

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» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | О.А. Сафарьян |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 г. | |

**ОТЧЕТ**

по производственной, проектно-технологической практике

в ФГАНУ НИИ «Спецвузавтоматика»

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. П. Ковалев

подпись, дата

Обозначение отчета ПП.990000.000 Группа ВКБ32

|  |  |
| --- | --- |
| Специальность 10.05.01 | Компьютерная безопасность |
| Специализация | Математические методы защиты информации |

Руководитель практики:

|  |  |  |  |
| --- | --- | --- | --- |
| от предприятия | зам. директора по научной работе | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | К. Ю. Гуфан |
| от кафедры | доцент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | О. С. Бурякова |

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

дата подпись преподавателя

Ростов-на-Дону

2025 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 производственной, проектно-технологической практике

в ФГАНУ НИИ «Спецвузавтоматика»

в период с «10» июня 2024 г. по «6» июля 2025 г.

Обучающийся Ковалев Данил Петрович

Обозначение отчета ПП.990000.000 Группа ВКБ32

Срок представления отчета на кафедру «\_\_» \_\_\_\_\_\_\_\_ 2025 г.

Содержание индивидуального задания

1. асимметричный алгоритм шифрования данных RSA;
2. оформление списка использованных информационных ресурсов по ГОСТ Р 7.0.100–2018;
3. оформление отчета по практике согласно документу “Правила оформления письменных работ обучающихся для технических направлений подготовки” от 16.12.2020 приказа 242 “О введение документов действие”.

|  |  |  |
| --- | --- | --- |
| Руководитель практики от  кафедры | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | О. С. Бурякова |
| Руководитель практики от  предприятия | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | К. Ю. Гуфан |
| Задание принял к исполнению | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | Д. П. Ковалев |



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» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | О.А. Сафарьян |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 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. | 09.06.2025 |
| 2 | Получение индивидуального задания. | 10.06.2025 |
| 3 | Ознакомление с теоретическим материалом. | 11.06.2025-15.06.2025 |
| 4 | Оформление программной реализации. | 16.06.2025-29.06.2025 |
| 5 | Оформление отчёта по практике. | 29.06.2025-02.07.2025 |
| 6 | Защита отчёта по практике на предприятии. | 03.07.2025 |
| 7 | Защита отчёта по практике на кафедре. | 04.07.2025-05.07.2025 |

Руководитель практики:

|  |  |  |  |
| --- | --- | --- | --- |
| от предприятия | зам. директора по научной работе | \_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | К. Ю. Гуфан |
| от кафедры | доцент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | О. С. Бурякова |

Ростов-на-Дону

2025 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 |
| 09.06.2025 | ФГАНУ НИИ «Спецвузавтоматика» | Знакомство с предприятием, прохождение вводного инструктажа. |  |
| 10.06.2025 | ФГАНУ НИИ «Спецвузавтоматика» | Ознакомление с территорией предприятия, прохождение первичного инструктажа по ТБ, ПБ |  |
| 10.06.2025 | ФГАНУ НИИ «Спецвузавтоматика» | Получение индивидуального задания. |  |
| 11.06.2025-15.06.2025 | ФГАНУ НИИ «Спецвузавтоматика» | Ознакомление с теоретическим материалом |  |
| 17.06.2025-29.06.2025 | ФГАНУ НИИ «Спецвузавтоматика» | Изучение общих характеристик методов сжатия информации. |  |
| 01.07.2025-02.07.2025 | ФГАНУ НИИ «Спецвузавтоматика» | Оформление отчёта по практике. |  |
| 03.07.2025 | ФГАНУ НИИ «Спецвузавтоматика» | Защита отчёта по практике на предприятии. |  |
| 04.07.2025-05.07.2025 | кафедра «КБИС» | Защита отчёта по практике на кафедре. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| от предприятия | зам. директора по научной работе | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | К. Ю. Гуфан |
| от кафедры | доцент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | О. С. Бурякова |

ОТЗЫВ-ХАРАКТЕРИСТИКА

Обучающийся Ковалев Данил Петрович

группа ВКБ32

кафедра «Кибербезопасность информационных систем»

вид практики учебная практика

наименование места практики ФГАНУ НИИ «Спецвузавтоматика»

Обучающийся выполнил задания программы практики:

изучил задание, собрал информацию, разработал структуру, организацию и управление проекта, реализовал код, протестировал его, изготовил готовый продукт.

Дополнительно ознакомился/изучил:

Логирование (logging), написание unit test (Pytest), GitHub Actions

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

|  |  |
| --- | --- |
|  | Руководитель практики  от предприятия  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_ г.  М.П. |
|  |  |

**Содержание**

[Введение 6](#_Toc201047431)

[1 Криптографический алгоритм RSA 8](#_Toc201047432)

[1.1 История алгоритма RSA.. 8](#_Toc201047433)

[1.2 Основная терминология 10](#_Toc201047434)

[1.3 Описание 13](#_Toc201047435)

[1.4 Критическая оценка криптостойкости криптоалгоритма RSA 15](#_Toc201047436)

[1.5 Применение 18](#_Toc201047437)

[1.6 Модификации RSA 18](#_Toc201047438)

[2 Программная реализация 20](#_Toc201047439)

[2.1 Высокоуровневый язык программирования Python 20](#_Toc201047440)

[2.2 Общая информация про программную реализацию 20](#_Toc201047441)

[2.3 Конфигурационные файлы 21](#_Toc201047446)

[2.4 Логика создания ключей 24](#_Toc201047447)

[Заключение 28](#_Toc201047448)

[Перечень использованных информационных ресурсов 29](#_Toc201047449)

[Приложение А 30](#_Toc201047450)

[Приложение Б 34](#_Toc201047451)

# Введение

Своим существованием Федеральное государственное автономное научное учреждение «Спецвузавтоматика» обязано инициативному изобретателю Алексею Аграновскому, выпускнику физического факультета Ростовского государственного университета, который с 1975 года работал в университетском вычислительном центре, начав с техника и закончив в должности начальника отдела автоматизации научных исследований. В 1992 году он основал на университетской базе конструкторское бюро «Спецвузавтоматика», которое в 2002 году превратилось во ФГАНУ НИИ «Специализированные вычислительные устройства защиты и автоматика» при Минобрнауки РФ. Аграновский оставался директором института до своей смерти в 2010 году.

