы

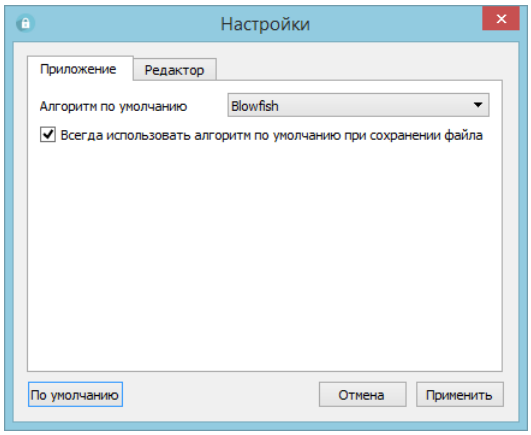

Рисунок 3.2 – Вид окна настроек: основные настройки

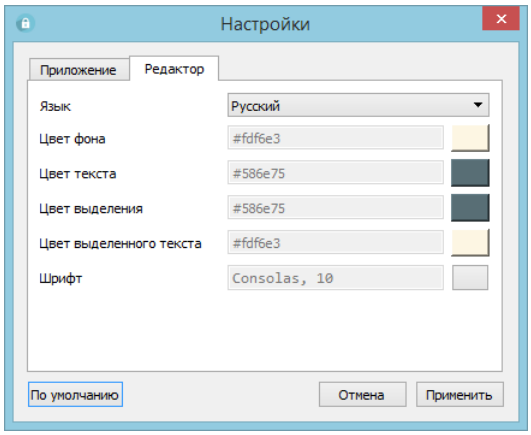

Рисунок 3.3 – Вид окна настроек: настройки редактора

3.6 Перевод приложения

Библиотека Qt5 имеет средство для интернационализации (перевода) приложений. С помощью утилиты `lupdate` создаются файлы локализаций, которые содержат перевод строк в графическом интерфейсе в формате XML. Редактирование файлов локализаций (перевод приложения) осуществляется утилитой `linguist`. Утилита `lrelease` позволяет “скомпилировать” файлы локализаций для использования в готовом приложении.

В разработанной программе имеется поддержка русского (рисунок 3.4) и английского (рисунок 3.5) языка.

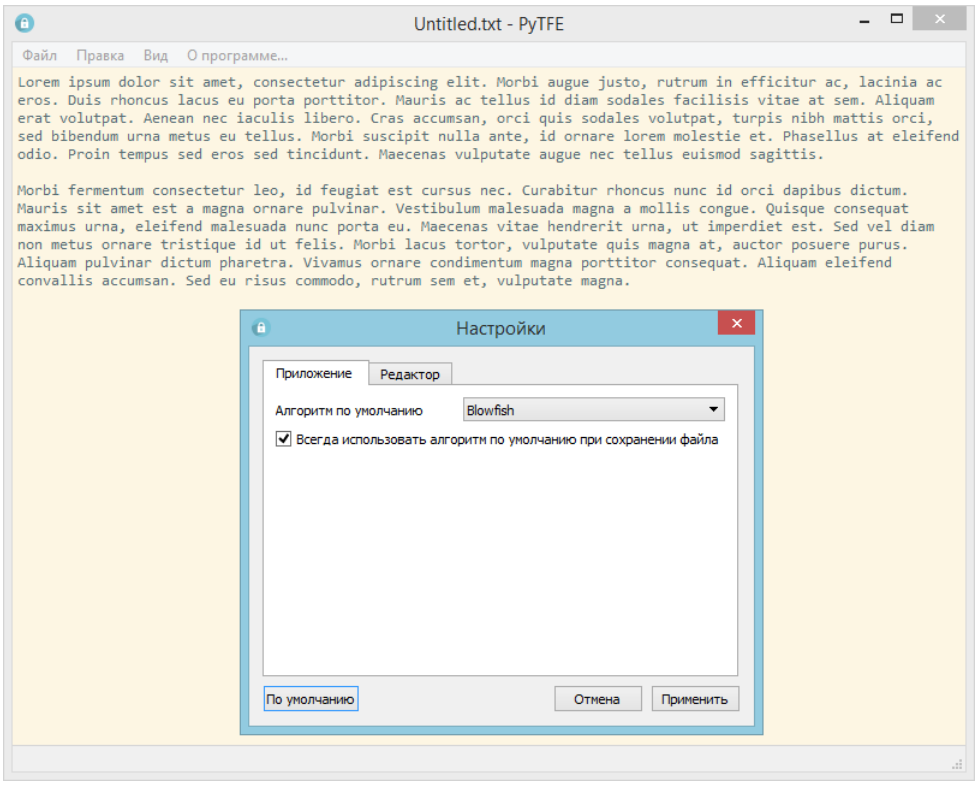

Рисунок 3.4 – Русский графический интерфейс

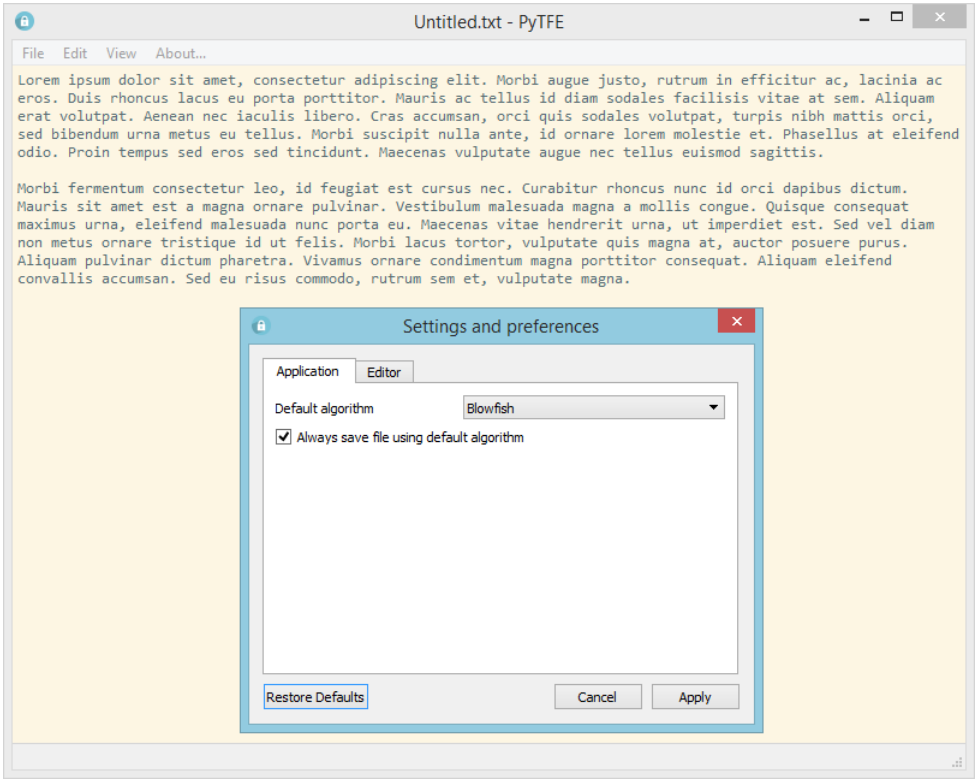

Рисунок 3.5 – Английский графический интерфейс

Заключение

При выполнении работы было спроектировано и реализовано программное средство, позволяющее пользователям работать с зашифрованными текстовыми файлами.

К достоинствам разработанной программы можно отнести:

– Простоту пользовательского интерфейса;

– Достаточный функционал;

– Наличие русского и английского переводов.

К недостаткам относятся:

– Снижение скорости шифрования из-за частичного использования Python в модулях шифрования;

– Использование метода шифрования EBC;

Основываясь на результатах работы можно сделать следующие выводы:

– Использование ассемблерного языка замедляет процесс написания программного кода и усложняет отладку программы.

– При использовании языка C упрощается процесс разработки и отладки программы в сравнении с языком ассемблера. Таким образом, язык C удобен для написания участков кода, от которых требуется высокая скорость выполнения.

– Использование интерпретируемого языка Python целесообразно для написания участков кода, от которых не требуется высокая скорость выполнения. Python значительно упрощает разработку и отладку программы в сравнении с языком C.

Список источников

[1] Лозовюк А. GnuPG – OpenSource шифрование и цифровые подписи. [Электронный ресурс]. – Режим доступа: http://citforum.ru/security/cryptography/gnupg/, свободный.

[2] Документация по Python 3.4.3 [Электронный ресурс], 1990-2015. – Режим доступа: https://docs.python.org/3.4/, свободный.

[3] Документация по Qt 5.5 [Электронный ресурс], 2015. – Режим доступа: http://doc.qt.io/qt-5/, свободный.