Целью деятельности ФГАНУ НИИ «Спецвузавтоматика» является   
создание и внедрение информационных технологий для развития инновационного потенциала различных государственных структур по решению их собственных задач в области науки, техники и образования.

Сфера компетенции ФГАНУ НИИ «Спецвузавтоматика» достаточно обширна, так как она включает в себя все аспекты работы в сферах   
безопасности и защиты информации, управления проектами и внедрения инновационных технологий. Также институтом проводятся различные научные исследования в области телекоммуникаций, создания технических средств  
связи, защиты и обработки информации, разрабатывается программное обеспечение по заказу и создаются электронные устройства, начиная от идеи и заканчивая серийными образцами.

В Институте создана организационная структура, позволяющая развивать прикладные исследования и научные школы, осваивать новые технологии, реализовывать инновационные продукты.

Существует несколько подразделений:

1. Научно-исследовательские лаборатории (НИЛ);
2. Южный региональный аттестационный центр (ЮРАЦ);
3. Центр трансфера технологий и коммерциализации (ЦТТК);
4. Центр космических технологий «Арктурус» (ЦКТ);
5. Национальный центр противодействия терроризму и экстремизму в образовательной среде и сети Интернет (НЦПТИ);

И кроме всего прочего существуют такие приоритетные направления деятельности:

1. Выполнение работ и исследований в сфере информационных технологий, естественных и технических наук, информационной безопасности, инновационным методам разработки программного обеспечения, проблематике информационного противодействия терроризму и экстремизму в сети Интернет и образовательной среде, телекоммуникациям и управлению, созданию новых образовательных технологий;
2. Координация научных исследований по перспективным направлениям области безопасности Российской Федерации;
3. Разработка и внедрение в сферу образования перспективных системных

и обучающих технологий;

1. Организация инновационной деятельности с учетом основных направлений социально-экономического развития Российской Федерации, реализации приоритетных направлений государственной политики Российской Федерации в научно-технической сфере;
2. Внедрение в производство новых технологий и техники по заказам заинтересованных предприятий и организаций;
3. Содействие развитию кадрового потенциала региона, в том числе в инновационной и проектной деятельности.

## 1 Криптографический алгоритм RSA

### История алгоритма RSA

Шифр предложен в 1977 г. Р. Ривестом (Ron Rivest) (Рис. 1), А. Шамиром (Adi Shamir) (Рис. 2) и Л. Адлеманом (Leonard Adleman) (Рис. 3) и назван  
по именам своих создателей. RSA — это алгоритм асимметричного шифрования и цифровой подписи. До сих пор RSA является одной из самых распространенных систем асимметричной криптографии.



Рисунок 1 – Рональд Линн Ривест



Рисунок 2 – Ади Шамир

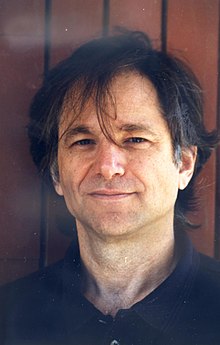


Рисунок 3 – Леонард Макс Адлеман

Идея асимметричной криптосистемы с открытым и закрытым ключом приписывается Уитфилду Диффи и Мартину Хеллману, которые опубликовали эту концепцию в 1976 году. Они также ввели цифровые подписи и попытались применить теорию чисел. В их формулировке использовался секретный ключ с общим доступом, созданный путем экспоненциализации некоторого числа, простого по модулю. Однако они оставили открытой проблему реализации односторонней функции, возможно, потому что сложность факторизации в то время не была хорошо изучена [1].

Рон Ривест, Ади Шамир и Леонард Адлеман из Массачусетского технологического института в течение года предприняли несколько попыток создать одностороннюю функцию, которую было бы трудно инвертировать. Ривест и Шамир, будучи компьютерными учеными, предложили множество потенциальных функций, а Адлеман, будучи математиком, отвечал за поиск их слабых мест. Они опробовали множество подходов, включая «ранцевый» и «перестановочные полиномы». Какое-то время они думали, что то, чего они хотели достичь, невозможно из-за противоречивых требований. В апреле 1977 года они провели Песах в доме одного из студентов и выпили много манишевицкого вина, а затем вернулись к себе домой около полуночи. Ривест, не в силах заснуть, лег на диван с учебником математики и начал думать о своей односторонней функции. Остаток ночи он провел, формализуя свою идею, и к рассвету большая часть статьи была готова. Алгоритм теперь известен как  
RSA - инициалы их фамилий в том же порядке, что и в их статье [2].

Клиффорд Кокс, английский математик, работавший в британской разведывательной службе Government Communications Headquarters (GCHQ), описал эквивалентную систему во внутреннем документе в 1973 году. Однако, учитывая относительно дорогие компьютеры, необходимые для ее реализации в то время, она считалась в основном курьезом и, насколько известно, так и не была применена.

Открытие Клиффорда Кокса было раскрыто только в 1997 году из-за его сверхсильного засекречивания. В августе 1977 года в колонке «Математические игры» Мартина Гарднера в журнале Scientific American с разрешения Рональда Ривеста появилось первое описание криптосистемы RSA. Читателям также было предложено дешифровать английскую фразу, зашифрованную описанным алгоритмом, показанную на рисунке 4.

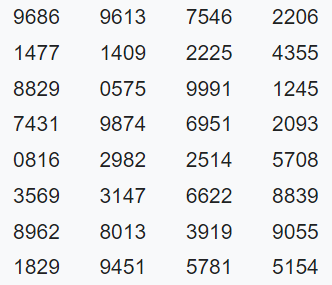


Рисунок 4 – Зашифрованная фраза с помощью RSA

### Основная терминология

Криптография – это область прикладной математики, основной задачей которой является защита цифровой информации от несанкционированного доступа [4]. Криптосистема – семейство выбираемых с помощью ключа обратимых преобразований, которые трансформируют защищаемый блок информации в шифрограмму и обратно [4]. Исходя из формулировки проблемы ясно, что криптосистема должнаобеспечивать:

1. Конфиденциальность**,** то есть невозможность прочтения посторонним лицомданных;
2. Целостность, то есть невозможность незаметного изменения информации;
3. Аутентификацию сторон коммуникации, то есть подтверждение подлинности авторства и других атрибутов информации.

Ключ шифрования может быть последовательностью символов, структурой данных, физическим предметом. Все зависит от способа шифрования.

Примеры ключей:

1. Таблица замены символов, используемая в шифре подстановки;
2. Секретное слово, используемое в шифре Виженера;
3. Цилиндр определённого диаметра, используемый в шифре Спарты;
4. Порядок и поворот символьных роторов, используемых в широко известной шифровальной машине ENIGMA.

По специфике используемых ключей все криптосистемы можно разделить на два типа: симметричные и асимметричные. Рассмотрим каждый тип шифрования.

Симметричное шифрование — это способ шифрования данных, при котором один и тот же ключ используется и для кодирования, и для восстановления информации [5].

Недостатки симметричного шифрования:

* 1. Проблема передача ключа;
  2. Проверка подлинности.

Достоинства симметричного шифрования:

1. Скорость;
2. Простота реализации;
3. Маленькая длина ключа.

Наиболее распространенные симметричные алгоритмы:

* 1. AES (Advanced Encryption System);
  2. ГОСТ 34.13–2018;
  3. RC5 (Rivest Cipher);
  4. Blowfish;
  5. Twofish.

Асимметричное шифрование — это метод шифрования данных, предполагающий использование двух ключей — открытого и закрытого. Открытый (публичный) ключ применяется для шифрования информации и может передаваться по незащищенным каналам. Закрытый (приватный) ключ применяется для расшифровки данных, зашифрованных открытым ключом. Открытый и закрытый ключи — это очень большие числа, связанные друг с другом определенной функцией, но так, что, зная одно, крайне сложно вычислить второе [6].

Недостатки асимметричного шифрования:

1. Низкая скорость операции дешифровки и расшифровки;
2. Защита открытых ключей.

Наиболее распространённые асимметричные алгоритмы:

1. RSA (аббревиатура от Rivest, Shamir и Adelman);
2. DSA (Digital Signature Algorithm);
3. ECDSA (Elliptic Curve Digital Signature Algorithm);
4. Схема Эль-Гамаля.

Теоретическиприватный ключ от асимметричного шифра  
можновычислить, зная публичный ключи алгоритм шифрования.

### 1.3 Описание криптографической системы

Криптографические системы с открытым ключом используют односторонние функции, которые обладают следующими свойствами:

1. Если известно , то вычислить легко;
2. Если известно , то для вычисления *x* нет эффективного способа.

Под односторонностью понимается не математически доказанная одно направленность, а невозможность вычислить обратное значение за разумный срок, используя современные вычислительные средства.

Криптографическая система с открытым ключом RSA основана на сложности задачи разложения на множители произведения двух больших простых чисел. Для шифрования используется операция возведения в степень  
по модулю большого числа. Для дешифрования за разумное время необходимо уметь вычислять функцию Эйлера от данного числа,  
для чего необходимо знать разложение числа на простые множители.

Каждый участник криптографической системы имеет открытый ключ, а также закрытый ключ. В RSA каждый ключ состоит из пары целых  
чисел. Каждый участник создаёт свой публичный и приватный  
ключ самостоятельно. Закрытый ключ каждый из них держит в секрете,  
а открытые ключи можно сообщать кому угодно или даже публиковать  
их. Открытый и закрытый ключи каждого участника обмена сообщениями  
в криптосистеме RSA являются взаимно обратными, то есть:

Для любых допустимых пар открытого и закрытого  
ключей существуют соответствующие функции шифрования  
 и расшифрования такие, что для любого  
сообщения , где М – множество допустимых сообщений,  
.

RSA-ключи генерируются следующим образом:

1. Выбираются два различных случайных простых числа и заданного размера (например, 2048 бит каждое);
2. Вычисляется их произведение , которое называется модулем;
3. Вычисляется значение функции Эйлера от числа :
4. выбирается целое число е, открытая экспонента, взаимно простое  
   со значением функции . Число берётся из интервала  
   . Обычно в качестве берут простые числа,  
   содержащие небольшое количество единичных бит в двоичной  
   записи, например, простые из чисел Ферма: 17, 257 или 65537, так как  
   в этом случае время, необходимое для шифрования с использованием  
   быстрого возведения в степень, будет меньше;
5. вычисляется число , секретная экспонента, удовлетворяющее сравнению . Обычно секретная экспонента вычисляется при помощи расширенного алгоритма Евклида;
6. пара публикуется в качестве открытого ключа RSA;
7. пара играет роль закрытого ключа RSA и держится в секрете.

Блок-схема работы алгоритма RSA представлено ниже на рисунке 5.

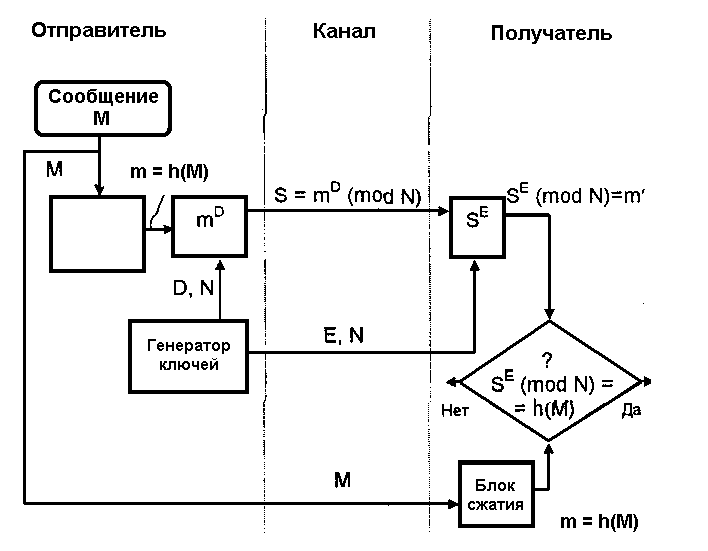


Рисунок 5 – Блок-схема работы алгоритма RSA

### 1.4 Критическая оценка криптостойкости криптоалгоритма RSA

Основной принцип работы алгоритма RSA заключается в использовании двух больших простых чисел и математической операции на основе  
этих чисел. Вначале генерируются два различных простых числа и ,  
после чего вычисляется их произведение . Затем выбирается  
целое число , такое что и , где  
. Используя значение e и модуль N,  
вычисляется число d обратное к е по модулю , то есть .