[4] Панасенко С. П. Алгоритмы шифрования. Специальный справочник. – СПб.: БХВ-Петербург, 2009. – 576 с.: ил.

[5] Matsui M., Yamagishi A. A New Method for Known Plaintext Attack of FEAL Cipher. [Электронный ресурс], 1992. – Режим доступа: http://link.springer.com/content/pdf/10.1007%2F3-540-47555-9_7.pdf

[6] Schneier, B. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish) [Электронный ресурс]. – Режим доступа: https://www.schneier.com/cryptography/archives/1994/09/description_of_a_new.html, свободный.

ПРИЛОЖЕНИЕ А

Реализация FEAL-4 на языке C – `feal4.c`

```
/*
 * FEAL4 algorithm implementation
 * Author:   Gleb Getmanenko
 * Date:     06.05.16
 * Language: C
 *
 * win32:
 * gcc -std=c99 -O3 -c feal4.c
 * gcc -s -shared -o feal4.dll feal4.o -Wl,--out-implib,libfeal4.a,--subsystem,windows
 *
 */

#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>
#include <windows.h>

void __stdcall __declspec(dllexport)
    _EncryptChunk( uint8_t* chunkptr, uint32_t nblocks, uint8_t* key);
void __stdcall __declspec(dllexport)
    _DecryptChunk( uint8_t* chunkptr, uint32_t nblocks, uint8_t* key);
void __stdcall __declspec(dllexport)
    _encrypt( uint8_t* data, uint8_t* key );
void __stdcall __declspec(dllexport)
    _decrypt( uint8_t* data, uint8_t* key );

void encrypt( uint8_t* data, uint8_t* key );
void decrypt( uint8_t* data, uint8_t* key );
uint32_t F( uint32_t sd, uint16_t sk );
uint8_t S0( uint8_t a, uint8_t b );
uint8_t S1( uint8_t a, uint8_t b );
uint8_t rol2( uint8_t a );


bool __stdcall __declspec(dllexport) APIENTRY
    DllMain( HANDLE hModule, DWORD fdwReason, LPVOID lpvReserved )
{
    return TRUE;
}

void __stdcall __declspec(dllexport)
    _EncryptChunk( uint8_t* chunkptr, uint32_t nblocks, uint8_t* key)
{
    for ( uint32_t i = 0; i < nblocks; i++ ) encrypt(&chunkptr[i*8],key);
}

void __stdcall __declspec(dllexport)
    _DecryptChunk( uint8_t* chunkptr , uint32_t nblocks , uint8_t* key)
```

```
{
    for ( uint32_t i = 0; i < nblocks; i++ ) decrypt(&chunkptr[i*8],key);
}

void __stdcall __declspec(dllexport)
    _encrypt( uint8_t* data , uint8_t* key )
{
    encrypt(data,key);
}

void __stdcall __declspec(dllexport)
    _decrypt( uint8_t* data , uint8_t* key )
{
    decrypt(data,key);
}


uint8_t S0( uint8_t a, uint8_t b ) { return rol2(a+b); }
uint8_t S1( uint8_t a, uint8_t b ) { return rol2(a+b+1); }
uint8_t rol2( uint8_t a ) { return ((a>>6)|(a<<2)); }

void encrypt( uint8_t* data , uint8_t* key )
{
    uint32_t L = ((uint32_t*)data)[1];
    uint32_t R = ((uint32_t*)data)[0];
    uint32_t t;

    L ^= ((uint32_t*)key)[1];
    R ^= ((uint32_t*)key)[0];
    R ^= L;
    for ( uint8_t i = 3; i > 0; i-- )
    {
        t = R;
        R = L ^ F(R,((uint16_t*)key)[i]);
        L = t;
    }
    L = L ^ F(R,((uint16_t*)key)[0]);
    R ^= L;
    L ^= ((uint32_t*)key)[1];
    R ^= ((uint32_t*)key)[0];

    ((uint32_t*)data)[1] = L;
    ((uint32_t*)data)[0] = R;
}

void decrypt( uint8_t* data , uint8_t* key )
{
    uint32_t L = ((uint32_t*)data)[1];
    uint32_t R = ((uint32_t*)data)[0];
    uint32_t t;

    L ^= ((uint32_t*)key)[1];
    R ^= ((uint32_t*)key)[0];
```

```
    R ^= L;
    for ( uint8_t i = 0; i < 3; i++ )
    {
        t = R;
        R = L ^ F(R,((uint16_t*)key)[i]);
        L = t;
    }
    L = L ^ F(R,((uint16_t*)key)[3]);
    R ^= L;
    L ^= ((uint32_t*)key)[1];
    R ^= ((uint32_t*)key)[0];

    ((uint32_t*)data)[1] = L;
    ((uint32_t*)data)[0] = R;
}

uint32_t F( uint32_t sd , uint16_t sk )
{
    uint8_t* ssd = (uint8_t*)&sd;
    uint8_t* ssk = (uint8_t*)&sk;

    ssd[1] ^= ssk[0];
    ssd[2] ^= ssk[1];
    ssd[1] ^= ssd[0];
    ssd[2] ^= ssd[3];

    ssd[1] = S1(ssd[1],ssd[2]);
    ssd[2] = S0(ssd[2],ssd[1]);
    ssd[0] = S0(ssd[0],ssd[1]);
    ssd[3] = S1(ssd[3],ssd[2]);

    return sd;
}
```

Реализация FEAL-4 на языке ассемблера – feal4.asm

```
format PE GUI 4.0 DLL
entry DllMain

include 'win32a.inc'

; ------------------------------

section '.text' code readable executable

proc DllMain hinstDLL, fdwReason, lpvReserved
    mov eax, TRUE
    ret
endp

proc EncryptChunk chunkptr,nblocks,keyptr
    pusha
    mov edi,[chunkptr]
    mov esi,[keyptr]
    mov ecx,[nblocks]
    @@:
        push ecx ; save ecx
        push edi ; save edi
        push esi ; save esi
        push dword [esi+4]    ; key[63:32]
        push dword [esi]      ; key[31:0]
        push dword [edi+4]    ; block[63:32]
        push dword [edi]      ; block[31:0]

        call _encrypt

        add esp,4*4
        pop esi ; restore esi
        pop edi ; restore edi
        pop ecx ; restore ecx
        mov [edi+4],edx ; save encrypted block[63:32]
        mov [edi],eax   ; save encrypted block[31:0]
        add edi,8
    loop @b
    popa
    mov eax,0
    ret
endp

proc DecryptChunk chunkptr,nblocks,keyptr
    pusha
    mov edi,[chunkptr]
    mov esi,[keyptr]
    mov ecx,[nblocks]
    @@:
        push ecx ; save ecx
        push edi ; save edi
```

```
        push esi ; save esi
        push dword [esi+4]    ; key[63:32]
        push dword [esi]      ; key[31:0]
        push dword [edi+4]    ; block[63:32]
        push dword [edi]      ; block[31:0]

        call _decrypt

        add esp,4*4
        pop esi ; restore esi
        pop edi ; restore edi
        pop ecx ; restore ecx
        mov [edi+4],edx ; save decrypted block[63:32]
        mov [edi],eax   ; save decrypted block[31:0]
        add edi,8
    loop @b
    popa
    mov eax,0
    ret
endp

proc encrypt dataptr,keyptr,resptr
    mov edi,[keyptr]
    push dword [edi+4]
    push dword [edi]
    mov edi,[dataptr]
    push dword [edi+4]
    push dword [edi]
    call _encrypt

    mov edi,[resptr]
    mov dword [edi],eax
    mov dword [edi+4],edx
    mov eax,0
    ret
endp

proc decrypt dataptr,keyptr,resptr
    mov edi,[keyptr]
    push dword [edi+4]
    push dword [edi]
    mov edi,[dataptr]
    push dword [edi+4]
    push dword [edi]
    call _decrypt

    mov edi,[resptr]
    mov dword [edi],eax
    mov dword [edi+4],edx
    mov eax,0
    ret
endp
```

```
F:
    ; sdata : eax     : bh:bl:ah:al
    ; key16 : cx
    push ebp
    mov ebp, esp

    ; eax -> bh:bl:ah:al
    ror eax,16
    mov bx,ax
    rol eax,16

    xor ah, cl
    xor bl, ch
    xor ah, al
    xor bl, bh

    add ah, bl
    add ah, 1
    rol ah, 2

    add bl, ah
    rol bl, 2

    add al, ah
    rol al, 2

    add bh, bl
    add bh, 1
    rol bh, 2

    ; bh:bl:ah:al -> eax
    ror eax,16
    mov ax,bx
    rol eax,16

    pop ebp
    ret
_encrypt:
    ; data : [ebp+8]  : edx:eax
    ; key  : [ebp+16]
    push ebp
    mov ebp, esp

    ; load data to edx:eax
    mov eax, [ebp+8]  ; right data
    mov edx, [ebp+12] ; left data

    ; data xor key
    xor eax, [ebp+16]
    xor edx, [ebp+20]

    ; left xor right
    xor eax, edx
```

```
    repeat 3 ;;;
        push eax ; temp right
        mov cx, word [ebp+(22-(%-1)*2)] ; key16
        call F ; F
        xor eax, edx ; RIGHT = F(right,key16) xor left
        pop edx ; LEFT = temp right
    end repeat ;;;

    push eax ; temp right
    mov cx, word [ebp+16] ; key16
    call F
    xor edx, eax ; LEFT = F(right,key16) xor left
    pop eax ; RIGHT = temp right

    ; left xor right
    xor eax, edx

    ; data xor key
    xor eax, [ebp+16]
    xor edx, [ebp+20]

    pop ebp
    ret
_decrypt:
    ; data : [ebp+8]  : edx:eax
    ; key  : [ebp+16]
    push ebp
    mov ebp, esp

    ; load data to edx:eax
    mov eax, [ebp+8]  ; right data
    mov edx, [ebp+12] ; left data

    ; data xor key
    xor eax, [ebp+16]
    xor edx, [ebp+20]

    ; left xor right
    xor eax, edx

    repeat 3 ;;;
        push eax ; temp right
        mov cx, word [ebp+(16+(%-1)*2)] ; key16
        call F ; F
        xor eax, edx ; RIGHT = F(right,key16) xor left
        pop edx ; LEFT = temp right
    end repeat ;;;

    push eax ; temp right
    mov cx, word [ebp+22] ; key16
    call F
    xor edx, eax ; LEFT = F(right,key16) xor left
```

```
    pop eax ; RIGHT = temp right

    ; left xor right
    xor eax, edx

    ; data xor key
    xor eax, [ebp+16]
    xor edx, [ebp+20]

    pop ebp
    ret

mov eax,EncryptChunk ; force relocation

section '.edata' export data readable

export 'FEAL4.DLL',\
    encrypt, 'encrypt',\
    EncryptChunk, 'EncryptChunk',\
    decrypt, 'decrypt',\
    DecryptChunk, 'DecryptChunk'

section '.reloc' fixups data readable discardable
```