Итак, открытым ключом являются значения (N, e), которые распространяются всем желающим шифровать сообщения. Для шифрования сообщения m отправитель вычисляет .  
Получатель сообщения может расшифровать его, используя свой закрытый ключ, вычисляя

Критическая оценка алгоритма RSA связана с его  
безопасностью. Существует несколько известных методов взлома RSA,  
таких как атака факторизации и атака Гельфонде-Шенкса [2]. В первом  
случае злоумышленник пытается найти простые множители числа N, что позволит ему вычислить значение и восстановить закрытый ключ. Второй метод основан на анализе частоты использования определенных чисел  
в шифр тексте.

Однако при правильном выборе значений и , а также  
 и , алгоритм RSA остается достаточно безопасным. Кроме того,  
для обеспечения дополнительной защиты можно использовать  
протоколы подписи сообщений и дополнительные методы шифрования,  
такие как AES.

Некоторые факты о RSA:

1. RSA является одним из самых распространенных криптографических алгоритмов в мире;
2. Длина ключа RSA напрямую влияет на его безопасность: чем длиннее ключ, тем сложнее его взломать;
3. RSA используется для защиты данных во многих областях, включая банковское дело, электронную почту, интернет-транзакции и т. д.;
4. RSA является асимметричным алгоритмом, то есть для шифрования и расшифровки сообщений используются разные ключи;
5. RSA может быть использован не только для шифрования, но и для цифровой подписи сообщений;
6. RSA может также использоваться для аутентификации пользователя, например, при входе на сайт или в систему;
7. В 2017 году команда криптографов из Китая и США объявила о создании самого большого квантового компьютера, способного взламывать алгоритмы шифрования, включая RSA. Однако разработчики RSA отмечают,  
   что существуют методы защиты от квантовых компьютеров, например, использование алгоритмов симметричного шифрования;
8. RSA также имеет некоторые недостатки, включая  
   высокую вычислительную сложность и необходимость дополнительных  
   мер безопасности, таких как хранение ключей и защита от атак подбора ключа;
9. Несмотря на свои недостатки, RSA остается одним  
   из наиболее распространенных и широко используемых криптографических алгоритмов благодаря своей простоте и эффективности.

Существуют разные способы повышения безопасности алгоритма RSA. Распишем ниже все рекомендации и подходы, которые используются на момент 2024 года.

Например, одним из подходов к повышению безопасности алгоритма RSA является использование более длинных ключей. Существует концепция «квантовой безопасности», которая предполагает использование ключей длиной в несколько тысяч бит, чтобы защитить данные от будущих квантовых вычислений.

Другой подход к повышению безопасности – это использование новых методов шифрования, которые могут быть применены вместо или в сочетании с алгоритмом RSA. Например, квантовые алгоритмы шифрования, такие как GGH13, обещают быть более устойчивыми к атакам квантовых компьютеров. Однако эти алгоритмы пока еще находятся на стадии исследований и не готовы к широкому применению.

Ещё один подход повышения безопасности – это использования цифровых подписей и цепочек доверия для сообщений во избегание атаки «человек посередине», при передаче данных с помощью RSA. Злоумышленник может подменить ключ, когда собеседник попросит открытый ключ, тем самым сможет прочитать конфиденциальную информацию.

И самый главный подход, который практикуют во многих компаниях – периодическое обновление ключей для предотвращения возможных атак подбора ключа.

Подведя все итоги, на 2010 год система шифрования на основе RSA считалась надёжной начиная с размера в 1024 бита. Однако в конце декабря ушедшего года китайские исследователи продемонстрировали возможность взлома длинных RSA-ключей с помощью квантовых компьютеров [4]. Так, для вскрытия 48-битного RSA оказалось достаточно 10-кьюбитного квантового компьютера, а для 2048-битного - системы из 372 кьюбитов. А мощность IBM Osprey, ввод которого должен состояться уже совсем скоро, составит 433 кьюбита. Раньше считалось, что для взлома RSA понадобится не менее 20 млн кьюбитов, причем даже такой системе, появления которой стоило ожидать не менее 25 лет, будет необходимо восемь часов работы [3].

Бизнес-консультант по информационной безопасности, Positive Technologies Алексей Лукацкий делает такие далеко идущие выводы: «И, хотя про взломы RSA уже не один фейк опубликован, в данном случае стоит прислушаться к Брюсу Шнайеру. И хотя еще предстоит перепроверить результаты исследований китайцев, есть вероятность, что они действительно сделали то, что пишут, а именно - сломали 2048-битный ключ RSA. А значит, 2023 г. поставит перед многими ИБ-специалистами совершенно новые задачи и заставит пересмотреть все свои планы. Менять основы ИБ, лежащие в фундаменте многих систем электронной коммерции, - штука непростая…» Если сообщение китайских ученых соответствует действительности, алгоритм RSA-2048 будет взломан квантовыми компьютерами в ближайшие пару лет.

### Применение алгоритма RSA

Криптографический алгоритм RSA является одним из самых популярных и широко используемых алгоритмов шифрования. RSA основан на сложности разложения на множителей (факторизации) больших простых чисел.

Ниже приведены применения алгоритма RSA:

1. RSA может быть использован для защиты писем электронной почты от несанкционированного доступа;
2. RSA используется для аутентификации пользователя. Например, при использовании системы двухфакторной аутентификации;
3. RSA используется во многих программах для проверки того, что обновления программного обеспечения исходят от оригинального разработчика. Образно говоря, RSA используется для создания цифровой подписи версии продукта.
4. RSA использовался для генерации SSH ключей раньше, пока не вышли новые алгоритмы ассиметричного шифрования;

### 1.6 Модификации RSA

Существует несколько модификаций RSA, которые были разработаны в течение последних нескольких десятилетий.

RSA-OAEP (Optimal Asymmetric Encryption Padding) — это расширение, которое обеспечивает более безопасную передачу данных, чем базовая версия RSA. Здесь используются дополнительные алгоритмы хеширования и маскирования для защиты от атак, связанных с возможностью получения одного и того же шифр текста при использовании одного и того же открытого ключа. Алгоритм включает в себя функции хеширования, генерацию случайных строк и преобразования, которые увеличивают стойкость криптографической системы.

Multi-Prime RSA — это модификация RSA, которая использует  
более двух простых чисел в формуле для генерации ключей. Это  
позволяет создавать более короткие ключи с той же степенью безопасности, но данная модификация менее эффективна при вычислении шифрования и  
расшифрования. Например, если использовать три простых числа , q и r,  
то формула для вычисления открытого ключа будет иметь следующий вид: .

RSA-CRT (Chinese Remainder Theorem) — это модификация RSA, которая использует китайскую теорему об остатках для ускорения расшифрования. Ускорение дешифрации достигается путем разбиения больших чисел на меньшие части, а после этого используется китайская теорема для быстрого объединения результатов.

Каждая из этих модификаций RSA предназначена для решения определенных проблем или повышения эффективности алгоритма.  
Например, модификация RSA-OAEP увеличивает  
уровень безопасности шифрования, Multi-Prime RSA позволяет создавать короткие ключи, а RSA-CRT ускоряет процесс расшифрования. Однако  
следует помнить, что любая модификация может иметь свои собственные уязвимости и недостатки.

## 2 Программная реализация

Алгоритм RSA будет реализован на языке Python, среда разработки – IDE PyCharm. Будут показаны примеры и документация.

### 2.1 Высокоуровневый язык программирования Python

Python является высокоуровневым языком программирования общего назначения и ориентирован на лаконичность и простоту кода, а также на легкость его чтения. Синтаксис ядра языка минималистичен, а стандартная библиотека языка Python включает большой объем различных функций. Помимо стандартной библиотеки для языка Python существует большое количество дополнительных библиотек. Язык обладает динамической типизацией, автоматическим управлением памятью. Так же для языка Python существует множество фреймворков, которые помогают ускорить работу написания кода. Как уже говорилось, данный язык программирования ориентирован на простоту и читаемость написанного кода.

### 2.2. Общая информация про программную реализацию

В начале работы над проектом рассматривалась возможность разработки собственной реализации алгоритма RSA. Однако, учитывая существующие решения и стремление избежать ненужных затрат времени и ресурсов, было принято решение воспользоваться существующей реализацией, предложенной в репозитории от Sybren Stuvel под названием "python-rsa". Целью стало не полное переписывание, а дополнение и оптимизация имеющейся реализации.

Для этого были внесены изменения в структуру кода, направленные на улучшение читаемости и производительности. Эти изменения были оформлены в виде pull request на GitHub, адресованного к основному репозиторию "python-rsa" (https://github.com/sybrenstuvel/python-rsa), что позволяет сообществу разработчиков оценить предложенные улучшения и принять решение о их интеграции в основной код. Все представлено на рисунке 6.

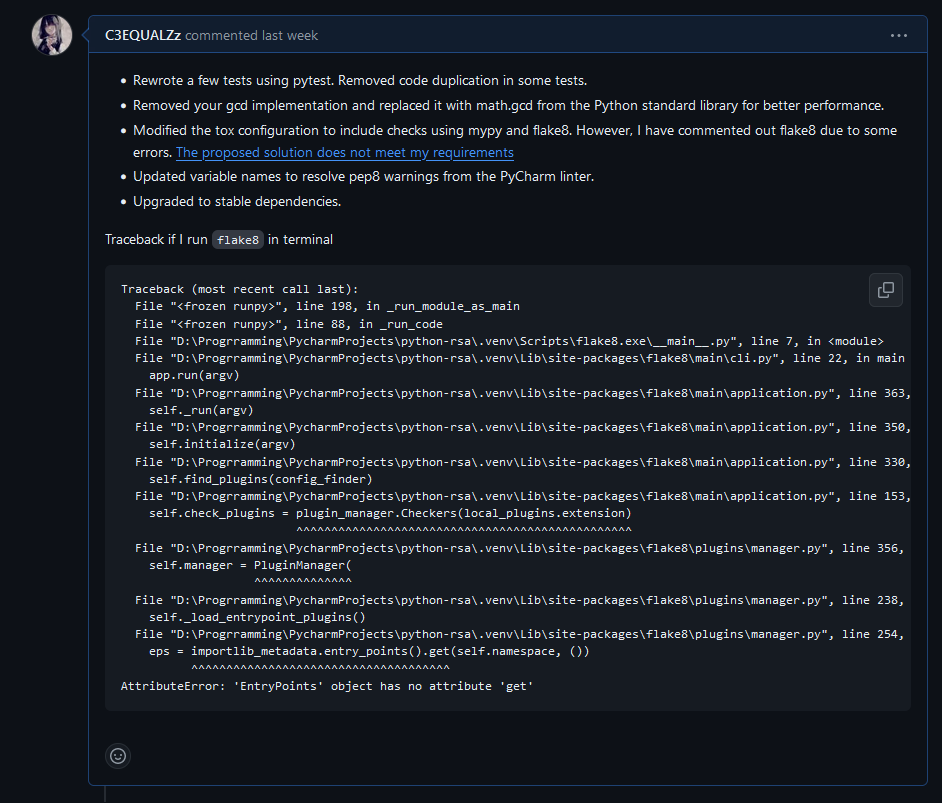


Рисунок 6 – Запрос на слияние на GitHub



### 2.3 Конфигурационные файлы

В данном разделе мы подробно рассмотрим механизмы настройки, используемые в нашем проекте. Начнем с tox.ini, который является ключевым элементом для управления процессами тестирования. Этот файл позволяет настраивать различные аспекты тестирования, включая выбор версий Python для выполнения тестов. В нашем проекте мы выбрали диапазон версий Python от 3.8 до 3.12, чтобы обеспечить максимальную совместимость и кроссплатформенность наших тестов. Конфигурация, представленная на рисунке 7, демонстрирует, как именно настроены эти параметры.

Файл .ini предоставляет нам возможность не только указывать зависимости, необходимые для запуска тестов, но и определять конкретные задачи или команды, которые должны быть выполнены перед или после процесса тестирования. Это может включать в себя такие действия, как установка зависимостей, очистка временных данных или запуск дополнительных скриптов для подготовки тестовой среды.