Реализация Blowfish на языке C – `blowfish.c`

```
/*
 * Blowfish algorithm implementation
 * Author:   Gleb Getmanenko
 * Date:     02.05.16
 * Language: C
 *
 * win32:
 * gcc -std=c99 -O3 -c feal4.c
 * gcc -s -shared -o feal4.dll feal4.o -Wl,--out-implib,libfeal4.a,--subsystem,windows
 *
 */

#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>
#include <windows.h>

#include "blowfishPS.c"

typedef struct {
    uint32_t P[18];
    uint32_t S[4][256];
} KEY;

void __stdcall __declspec(dllexport)
    _EncryptChunk( uint8_t* chunkptr , uint32_t nblocks , KEY* key);
void __stdcall __declspec(dllexport)
    _DecryptChunk( uint8_t* chunkptr , uint32_t nblocks , KEY* key);
KEY* __stdcall __declspec(dllexport) _gen_key192( uint8_t* skey );
KEY* __stdcall __declspec(dllexport) __gen_key192( uint8_t* skey , KEY* key );

void encrypt( uint8_t* data , KEY* key );
void decrypt( uint8_t* data , KEY* key );
uint32_t F( uint32_t R, KEY* key );

void __stdcall _EncryptChunk( uint8_t* chunkptr , uint32_t nblocks , KEY* key)
{
    for ( uint32_t i = 0; i < nblocks; i++ )
    {
        encrypt(&chunkptr[i*8],key);
    }
}

void __stdcall _DecryptChunk( uint8_t* chunkptr , uint32_t nblocks , KEY* key)
{
    uint8_t* data = (uint8_t*)malloc(8);
    for ( uint32_t i = 0; i < nblocks; i++ )
    {
        decrypt(&chunkptr[i*8],key);
    }
}
```

```
void encrypt( uint8_t* data , KEY* key )
{
    uint32_t L = ((uint32_t*)data)[1];
    uint32_t R = ((uint32_t*)data)[0];
    uint32_t t;

    for ( uint8_t i = 0; i < 16; i++ )
    {
        t = R;
        R = L ^ key->P[i];
        L = t ^ F(R,key);
    }

    t = L ^ key->P[16];
    L = R ^ key->P[17];
    R = t;

    ((uint32_t*)data)[1] = L;
    ((uint32_t*)data)[0] = R;
}

void decrypt( uint8_t* data , KEY* key )
{
    uint32_t L = ((uint32_t*)data)[1];
    uint32_t R = ((uint32_t*)data)[0];
    uint32_t t;

    for ( uint8_t i = 17; i > 1; i-- )
    {
        t = R;
        R = L ^ key->P[i];
        L = t ^ F(R,key);
    }

    t = L ^ key->P[1];
    L = R ^ key->P[0];
    R = t;

    ((uint32_t*)data)[1] = L;
    ((uint32_t*)data)[0] = R;
}

uint32_t F(uint32_t R, KEY* key)
{
    uint8_t* r = (uint8_t*)&R;
    uint32_t q[4];

    q[0] = key->S[3][ r[0] ];
    q[1] = key->S[2][ r[1] ];
    q[2] = key->S[1][ r[2] ];
    q[3] = key->S[0][ r[3] ];
```

```
    q[2] += q[3];
    q[1] ^= q[2];
    q[0] += q[1];

    return q[0];
}

KEY* __stdcall _gen_key192( uint8_t* skey )
{
    KEY* key = (KEY*)malloc(sizeof(KEY));
    __gen_key192(skey,key);
    return key;
}

KEY* __stdcall __declspec(dllexport) __gen_key192( uint8_t* skey , KEY* key )
{
    /* Function generates P keys and S tables from a 192 bit secret key */
    uint32_t* skey32 = (uint32_t*)skey;

    // Initialize P and S
    memcpy((void*)key->P,(void*)&P_INIT,sizeof(uint32_t)*18);
    memcpy((void*)key->S,(void*)&S_INIT,sizeof(uint32_t)*4*256);

    // XOR P keys with secret key
    for ( int i = 0; i < 18; i++ ) key->P[i] ^= skey32[i%6];

    // Encipher P keys and S tables
    uint8_t* data = (uint8_t*)malloc(8);
    uint32_t* data32 = (uint32_t*)data;
    memset(data,0,8);

    for ( int i = 0; i < 18; i+=2 )
    {
        encrypt(data,key);
        key->P[i] = data32[0];
        key->P[i+1] = data32[1];
    }
    for ( int i = 0; i < 4; i++ )
    {
        for ( int j = 0; j < 256; j+=2 )
        {
            encrypt(data,key);
            key->S[i][j] = data32[0];
            key->S[i][j+1] = data32[1];
        }
    }

    return key;
}

bool __stdcall __declspec(dllexport) APIENTRY DllMain(
    HANDLE hModule, DWORD fdwReason, LPVOID lpvReserved
)
```

```
{
    return TRUE;
}
```

Реализация Blowfish на языке ассемблера – `blowfish.asm`

```
;
; Blowfish algorithm implementation
; Author:   Gleb Getmanenko
; Date:     02.05.16
; Language: asm (fasm)
;

format PE GUI 4.0 DLL
entry DllMain

include 'win32a.inc'

include 'blowfishPS.asm'

struct KEY
    P  dd 18  dup (?)
    S1 dd 256 dup (?)
    S2 dd 256 dup (?)
    S3 dd 256 dup (?)
    S4 dd 256 dup (?)
ends

section '.text' code readable executable

proc DllMain hinstDLL, fdwReason, lpvReserved
    mov eax, TRUE
    ret
endp

encrypt:
    ; encrypt a block of data
    ; [esp+4] - pointer to data
    ; [esp+8] - pointer to key

    mov edi, [esp+4]
    mov eax, [edi+0]
    mov edx, [edi+4]

    mov edi, [esp+8]

    repeat 16  ;;;
        push eax
        mov eax, edx
        xor eax, [edi+KEY.P+(%-1)*4]
        pop edx
        call F
    end repeat ;;;

    mov ebx, edx
    xor ebx, [edi+KEY.P+16*4]
    mov edx, eax
```

```
    xor edx, [edi+KEY.P+17*4]
    mov eax, ebx

    mov edi, [esp+4]
    mov [edi+0], eax
    mov [edi+4], edx

    ret

decrypt:
    ; decrypt a block of data
    ; [esp+4] - pointer to data
    ; [esp+8] - pointer to key

    mov edi, [esp+4]
    mov eax, [edi+0]
    mov edx, [edi+4]

    mov edi, [esp+8]

    repeat 16  ;;;
        push eax
        mov eax, edx
        xor eax, [edi+KEY.P+(17-(%-1))*4]
        pop edx
        call F
    end repeat ;;;

    mov ebx, edx
    xor ebx, [edi+KEY.P+1*4]
    mov edx, eax
    xor edx, [edi+KEY.P+0*4]
    mov eax, ebx

    mov edi, [esp+4]
    mov [edi+0], eax
    mov [edi+4], edx

    ret

F:
    push eax
    push edx

    movzx eax, byte [esp+4+0]
    movzx ebx, byte [esp+4+1]
    movzx ecx, byte [esp+4+2]
    movzx edx, byte [esp+4+3]

    mov eax, [edi+KEY.S4+eax*4]
    mov ebx, [edi+KEY.S3+ebx*4]
    mov ecx, [edi+KEY.S2+ecx*4]
    mov edx, [edi+KEY.S1+edx*4]
```

```
    add ecx, edx
    xor ebx, ecx
    add eax, ebx

    pop edx
    xor edx, eax
    pop eax
    ret

proc Encrypt dataptr,keyptr
    pusha
    pushd [keyptr]
    pushd [dataptr]
    call encrypt
    add esp,8
    popa
    mov eax,0
    ret
endp

proc Decrypt dataptr,keyptr
    pusha
    pushd [keyptr]
    pushd [dataptr]
    call decrypt
    add esp,8
    popa
    mov eax,0
    ret
endp

proc EncryptChunk chunkptr,nblocks,keyptr
    pushad

    mov edi, [chunkptr]
    ;mov esi, [keyptr]
    mov ecx, [nblocks]
    @@:
        push ecx      ; save counter
        push [keyptr] ; key ptr
        push edi      ; data ptr
        call encrypt
        add esp, 2*4
        pop ecx
        add edi, 8
    loop @b

    popad
    mov eax,0
    ret
endp
```

```
proc DecryptChunk chunkptr,nblocks,keyptr
    pushad

    mov edi, [chunkptr]
    ;mov esi, [keyptr]
    mov ecx, [nblocks]
    @@:
        push ecx      ; save counter
        push [keyptr] ; key ptr
        push edi      ; data ptr
        call decrypt
        add esp, 2*4
        pop ecx
        add edi, 8
    loop @b

    popad
    mov eax,0
    ret
endp

proc _gen_key192 skey,key
    pushad

    ; init P keys and S tables
    pushd 18*4+4*256*4 ; length of initial key
    push PS_INIT
    push [key]
    call mmemcpy
    add esp, 12

    ; cycled xor with skey (24 bytes)
    mov edi, [skey]
    mov esi, [key]
    mov ecx, 0
    @@:
        ; key->P[i] ^= skey32[i%6]
        mov eax, ecx
        ; eax = eax % 6
        ror eax, 16
        mov dx, ax
        rol eax, 16
        mov bx, 6
        div bx
        movsx eax, dx
        ;
        mov eax, [edi+4*eax]
        xor [esi+4*ecx], eax
        inc ecx
        cmp ecx, 18
    jnae @b

    ; encipher P keys and S tables
```

```
    ; init zero block
    pushd 0
    pushd 0
    mov eax, esp ; pointer to zero block
    ; for ( int i = 0; i < 1042; i+=2 )
    xor ecx, ecx ; int i = 0
    @@:
        push ecx
        ; encrypt(data,key);
        push esi ; key pointer
        push eax ; data pointer
        call encrypt

        ; update key
        pop eax ; data pointer
        pop esi ; key pointer
        pop ecx ; i
        mov ebx, [eax+0]
        mov [esi+ecx*4+0], ebx
        mov ebx, [eax+4]
        mov [esi+ecx*4+4], ebx

        add ecx, 2
        cmp ecx, 1042
    jnae @b
    add esp, 8

    popad
    mov eax, [key]
    ret
endp

mmemcpy:
    ; copy [ebp+16] bytes from *[ebp+12] to *[ebp+8]
    push ebp
    mov ebp, esp

    mov ecx, [ebp+16]
    cmp ecx, 0
    jz endloop
    mov edi, [ebp+12]
    mov esi, [ebp+8]
    @@:
        mov al, [edi]
        mov [esi], al
        inc edi
        inc esi
    loop @b
    endloop:

    mov esp, ebp
    pop ebp
    ret
```

```
mov eax, encrypt ; force relocation

section '.edata' export data readable

export 'blowfish.dll',\
    Encrypt, 'encrypt',\
    Decrypt, 'decrypt',\
    EncryptChunk, 'EncryptChunk',\
    DecryptChunk, 'DecryptChunk',\
    _gen_key192, '_gen_key192'

section '.reloc' fixups data readable discardable
```

ПРИЛОЖЕНИЕ Б

Связка динамической библиотеки FEAL-4 с Python – `feal4.py`

```
# -*- coding: utf-8 -*-

import sys
import os.path
from ctypes import *

class feal4:
    key = None
    # key length = 64 bits = 8 bytes
    keylength = 8
    # block length = 64 bits = 8 bytes
    blocksize = 8

    def __init__(self, key=None):
        if not isinstance(key, bytes):
            raise TypeError('Key must be bytes')
        if len(key) < self.keylength:
            raise ValueError('Key size must be at least %i bytes'%self.keylength)

        if getattr(sys, 'frozen', False):
            path = os.path.dirname(os.path.realpath(sys.executable))
        else:
            path = os.path.dirname(os.path.realpath(sys.argv[0]))
        libpath = os.path.join(path, 'mlibs', 'win32', 'feal4.dll')
        self.dll = WinDLL(libpath)

        self.key = create_string_buffer(key)

    def __del__(self):
        try:
            libHandle = self.dll._handle
            del self.dll
            windll.kernel32.FreeLibrary(libHandle)
        except:
            pass

    def Encrypt(self, data):
        _data = create_string_buffer(data)
        self.dll.Encrypt(_data, self.key)
        return _data.raw[:-1]

    def Decrypt(self, data):
        _data = create_string_buffer(data)
        self.dll.Decrypt(_data, self.key)
        return _data.raw[:-1]

    def EncryptChunk(self, chunk, clen):
        _chunk = create_string_buffer(chunk)
```

```
        _clen  = c_ulong(clen)
        self.dll.EncryptChunk(_chunk, _clen, self.key)
        return _chunk.raw[:-1]

    def DecryptChunk(self, chunk, clen):
        _chunk = create_string_buffer(chunk)
        _clen  = c_ulong(clen)
        self.dll.DecryptChunk(_chunk, _clen, self.key)
        return _chunk.raw[:-1]
```

Связка динамической библиотеки Blowfish с Python – `blowfish.py`

```
# -*- coding: utf-8 -*-

import sys
import os.path
from ctypes import *

class blowfish:
    key  = None
    skey = None
    # key length = (18 keys)x(32 bits) + (4 tables)x(256 values)x(32 bits)
    keylength  = 18*4 + 4*256*4
    # secret key length = 192 bits = 24 bytes
    skeylength = 192//8
    # block length = 64 bits = 8 bytes
    blocksize  = 8

    def __init__(self, skey=None):
        if not isinstance(skey, bytes):
            raise TypeError('Key must be bytes')
        if len(skey) < self.skeylength:
            raise ValueError('Key size must be at least %i bytes'%self.skeylength)

        if getattr(sys, 'frozen', False):
            path = os.path.dirname(os.path.realpath(sys.executable))
        else:
            path = os.path.dirname(os.path.realpath(sys.argv[0]))
        libpath = os.path.join(path, 'mlibs', 'win32', 'blowfish.dll')

        self.dll = WinDLL(libpath)

        self.skey = create_string_buffer(skey)
        self.key  = create_string_buffer(self.keylength)

        self.dll._gen_key192(self.skey, self.key)

    def __del__(self):
        try:
            del self.key
            libHandle = self.dll._handle
            del self.dll
            windll.kernel32.FreeLibrary(libHandle)
        except:
            pass

    def Encrypt(self, data):
        _data = create_string_buffer(data)
        self.dll.Encrypt(_data, self.key)
        return _data.raw[:-1]

    def Decrypt(self, data):
        _data = create_string_buffer(data)
```

```
        self.dll.Decrypt(_data, self.key)
        return _data.raw[:-1]

    def EncryptChunk(self, chunk, clen):
        _chunk = create_string_buffer(chunk)
        _clen  = c_ulong(clen)
        self.dll.EncryptChunk(_chunk, _clen, self.key)
        return _chunk.raw[:-1]

    def DecryptChunk(self, chunk, clen):
        _chunk = create_string_buffer(chunk)
        _clen  = c_ulong(clen)
        self.dll.DecryptChunk(_chunk, _clen, self.key)
        return _chunk.raw[:-1]
```