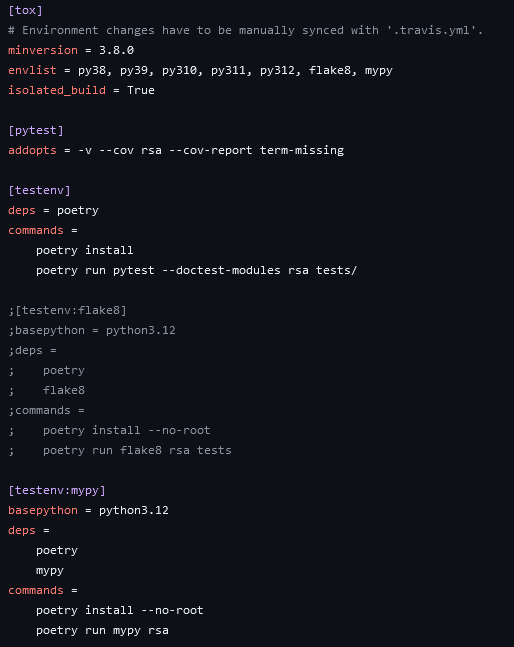


Рисунок 7 – Файл tox.ini

GitHub Actions служит мощным инструментом для автоматизации рабочих процессов в рамках CI/CD пайплайнов. Он позволяет разработчикам интегрировать различные этапы разработки в единый, автоматизированный поток работы, что существенно повышает эффективность и надежность процесса разработки. Используя GitHub Actions, мы можем автоматически запускать тесты, собирать проекты, развертывать их в облачных средах и даже генерировать документацию непосредственно из исходного кода.

В контексте проекта использование GitHub Actions для запуска тестов является ключевым шагом в обеспечении качества программы. Мы разработали специализированный YAML файл, представленный на рисунке 8, который описывает рабочий процесс для автоматического запуска тестов.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 8 – Файл tests.yml

Рассмотрим теперь параметры логирования, которые заданы в json файле. Здесь используется современный подход к логированию, поэтому большая часть настроек была вынесена. Настройки представлены на рисунке 9.

Логирование — это процесс записи сообщений о работе программы для дальнейшего анализа. Сообщения могут включать информацию о текущем состоянии программы, возникших ошибках, предупреждениях, информации для отладки и другой полезной информации. Логи помогают разработчикам понимать, что происходило в системе в момент возникновения проблемы.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 9 – Файл logger\_config.json

### 2.4 **Логика создания ключей**

Для начала пользователь выбирает длину ключа, которую он хочет создавать для кодирования сообщения. Дальше действуем, как показано на рисунке 10.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 10 – Пример создания ключа

Рассмотрим функцию new\_keys, которая представлена на рисунке 11, она генерирует новые открытые и закрытые ключи RSA и возвращает их в виде кортежа объектов PublicKey и PrivateKey*.* Здесь есть параметры по умолчанию, которые пользователь сам может определить:

* 1. number\_of\_bits: количество битов для хранения модуля n.
  2. accurate: если True, n будет точно иметь заданное количество битов.
  3. pool\_size: Количество процессов для генерации простых чисел.
  4. exponent: Экспонента для ключа.
  5. n\_primes: Количество простых чисел, составляющих модуль.

В свою очередь можно заметить, что в коде есть вызов gen\_keys, все аргументы передаются в данную функцию. Данная функция представлена на рисунке 12, она генерирует ключи RSA заданной длины. Можно заметить, что функция возвращает простые числа *p*, *q*, открытый ключ и закрытый ключ *d*.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 11 – Функция *new\_keys*

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 12 – Функция gen\_keys

Можно по вызовам дальше увидеть, что вызывается функция calculate\_keys\_custom\_exponent, которая рассчитывает открытый *e* и закрытый *d* ключи для алгоритма RSA на основе двух простых чисел *p* и *q*, а также заданной экспоненты. Данная функция представлена на рисунке 13.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 13 – Функция calculate\_keys\_custom\_exponent

В итоге мы получаем соответствующие ключи, которые позволяют нам шифровать наши сообщения. Пример использования представлен ниже на рисунке 14.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 14 – Пример для шифрования сообщения

Логика шифрования сообщения представлена ниже на рисунке 15, здесь есть логика перевода из байтов в целые числа, а ещё наоборот.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 15 – Функция для шифрования текста

# Заключение

В процессе выполнения задания на учебную практику была поставлена задача реализовать программное средство, использующее криптографический алгоритм RSA.

Во время выполнения задания был изучен и реализован криптографический алгоритм RSA, а также обнаружены модификации данного шифра. Так же был проведен анализ алгоритма и дана оценка его криптостойкости.

Изучение и реализация алгоритма позволили расширить знания в области алгоритмов и криптографии. Освоены принципы работы алгоритма, изучены недостатки и достоинства.

Подытожив, можно сделать вывод, что сам алгоритм не считается новым и в ближайшие 3–5 лет будет не актуален. Однако, по сей день, он может быть полезным инструментом для защиты конфиденциальной информации.

# Перечень использованных информационных ресурсов

1. **Гатченко Н. А, Исаев А. С, Яковлев А.Д.** Криптографическая защита информации. — Санкт-Петербург : НИУ ИТМО 2012. – 142 с. books.ifmo.ru : сайт. - Криптографическая защита информации. – URL: https://books.ifmo.ru/file/pdf/929.pdf (дата обращения 18.06.2025) – Текст: непосредственный.
2. **Рыжков Г., Волкова И. Б.** RSA – алгоритм : исследовательский проект. – Хабаровск : Академия педагогических проектов Российской Федерации – 9 с. педпроект.рф : сайт. - RSA – алгоритм. – URL: https://педпроект.рф/edu-05-2024-pb-154747/ (дата обращения 18.06.2025). – Текст : электронный.
3. SecurityLab.ru: Информационный портал по безопасности : сайт. –Квантовые компьютеры могут в любой момент взломать RSA-шифрование. –URL: https://www.securitylab.ru/news/535527.php (дата обращения 18.06.2025). – Текст: электронный.
4. **Сущевский Д. Г, Панченко О. В, Кугураков В. Н.** Cовременные криптосистемы и их особенности. – Казань : Вестник Казанского технологического университета. – 4 c. kpfu.ru : сайт. - Cовременные криптосистемы и их особенности. – URL: https://repository.kpfu.ru/?p\_id=125683 (дата обращения 18.06.2025). – Текст : электронный.
5. Encyclopedia.kaspersky.ru: База знаний и глоссарий: сайт. – Симметричное шифрование. – URL: https://encyclopedia.kaspersky.ru/glossary/symmetric-encryption/ (дата обращения 18.06.2025). – Текст : электронный.
6. Encyclopedia.kaspersky.ru: База знаний и глоссарий : сайт. – Асимметричное шифрование. – URL: https://encyclopedia.kaspersky.ru/glossary/asymmetric-encryption/ (дата обращения 18.06.2025). – Текст : электронный.