Модуль для шифрования – `tfe.py`

```
# -*- coding: utf-8 -*-

import os
import sys
import traceback

from hashlib import md5

from .pbkdf2 import PBKDF2 as pbkdf2

from .feal4      import feal4
from .blowfish   import blowfish

algtab = {
    "FEAL 4": {
        "module"    : feal4,
        "genkey"    : lambda pas, salt: pbkdf2(pas, salt).read(8),
        "blocksize" : 8,
        "chunksize" : 2*8*8*1024, # 2*64K
        "id"        : 1,
    },
    "Blowfish": {
        "module"    : blowfish,
        "genkey"    : lambda pas, salt: pbkdf2(pas, salt).read(24),
        "blocksize" : 8,
        "chunksize" : 2*8*8*1024, # 2*64K
        "id"        : 2,
    },
}

hashtab = {
    "MD5": {
        "function" : lambda x: md5(x).digest(),
        "id"       : 1,
        "length"   : 16,
    }
}

FILE_SIG = b'tfe\x42'
HEADER_LENGTH = 16
HASH_DATA_LENGTH = 512
SALT_LENGTH = 16

def buildheader(**kwargs):
    header = b''
    header += FILE_SIG
    flds = [ ('algorithm', 1), ('hashtype', 1), ('offset', 2), ('filesize', 8) ]
    for fld, n in flds:
        q = kwargs[fld]
        b = q.to_bytes(n, sys.byteorder)
        header += b
```

```
    if len(header)!=16: raise
    return header

def buildemptyheader(**kwargs):
    header = b''
    header += FILE_SIG
    flds = [ ('algorithm',1),('hashtype',1),('offset',2),('filesize',8) ]
    for fld,n in flds:
        q = kwargs[fld]
        b = q.to_bytes(n,sys.byteorder)
        header += b
    if len(header)!=16: raise
    return header

def parseheader(header):
    if header[0:4] != FILE_SIG: raise ValueError('Not valid file')
    hdr = []
    # flds = [ ('algorithm', 1), ('hashtype', 1), ('offset', 2), ('filesize', 8) ]
    algid    = header[4]
    hashid   = header[5]
    offset   = int.from_bytes(header[6:8], sys.byteorder)
    filesize = int.from_bytes(header[8:16], sys.byteorder)
    # find alg
    for a in algtab:
        if algtab[a]['id'] == algid: algorithm = a
    # find hash
    for h in hashtab:
        if hashtab[h]['id'] == hashid: hashtype = h
    return algorithm, hashtype, offset, filesize

def readchunk(bi, bls, chunksize):
    res = bi.read(chunksize)
    L = len(res)
    if L == 0: return b'',0
    if L%bls != 0:
        q = (bls - L%bls)%bls
        res = res + b'\x42'*q
    return res, len(res)//bls

def EncryptBuffer(bi, bo, blength, pas, alg="Blowfish", hasht="MD5"):
    # alg
    _alg = algtab[alg]
    _bls = _alg['blocksize']
    _chs = _alg['chunksize']
    salt = os.urandom(SALT_LENGTH)
    _key = _alg['genkey'](pas, salt)
    _enc = _alg['module'](_key)
    _algid = _alg['id']

    # hash
    _hashf   = hashtab[hasht]["function"]
    _hashid  = hashtab[hasht]["id"]
    _hashlen = hashtab[hasht]["length"]
```

```
    # header
    bleft = blength
    chunk,clen = readchunk(bi, _bls, min(_chs, bleft))
    bleft -= len(chunk)
    if clen == 0:
        # no data in buffer
        header = buildheader(
            algorithm = _algid,
            hashtype  = _hashid,
            offset    = 0,
            filesize  = 0
        )
        bo.write(header)
        del _enc
        return

    hash = _hashf(chunk[:HASH_DATA_LENGTH])
    _offset = HEADER_LENGTH + len(hash) + SALT_LENGTH
    header = buildheader(
        algorithm = _algid,
        hashtype  = _hashid,
        offset    = _offset,
        filesize  = blength
    )

    # write header and first chunk
    bo.write(header)
    bo.write(hash)
    bo.write(salt)
    echunk = _enc.EncryptChunk(chunk, clen)
    bo.write(echunk)

    # encrypt remaining chunks
    while bleft > 0:
        chunk, clen = readchunk(bi, _bls, min(_chs, bleft))
        bleft -= len(chunk)
        if clen == 0: break
        echunk = _enc.EncryptChunk(chunk, clen)
        bo.write(echunk)
    del _enc
    if bleft > 0:
        raise Exception('Buffer size specified is greater than actual data')

def DecryptBuffer(bi, bo, pas):
    BEG = bi.tell()
    header = bi.read(HEADER_LENGTH)
    alg, hasht, offset, blength = parseheader(header)
    if blength == 0: return

    # init hash
    _hashf   = hashtab[hasht]["function"]
    _hashid  = hashtab[hasht]["id"]
```

```
    _hashlen = hashtab[hasht]["length"]

    # read hash and salt
    bi.seek(BEG+HEADER_LENGTH, 0)
    hashA = bi.read(_hashlen)
    salt  = bi.read(SALT_LENGTH)
    bi.seek(BEG+offset, 0)

    # init alg
    _alg = algtab[alg]
    _bls = _alg['blocksize']
    _chs = _alg['chunksize']
    _key = _alg['genkey'](pas, salt)
    _dec = _alg['module'](_key)

    # decr
    bleft = blength + (_bls - blength%_bls)%_bls
    echunk, clen = readchunk(bi, _bls, min(_chs, bleft))
    bleft -= len(echunk)
    chunk = _dec.DecryptChunk(echunk, clen)

    # check hash
    hashD = _hashf( chunk[:HASH_DATA_LENGTH] )
    if hashA != hashD:
        raise ValueError('Bad key')
    bo.write(chunk[:blength])
    blength -= len(echunk)

    # decrypt remaining chunks
    while bleft > 0:
        echunk, clen = readchunk(bi, _bls, min(_chs, bleft))
        bleft -= len(echunk)
        if clen == 0: break
        chunk = _dec.DecryptChunk(echunk, clen)
        bo.write(chunk[:blength])
        blength -= len(chunk)
    del _dec

def isTfeFile(path):
    f = open(path,'rb')
    try:
        header = f.read(HEADER_LENGTH)
        if header == b'':
            return 'empty'
        alg, hasht, offset, blength = parseheader(header)
        if blength == 0:
            return 'empty'
        if (alg in algtab) and (hasht in hashtab):
            return 'ok'
        else:
            return 'not supported'
    except ValueError as e:
        traceback.print_exc()
```

```
            return 'not tfe'
        else:
            traceback.print_exc()
        finally:
            f.close()

    def whatAlgoIn(path):
        f = open(path,'rb')
        try:
            header = f.read(HEADER_LENGTH)
            alg, hasht, offset, blength = parseheader(header)
            return alg
        except:
            traceback.print_exc()
            #raise e
        finally:
            f.close()

    # ---------- ФУНКЦИИ ДЛЯ ФАЙЛОВ ----------

    def EncryptFile(alg, pas, filenameIN, filenameOUT=None, r=False):
        if filenameOUT == None:
            if r:
                filenameOUT = filenameIN + '$temp$'
            else:
                fn, ext = os.path.splitext(filenameIN)
                filenameOUT = fn + '_enc' + ext
        if filenameOUT == filenameIN:
            filenameOUT += '$temp$'
            r = True

        try:
            fi = open(filenameIN, 'rb')
        except Exception as e:
            raise e

        try:
            fo = open(filenameOUT, 'wb')
        except:
            fi.close()
            raise e

        filesize = os.path.getsize(filenameIN)

        try:
            EncryptBuffer(fi, fo, filesize, pas, alg)
        except Exception as e:
            fi.close()
            fo.close()
            os.remove(filenameOUT)
            raise e
        fi.close()
        fo.close()
```

```
    if r:
        os.remove(filenameIN)
        os.rename(filenameOUT, filenameIN)

def DecryptFile(pas, filenameIN, filenameOUT=None, r=False):
    if filenameOUT == None:
        if r:
            filenameOUT = filenameIN + '$temp$'
        else:
            fn, ext = os.path.splitext(filenameIN)
            filenameOUT = fn + '_dec' + ext
    if filenameOUT == filenameIN:
        filenameOUT += '$temp$'
        r = True

    try:
        fi = open(filenameIN, 'rb')
    except Exception as e:
        raise e

    try:
        fo = open(filenameOUT, 'wb')
    except:
        fi.close()
        raise e

    try:
        DecryptBuffer(fi,fo,pas)
    except Exception as e:
        fi.close()
        fo.close()
        os.remove(filenameOUT)
        raise e

    fi.close()
    fo.close()
    if r:
        os.remove(filenameIN)
        os.rename(filenameOUT, filenameIN)
```

ПРИЛОЖЕНИЕ В

Начальные значения ключей и таблиц замены алгоритма Blowfish

```
uint32_t P_INIT[18] = {
    0x243F6A88, 0x85A308D3, 0x13198A2E, 0x03707344, 0xA4093822,
    0x299F31D0, 0x082EFA98, 0xEC4E6C89, 0x452821E6, 0x38D01377,
    0xBE5466CF, 0x34E90C6C, 0xC0AC29B7, 0xC97C50DD, 0x3F84D5B5,
    0xB5470917, 0x9216D5D9, 0x8979FB1B
};

uint32_t S_INIT[4][256] = {
    {
        0xD1310BA6, 0x98DFB5AC, 0x2FFD72DB, 0xD01ADFB7,
        0xB8E1AFED, 0x6A267E96, 0xBA7C9045, 0xF12C7F99,
        0x24A19947, 0xB3916CF7, 0x0801F2E2, 0x858EFC16,
        0x636920D8, 0x71574E69, 0xA458FEA3, 0xF4933D7E,
        0x0D95748F, 0x728EB658, 0x718BCD58, 0x82154AEE,
        0x7B54A41D, 0xC25A59B5, 0x9C30D539, 0x2AF26013,
        0xC5D1B023, 0x286085F0, 0xCA417918, 0xB8DB38EF,
        0x8E79DCB0, 0x603A180E, 0x6C9E0E8B, 0xB01E8A3E,
        0xD71577C1, 0xBD314B27, 0x78AF2FDA, 0x55605C60,
        0xE65525F3, 0xAA55AB94, 0x57489862, 0x63E81440,
        0x55CA396A, 0x2AAB10B6, 0xB4CC5C34, 0x1141E8CE,
        0xA15486AF, 0x7C72E993, 0xB3EE1411, 0x636FBC2A,
        0x2BA9C55D, 0x741831F6, 0xCE5C3E16, 0x9B87931E,
        0xAFD6BA33, 0x6C24CF5C, 0x7A325381, 0x28958677,
        0x3B8F4898, 0x6B4BB9AF, 0xC4BFE81B, 0x66282193,
        0x61D809CC, 0xFB21A991, 0x487CAC60, 0x5DEC8032,
        0xEF845D5D, 0xE98575B1, 0xDC262302, 0xEB651B88,
        0x23893E81, 0xD396ACC5, 0x0F6D6FF3, 0x83F44239,
        0x2E0B4482, 0xA4842004, 0x69C8F04A, 0x9E1F9B5E,
        0x21C66842, 0xF6E96C9A, 0x670C9C61, 0xABD388F0,
        0x6A51A0D2, 0xD8542F68, 0x960FA728, 0xAB5133A3,
        0x6EEF0B6C, 0x137A3BE4, 0xBA3BF050, 0x7EFB2A98,
        0xA1F1651D, 0x39AF0176, 0x66CA593E, 0x82430E88,
        0x8CEE8619, 0x456F9FB4, 0x7D84A5C3, 0x3B8B5EBE,
        0xE06F75D8, 0x85C12073, 0x401A449F, 0x56C16AA6,
        0x4ED3AA62, 0x363F7706, 0x1BFEDF72, 0x429B023D,
        0x37D0D724, 0xD00A1248, 0xDB0FEAD3, 0x49F1C09B,
        0x075372C9, 0x80991B7B, 0x25D479D8, 0xF6E8DEF7,
        0xE3FE501A, 0xB6794C3B, 0x976CE0BD, 0x04C006BA,
        0xC1A94FB6, 0x409F60C4, 0x5E5C9EC2, 0x196A2463,
        0x68FB6FAF, 0x3E6C53B5, 0x1339B2EB, 0x3B52EC6F,
        0x6DFC511F, 0x9B30952C, 0xCC814544, 0xAF5EBD09,
        0xBEE3D004, 0xDE334AFD, 0x660F2807, 0x192E4BB3,
        0xC0CBA857, 0x45C8740F, 0xD20B5F39, 0xB9D3FBDB,
        0x5579C0BD, 0x1A60320A, 0xD6A100C6, 0x402C7279,
        0x679F25FE, 0xFB1FA3CC, 0x8EA5E9F8, 0xDB3222F8,
        0x3C7516DF, 0xFD616B15, 0x2F501EC8, 0xAD0552AB,
        0x323DB5FA, 0xFD238760, 0x53317B48, 0x3E00DF82,
        0x9E5C57BB, 0xCA6F8CA0, 0x1A87562E, 0xDF1769DB,
```

```
    0xD542A8F6, 0x287EFFC3, 0xAC6732C6, 0x8C4F5573,
    0x695B27B0, 0xBBCA58C8, 0xE1FFA35D, 0xB8F011A0,
    0x10FA3D98, 0xFD2183B8, 0x4AFCB56C, 0x2DD1D35B,
    0x9A53E479, 0xB6F84565, 0xD28E49BC, 0x4BFB9790,
    0xE1DDF2DA, 0xA4CB7E33, 0x62FB1341, 0xCEE4C6E8,
    0xEF20CADA, 0x36774C01, 0xD07E9EFE, 0x2BF11FB4,
    0x95DBDA4D, 0xAE909198, 0xEAAD8E71, 0x6B93D5A0,
    0xD08ED1D0, 0xAFC725E0, 0x8E3C5B2F, 0x8E7594B7,
    0x8FF6E2FB, 0xF2122B64, 0x8888B812, 0x900DF01C,
    0x4FAD5EA0, 0x688FC31C, 0xD1CFF191, 0xB3A8C1AD,
    0x2F2F2218, 0xBE0E1777, 0xEA752DFE, 0x8B021FA1,
    0xE5A0CC0F, 0xB56F74E8, 0x18ACF3D6, 0xCE89E299,
    0xB4A84FE0, 0xFD13E0B7, 0x7CC43B81, 0xD2ADA8D9,
    0x165FA266, 0x80957705, 0x93CC7314, 0x211A1477,
    0xE6AD2065, 0x77B5FA86, 0xC75442F5, 0xFB9D35CF,
    0xEBCDAF0C, 0x7B3E89A0, 0xD6411BD3, 0xAE1E7E49,
    0x00250E2D, 0x2071B35E, 0x226800BB, 0x57B8E0AF,
    0x2464369B, 0xF009B91E, 0x5563911D, 0x59DFA6AA,
    0x78C14389, 0xD95A537F, 0x207D5BA2, 0x02E5B9C5,
    0x83260376, 0x6295CFA9, 0x11C81968, 0x4E734A41,
    0xB3472DCA, 0x7B14A94A, 0x1B510052, 0x9A532915,
    0xD60F573F, 0xBC9BC6E4, 0x2B60A476, 0x81E67400,
    0x08BA6FB5, 0x571BE91F, 0xF296EC6B, 0x2A0DD915,
    0xB6636521, 0xE7B9F9B6, 0xFF34052E, 0xC5855664,
    0x53B02D5D, 0xA99F8FA1, 0x08BA4799, 0x6E85076A
},
{
    0x4B7A70E9, 0xB5B32944, 0xDB75092E, 0xC4192623,
    0xAD6EA6B0, 0x49A7DF7D, 0x9CEE60B8, 0x8FEDB266,
    0xECAA8C71, 0x699A17FF, 0x5664526C, 0xC2B19EE1,
    0x193602A5, 0x75094C29, 0xA0591340, 0xE4183A3E,
    0x3F54989A, 0x5B429D65, 0x6B8FE4D6, 0x99F73FD6,
    0xA1D29C07, 0xEFE830F5, 0x4D2D38E6, 0xF0255DC1,
    0x4CDD2086, 0x8470EB26, 0x6382E9C6, 0x021ECC5E,
    0x09686B3F, 0x3EBAEFC9, 0x3C971814, 0x6B6A70A1,
    0x687F3584, 0x52A0E286, 0xB79C5305, 0xAA500737,
    0x3E07841C, 0x7FDEAE5C, 0x8E7D44EC, 0x5716F2B8,
    0xB03ADA37, 0xF0500C0D, 0xF01C1F04, 0x0200B3FF,
    0xAE0CF51A, 0x3CB574B2, 0x25837A58, 0xDC0921BD,
    0xD19113F9, 0x7CA92FF6, 0x94324773, 0x22F54701,
    0x3AE5E581, 0x37C2DADC, 0xC8B57634, 0x9AF3DDA7,