# Приложение А

**Техническое задание на программное средство**

|  |  |
| --- | --- |
| «СОГЛАСОВАННО» | «УТВЕРЖДЕНО» |
| Руководитель проекта | Зав. кафедрой «КБИС» |
| / О. С. Бурякова / | \_\_\_\_\_\_\_\_\_/ О.А. Сафарьян / |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025г. | «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025г. |

**А.1 Введение**

Наименование разрабатываемого программного средства: «Программная реализация криптографического алгоритма RSA»

Область применения:

* лабораторные работы по предмету алгоритмы и структуры данных,

криптография;

* личное пользование;
* защита данных.

**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 шифровании и дешифровании сообщения, дешифрованного или же зашифрованного алгоритмом RSA. Изучение криптографического алгоритма RSA для решения различных прикладных задач в программировании.

Эксплуатационное назначение программного средства заключается в проведении лабораторных занятий по информационной безопасности в вузе.

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

**А.4.1 Требование к функциональным характеристикам**

Программа корректно использует алгоритмы шифрования и дешифрования RSA

Состав выполняемых функций:

* работать под управлением ОС Windows 7, Windows 8/8.1, а также

Windows 10;

* шифрование и расшифровывание алгоритмом RSA.

Входные данные:

* сообщение и длина ключа.

Выходные данные:

* зашифрованный и расшифрованный текст;

Данные программой хранятся до следующего использование программы или выхода из нее.

**А.4.2 Требование надежности**

Разрабатываемая программа должна удовлетворять следующим требованиям к надежности: отказы программы вследствие внутренних ошибок 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:

* 10 Мб 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.x требуется.

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

Программное средство требует семейство операционных систем GNU/Linux для взаимодействия с пользователем через терминал.

**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.

**А.6 Стадии и этапы разработки**

Ознакомление с теоретическим материалом (от 10.06.25 до 15.06.25);

Оформление программной реализации (от 16.06.25 до 29.06.25);

Испытание программного средства (от 29.06.25 до 3.07.25).

**А.7 Порядок и контроль приемки**

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

Разработчик технического задания: Ковалев Данил Петрович

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2025г \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

# Приложение Б

**Листинг 1**

Б.1 – Реализация «key.py»:

**import abc**

**import** **itertools**

**import** **math**

**import** **threading**

**import** **typing**

**import** **warnings**

**import** **rsa.core** **as** **core\_namespace**

**import** **rsa.helpers** **as** **helpers\_namespace**

**import** **rsa.logic**

**import** **rsa.pem**

**import** **rsa.utils** **as** **utils\_namespace**

**import** **rsa.utils.prime**

**import** **rsa.utils.randnum**

DEFAULT\_EXPONENT: typing.Final[int] = **65537**

T = typing.TypeVar("T", bound="AbstractKey")

**class** **AbstractKey**(metaclass=abc.ABCMeta):

"""Abstract superclass for private and public keys."""

\_\_slots\_\_ = ("n", "e", "blind\_factor", "blind\_factor\_inverse", "mutex")

**def** **\_\_init\_\_**(self, n: int, e: int) -> None:

self.n = n

self.e = e

# These will be computed properly on the first call to blind().

self.blind\_factor = self.blind\_factor\_inverse = -**1**

# Used to protect updates to the blinding factor in multi-threaded

# environments.

self.mutex = threading.Lock()

**@classmethod**

**@abc.abstractmethod**

**def** **\_load\_pkcs1\_pem**(cls: typing.Type[T], keyfile: bytes) -> T:

"""Loads a key in PKCS#1 PEM format, implement in a subclass.

:param keyfile: contents of a PEM-encoded file that contains

the public key.

:type keyfile: bytes

:return: the loaded key

:rtype: AbstractKey

"""

**@classmethod**

**@abc.abstractmethod**

**def** **\_load\_pkcs1\_der**(cls: typing.Type[T], keyfile: bytes) -> T:

"""Loads a key in PKCS#1 PEM format, implement in a subclass.

:param keyfile: contents of a DER-encoded file that contains

the public key.

:type keyfile: bytes

:return: the loaded key

:rtype: AbstractKey

"""

**@abc.abstractmethod**

**def** **\_save\_pkcs1\_pem**(self) -> bytes:

"""Saves the key in PKCS#1 PEM format, implement in a subclass.

:returns: the PEM-encoded key.

:rtype: bytes

"""

**@abc.abstractmethod**

**def** **\_save\_pkcs1\_der**(self) -> bytes:

"""Saves the key in PKCS#1 DER format, implement in a subclass.

:returns: the DER-encoded key.

:rtype: bytes

"""

**@classmethod**

**def** **load\_pkcs1**(cls: typing.Type[T], keyfile: bytes, file\_format: str = "PEM") -> T:

"""Loads a key in PKCS#1 DER or PEM format.

:param keyfile: contents of a DER- or PEM-encoded file that contains

the key.

:type keyfile: bytes

:param file\_format: the format of the file to load; 'PEM' or 'DER'

:type file\_format: str

:return: the loaded key

:rtype: AbstractKey

"""

methods = {

"PEM": cls.\_load\_pkcs1\_pem,

"DER": cls.\_load\_pkcs1\_der,

}

method = cls.\_assert\_format\_exists(file\_format, methods)

**return** method(keyfile)

**@staticmethod**

**def** **\_assert\_format\_exists**(

file\_format: str, methods: typing.Mapping[str, typing.Callable]

) -> typing.Callable:

"""Checks whether the given file format exists in 'methods'."""

**try**:

**return** methods[file\_format]

**except** **KeyError** **as** ex:

formats = ", ".join(sorted(methods.keys()))

**raise** **ValueError**(

f"Unsupported format: {file\_format}, try one of {formats}"

) **from** **ex**

**def** **save\_pkcs1**(self, file\_format: typing.Literal["PEM", "DER"] = "PEM") -> bytes:

"""Saves the key in PKCS#1 DER or PEM format.