    0xA9446146, 0x0FD0030E, 0xECC8C73E, 0xA4751E41,
    0xE238CD99, 0x3BEA0E2F, 0x3280BBA1, 0x183EB331,
    0x4E548B38, 0x4F6DB908, 0x6F420D03, 0xF60A04BF,
    0x2CB81290, 0x24977C79, 0x5679B072, 0xBCAF89AF,
    0xDE9A771F, 0xD9930810, 0xB38BAE12, 0xDCCF3F2E,
    0x5512721F, 0x2E6B7124, 0x501ADDE6, 0x9F84CD87,
    0x7A584718, 0x7408DA17, 0xBC9F9ABC, 0xE94B7D8C,
    0xEC7AEC3A, 0xDB851DFA, 0x63094366, 0xC464C3D2,
    0xEF1C1847, 0x3215D908, 0xDD433B37, 0x24C2BA16,
    0x12A14D43, 0x2A65C451, 0x50940002, 0x133AE4DD,
    0x71DFF89E, 0x10314E55, 0x81AC77D6, 0x5F11199B,
    0x043556F1, 0xD7A3C76B, 0x3C11183B, 0x5924A509,
```

```
    0xF28FE6ED, 0x97F1FBFA, 0x9EBABF2C, 0x1E153C6E,
    0x86E34570, 0xEAE96FB1, 0x860E5E0A, 0x5A3E2AB3,
    0x771FE71C, 0x4E3D06FA, 0x2965DCB9, 0x99E71D0F,
    0x803E89D6, 0x5266C825, 0x2E4CC978, 0x9C10B36A,
    0xC6150EBA, 0x94E2EA78, 0xA5FC3C53, 0x1E0A2DF4,
    0xF2F74EA7, 0x361D2B3D, 0x1939260F, 0x19C27960,
    0x5223A708, 0xF71312B6, 0xEBADFE6E, 0xEAC31F66,
    0xE3BC4595, 0xA67BC883, 0xB17F37D1, 0x018CFF28,
    0xC332DDEF, 0xBE6C5AA5, 0x65582185, 0x68AB9802,
    0xEECEA50F, 0xDB2F953B, 0x2AEF7DAD, 0x5B6E2F84,
    0x1521B628, 0x29076170, 0xECDD4775, 0x619F1510,
    0x13CCA830, 0xEB61BD96, 0x0334FE1E, 0xAA0363CF,
    0xB5735C90, 0x4C70A239, 0xD59E9E0B, 0xCBAADE14,
    0xEECC86BC, 0x60622CA7, 0x9CAB5CAB, 0xB2F3846E,
    0x648B1EAF, 0x19BDF0CA, 0xA02369B9, 0x655ABB50,
    0x40685A32, 0x3C2AB4B3, 0x319EE9D5, 0xC021B8F7,
    0x9B540B19, 0x875FA099, 0x95F7997E, 0x623D7DA8,
    0xF837889A, 0x97E32D77, 0x11ED935F, 0x16681281,
    0x0E358829, 0xC7E61FD6, 0x96DEDFA1, 0x7858BA99,
    0x57F584A5, 0x1B227263, 0x9B83C3FF, 0x1AC24696,
    0xCDB30AEB, 0x532E3054, 0x8FD948E4, 0x6DBC3128,
    0x58EBF2EF, 0x34C6FFEA, 0xFE28ED61, 0xEE7C3C73,
    0x5D4A14D9, 0xE864B7E3, 0x42105D14, 0x203E13E0,
    0x45EEE2B6, 0xA3AAABEA, 0xDB6C4F15, 0xFACB4FD0,
    0xC742F442, 0xEF6ABBB5, 0x654F3B1D, 0x41CD2105,
    0xD81E799E, 0x86854DC7, 0xE44B476A, 0x3D816250,
    0xCF62A1F2, 0x5B8D2646, 0xFC8883A0, 0xC1C7B6A3,
    0x7F1524C3, 0x69CB7492, 0x47848A0B, 0x5692B285,
    0x095BBF00, 0xAD19489D, 0x1462B174, 0x23820E00,
    0x58428D2A, 0x0C55F5EA, 0x1DADF43E, 0x233F7061,
    0x3372F092, 0x8D937E41, 0xD65FECF1, 0x6C223BDB,
    0x7CDE3759, 0xCBEE7460, 0x4085F2A7, 0xCE77326E,
    0xA6078084, 0x19F8509E, 0xE8EFD855, 0x61D99735,
    0xA969A7AA, 0xC50C06C2, 0x5A04ABFC, 0x800BCADC,
    0x9E447A2E, 0xC3453484, 0xFDD56705, 0x0E1E9EC9,
    0xDB73DBD3, 0x105588CD, 0x675FDA79, 0xE3674340,
    0xC5C43465, 0x713E38D8, 0x3D28F89E, 0xF16DFF20,
    0x153E21E7, 0x8FB03D4A, 0xE6E39F2B, 0xDB83ADF7
},
{
    0xE93D5A68, 0x948140F7, 0xF64C261C, 0x94692934,
    0x411520F7, 0x7602D4F7, 0xBCF46B2E, 0xD4A20068,
    0xD4082471, 0x3320F46A, 0x43B7D4B7, 0x500061AF,
    0x1E39F62E, 0x97244546, 0x14214F74, 0xBF8B8840,
    0x4D95FC1D, 0x96B591AF, 0x70F4DDD3, 0x66A02F45,
    0xBFBC09EC, 0x03BD9785, 0x7FAC6DD0, 0x31CB8504,
    0x96EB27B3, 0x55FD3941, 0xDA2547E6, 0xABCA0A9A,
    0x28507825, 0x530429F4, 0x0A2C86DA, 0xE9B66DFB,
    0x68DC1462, 0xD7486900, 0x680EC0A4, 0x27A18DEE,
    0x4F3FFEA2, 0xE887AD8C, 0xB58CE006, 0x7AF4D6B6,
    0xAACE1E7C, 0xD3375FEC, 0xCE78A399, 0x406B2A42,
    0x20FE9E35, 0xD9F385B9, 0xEE39D7AB, 0x3B124E8B,
    0x1DC9FAF7, 0x4B6D1856, 0x26A36631, 0xEAE397B2,
```

```
    0x3A6EFA74, 0xDD5B4332, 0x6841E7F7, 0xCA7820FB,
    0xFB0AF54E, 0xD8FEB397, 0x454056AC, 0xBA489527,
    0x55533A3A, 0x20838D87, 0xFE6BA9B7, 0xD096954B,
    0x55A867BC, 0xA1159A58, 0xCCA92963, 0x99E1DB33,
    0xA62A4A56, 0x3F3125F9, 0x5EF47E1C, 0x9029317C,
    0xFDF8E802, 0x04272F70, 0x80BB155C, 0x05282CE3,
    0x95C11548, 0xE4C66D22, 0x48C1133F, 0xC70F86DC,
    0x07F9C9EE, 0x41041F0F, 0x404779A4, 0x5D886E17,
    0x325F51EB, 0xD59BC0D1, 0xF2BCC18F, 0x41113564,
    0x257B7834, 0x602A9C60, 0xDFF8E8A3, 0x1F636C1B,
    0x0E12B4C2, 0x02E1329E, 0xAF664FD1, 0xCAD18115,
    0x6B2395E0, 0x333E92E1, 0x3B240B62, 0xEEBEB922,
    0x85B2A20E, 0xE6BA0D99, 0xDE720C8C, 0x2DA2F728,
    0xD0127845, 0x95B794FD, 0x647D0862, 0xE7CCF5F0,
    0x5449A36F, 0x877D48FA, 0xC39DFD27, 0xF33E8D1E,
    0x0A476341, 0x992EFF74, 0x3A6F6EAB, 0xF4F8FD37,
    0xA812DC60, 0xA1EBDDF8, 0x991BE14C, 0xDB6E6B0D,
    0xC67B5510, 0x6D672C37, 0x2765D43B, 0xDCD0E804,
    0xF1290DC7, 0xCC00FFA3, 0xB5390F92, 0x690FED0B,
    0x667B9FFB, 0xCEDB7D9C, 0xA091CF0B, 0xD9155EA3,
    0xBB132F88, 0x515BAD24, 0x7B9479BF, 0x763BD6EB,
    0x37392EB3, 0xCC115979, 0x8026E297, 0xF42E312D,
    0x6842ADA7, 0xC66A2B3B, 0x12754CCC, 0x782EF11C,
    0x6A124237, 0xB79251E7, 0x06A1BBE6, 0x4BFB6350,
    0x1A6B1018, 0x11CAEDFA, 0x3D25BDD8, 0xE2E1C3C9,
    0x44421659, 0x0A121386, 0xD90CEC6E, 0xD5ABEA2A,
    0x64AF674E, 0xDA86A85F, 0xBEBFE988, 0x64E4C3FE,
    0x9DBC8057, 0xF0F7C086, 0x60787BF8, 0x6003604D,
    0xD1FD8346, 0xF6381FB0, 0x7745AE04, 0xD736FCCC,
    0x83426B33, 0xF01EAB71, 0xB0804187, 0x3C005E5F,
    0x77A057BE, 0xBDE8AE24, 0x55464299, 0xBF582E61,
    0x4E58F48F, 0xF2DDFDA2, 0xF474EF38, 0x8789BDC2,
    0x5366F9C3, 0xC8B38E74, 0xB475F255, 0x46FCD9B9,
    0x7AEB2661, 0x8B1DDF84, 0x846A0E79, 0x915F95E2,
    0x466E598E, 0x20B45770, 0x8CD55591, 0xC902DE4C,
    0xB90BACE1, 0xBB8205D0, 0x11A86248, 0x7574A99E,
    0xB77F19B6, 0xE0A9DC09, 0x662D09A1, 0xC4324633,
    0xE85A1F02, 0x09F0BE8C, 0x4A99A025, 0x1D6EFE10,
    0x1AB93D1D, 0x0BA5A4DF, 0xA186F20F, 0x2868F169,
    0xDCB7DA83, 0x573906FE, 0xA1E2CE9B, 0x4FCD7F52,
    0x50115E01, 0xA70683FA, 0xA002B5C4, 0x0DE6D027,
    0x9AF88C27, 0x773F8641, 0xC3604C06, 0x61A806B5,
    0xF0177A28, 0xC0F586E0, 0x006058AA, 0x30DC7D62,
    0x11E69ED7, 0x2338EA63, 0x53C2DD94, 0xC2C21634,
    0xBBCBEE56, 0x90BCB6DE, 0xEBFC7DA1, 0xCE591D76,
    0x6F05E409, 0x4B7C0188, 0x39720A3D, 0x7C927C24,
    0x86E3725F, 0x724D9DB9, 0x1AC15BB4, 0xD39EB8FC,
    0xED545578, 0x08FCA5B5, 0xD83D7CD3, 0x4DAD0FC4,
    0x1E50EF5E, 0xB161E6F8, 0xA28514D9, 0x6C51133C,
    0x6FD5C7E7, 0x56E14EC4, 0x362ABFCE, 0xDDC6C837,
    0xD79A3234, 0x92638212, 0x670EFA8E, 0x406000E0
},
{
```

```
0x3A39CE37, 0xD3FAF5CF, 0xABC27737, 0x5AC52D1B,
0x5CB0679E, 0x4FA33742, 0xD3822740, 0x99BC9BBE,
0xD5118E9D, 0xBF0F7315, 0xD62D1C7E, 0xC700C47B,
0xB78C1B6B, 0x21A19045, 0xB26EB1BE, 0x6A366EB4,
0x5748AB2F, 0xBC946E79, 0xC6A376D2, 0x6549C2C8,
0x530FF8EE, 0x468DDE7D, 0xD5730A1D, 0x4CD04DC6,
0x2939BBDB, 0xA9BA4650, 0xAC9526E8, 0xBE5EE304,
0xA1FAD5F0, 0x6A2D519A, 0x63EF8CE2, 0x9A86EE22,
0xC089C2B8, 0x43242EF6, 0xA51E03AA, 0x9CF2D0A4,
0x83C061BA, 0x9BE96A4D, 0x8FE51550, 0xBA645BD6,
0x2826A2F9, 0xA73A3AE1, 0x4BA99586, 0xEF5562E9,
0xC72FEFD3, 0xF752F7DA, 0x3F046F69, 0x77FA0A59,
0x80E4A915, 0x87B08601, 0x9B09E6AD, 0x3B3EE593,
0xE990FD5A, 0x9E34D797, 0x2CF0B7D9, 0x022B8B51,
0x96D5AC3A, 0x017DA67D, 0xD1CF3ED6, 0x7C7D2D28,
0x1F9F25CF, 0xADF2B89B, 0x5AD6B472, 0x5A88F54C,
0xE029AC71, 0xE019A5E6, 0x47B0ACFD, 0xED93FA9B,
0xE8D3C48D, 0x283B57CC, 0xF8D56629, 0x79132E28,
0x785F0191, 0xED756055, 0xF7960E44, 0xE3D35E8C,
0x15056DD4, 0x88F46DBA, 0x03A16125, 0x0564F0BD,
0xC3EB9E15, 0x3C9057A2, 0x97271AEC, 0xA93A072A,
0x1B3F6D9B, 0x1E6321F5, 0xF59C66FB, 0x26DCF319,
0x7533D928, 0xB155FDF5, 0x03563482, 0x8ABA3CBB,
0x28517711, 0xC20AD9F8, 0xABCC5167, 0xCCAD925F,
0x4DE81751, 0x3830DC8E, 0x379D5862, 0x9320F991,
0xEA7A90C2, 0xFB3E7BCE, 0x5121CE64, 0x774FBE32,
0xA8B6E37E, 0xC3293D46, 0x48DE5369, 0x6413E680,
0xA2AE0810, 0xDD6DB224, 0x69852DFD, 0x09072166,
0xB39A460A, 0x6445C0DD, 0x586CDECF, 0x1C20C8AE,
0x5BBEF7DD, 0x1B588D40, 0xCCD2017F, 0x6BB4E3BB,
0xDDA26A7E, 0x3A59FF45, 0x3E350A44, 0xBCB4CDD5,
0x72EACEA8, 0xFA6484BB, 0x8D6612AE, 0xBF3C6F47,
0xD29BE463, 0x542F5D9E, 0xAEC2771B, 0xF64E6370,
0x740E0D8D, 0xE75B1357, 0xF8721671, 0xAF537D5D,
0x4040CB08, 0x4EB4E2CC, 0x34D2466A, 0x0115AF84,
0xE1B00428, 0x95983A1D, 0x06B89FB4, 0xCE6EA048,
0x6F3F3B82, 0x3520AB82, 0x011A1D4B, 0x277227F8,
0x611560B1, 0xE7933FDC, 0xBB3A792B, 0x344525BD,
0xA08839E1, 0x51CE794B, 0x2F32C9B7, 0xA01FBAC9,
0xE01CC87E, 0xBCC7D1F6, 0xCF0111C3, 0xA1E8AAC7,
0x1A908749, 0xD44FBD9A, 0xD0DADECB, 0xD50ADA38,
0x0339C32A, 0xC6913667, 0x8DF9317C, 0xE0B12B4F,
0xF79E59B7, 0x43F5BB3A, 0xF2D519FF, 0x27D9459C,
0xBF97222C, 0x15E6FC2A, 0x0F91FC71, 0x9B941525,
0xFAE59361, 0xCEB69CEB, 0xC2A86459, 0x12BAA8D1,
0xB6C1075E, 0xE3056A0C, 0x10D25065, 0xCB03A442,
0xE0EC6E0E, 0x1698DB3B, 0x4C98A0BE, 0x3278E964,
0x9F1F9532, 0xE0D392DF, 0xD3A0342B, 0x8971F21E,
0x1B0A7441, 0x4BA3348C, 0xC5BE7120, 0xC37632D8,
0xDF359F8D, 0x9B992F2E, 0xE60B6F47, 0x0FE3F11D,
0xE54CDA54, 0x1EDAD891, 0xCE6279CF, 0xCD3E7E6F,
0x1618B166, 0xFD2C1D05, 0x848FD2C5, 0xF6FB2299,
0xF523F357, 0xA6327623, 0x93A83531, 0x56CCCD02,
```

```
        0xACF08162, 0x5A75EBB5, 0x6E163697, 0x88D273CC,
        0xDE966292, 0x81B949D0, 0x4C50901B, 0x71C65614,
        0xE6C6C7BD, 0x327A140A, 0x45E1D006, 0xC3F27B9A,
        0xC9AA53FD, 0x62A80F00, 0xBB25BFE2, 0x35BDD2F6,
        0x71126905, 0xB2040222, 0xB6CBCF7C, 0xCD769C2B,
        0x53113EC0, 0x1640E3D3, 0x38ABBD60, 0x2547ADF0,
        0xBA38209C, 0xF746CE76, 0x77AFA1C5, 0x20756060,
        0x85CBFE4E, 0x8AE88DD8, 0x7AAAF9B0, 0x4CF9AA7E,
        0x1948C25C, 0x02FB8A8C, 0x01C36AE4, 0xD6EBE1F9,
        0x90D4F869, 0xA65CDEA0, 0x3F09252D, 0xC208E69F,
        0xB74E6132, 0xCE77E25B, 0x578FDFE3, 0x3AC372E6
    }
};
```