:param file\_format: the format to save; 'PEM' or 'DER'

:type file\_format: str

:returns: the DER- or PEM-encoded key.

:rtype: bytes

"""

methods = {

"PEM": self.\_save\_pkcs1\_pem,

"DER": self.\_save\_pkcs1\_der,

}

method = self.\_assert\_format\_exists(file\_format, methods)

**return** method()

**def** **blind**(self, message: int) -> typing.Tuple[int, int]:

"""Performs blinding on the message.

:param message: the message, as integer, to blind.

:return: tuple (the blinded message, the inverse of the used blinding factor)

The blinding is such that message = unblind(decrypt(blind(encrypt(message))).

See https://en.wikipedia.org/wiki/Blinding\_%28cryptography%29

"""

blind\_factor, blind\_factor\_inverse = self.\_update\_blinding\_factor()

blinded = (message \* pow(blind\_factor, self.e, self.n)) % self.n

**return** blinded, blind\_factor\_inverse

**def** **unblind**(self, blinded: int, blind\_factor\_inverse: int) -> int:

"""Performs blinding on the message using random number 'blind\_factor\_inverse'.

:param blinded: the blinded message, as integer, to unblind.

:param blind\_factor\_inverse: the factor to unblind with.

:return: the original message.

The blinding is such that message = unblind(decrypt(blind(encrypt(message))).

See https://en.wikipedia.org/wiki/Blinding\_%28cryptography%29

"""

**return** (blind\_factor\_inverse \* blinded) % self.n

**def** **\_initial\_blinding\_factor**(self) -> int:

**for** \_ **in** range(**1000**):

blind\_r = rsa.utils.randnum.randint(self.n - **1**)

**if** rsa.utils.prime.are\_relatively\_prime(self.n, blind\_r):

**return** blind\_r

**raise** **RuntimeError**("unable to find blinding factor")

**def** **\_update\_blinding\_factor**(self) -> typing.Tuple[int, int]:

"""Update blinding factors.

Computing a blinding factor is expensive, so instead this function

does this once, then updates the blinding factor as per section 9

of 'A Timing Attack against RSA with the Chinese Remainder Theorem'

by Werner Schindler.

See https://tls.mbed.org/public/WSchindler-RSA\_Timing\_Attack.pdf

:return: the new blinding factor and its inverse.

"""

**with** self.mutex:

**if** self.blind\_factor < **0**:

# Compute initial blinding factor, which is rather slow to do.

self.blind\_factor = self.\_initial\_blinding\_factor()

self.blind\_factor\_inverse = rsa.helpers.common.inverse(self.blind\_factor, self.n)

**else**:

# Reuse previous blinding factor.

self.blind\_factor = pow(self.blind\_factor, **2**, self.n)

self.blind\_factor\_inverse = pow(self.blind\_factor\_inverse, **2**, self.n)

**return** self.blind\_factor, self.blind\_factor\_inverse

**class** **PublicKey**(AbstractKey):

"""Represents a public RSA key.

This key is also known as the 'encryption key'. It contains the 'n' and 'e'

values.

Supports attributes as well as dictionary-like access. Attribute access is

faster, though.

>>> PublicKey(5, 3)

PublicKey(5, 3)

>>> key = PublicKey(5, 3)

>>> key.n

5

>>> key['n']

5

>>> key.e

3

>>> key['e']

3

"""

\_\_slots\_\_ = ()

**def** **\_\_getitem\_\_**(self, key: str) -> int:

**return** getattr(self, key)

**def** **\_\_repr\_\_**(self) -> str:

**return** f"PublicKey({self.n}, {self.e})"

**def** **\_\_getstate\_\_**(self) -> typing.Tuple[int, int]:

"""Returns the key as tuple for pickling."""

**return** self.n, self.e

**def** **\_\_setstate\_\_**(self, state: typing.Tuple[int, int]) -> None:

"""Sets the key from tuple."""

self.n, self.e = state

AbstractKey.\_\_init\_\_(self, self.n, self.e)

**def** **\_\_eq\_\_**(self, other: typing.Any) -> bool:

**if** other **is** None:

**return** False

**if** **not** isinstance(other, PublicKey):

**return** False

**return** self.n == other.n **and** self.e == other.e

**def** **\_\_ne\_\_**(self, other: typing.Any) -> bool:

**return** **not** (self == other)

**def** **\_\_hash\_\_**(self) -> int:

**return** hash((self.n, self.e))

**@classmethod**

**def** **\_load\_pkcs1\_der**(cls, keyfile: bytes) -> "PublicKey":

"""Loads a key in PKCS#1 DER format.

:param keyfile: contents of a DER-encoded file that contains the public

key.

:return: a PublicKey object

First let's construct a DER encoded key:

>>> import base64

>>> b64der = 'MAwCBQCNGmYtAgMBAAE='

>>> der = base64.standard\_b64decode(b64der)

This loads the file:

>>> PublicKey.\_load\_pkcs1\_der(der)

PublicKey(2367317549, 65537)

"""

**from** **pyasn1.codec.der** **import** decoder

**from** **rsa.core.classes** **import** AsnPubKey

private, \_ = decoder.decode(keyfile, asn1Spec=AsnPubKey())

**return** cls(n=int(private["modulus"]), e=int(private["publicExponent"]))

**def** **\_save\_pkcs1\_der**(self) -> bytes:

"""Saves the public key in PKCS#1 DER format.

:returns: the DER-encoded public key.

:rtype: bytes

"""

**from** **pyasn1.codec.der** **import** encoder

**from** **rsa.core.classes** **import** AsnPubKey

# Create the ASN object

asn\_key = AsnPubKey()

asn\_key.setComponentByName("modulus", self.n)

asn\_key.setComponentByName("publicExponent", self.e)

**return** encoder.encode(asn\_key)

**@classmethod**

**def** **\_load\_pkcs1\_pem**(cls, keyfile: bytes) -> "PublicKey":

"""Loads a PKCS#1 PEM-encoded public key file.

The contents of the file before the "-----BEGIN RSA PUBLIC KEY-----" and

after the "-----END RSA PUBLIC KEY-----" lines is ignored.

:param keyfile: contents of a PEM-encoded file that contains the public

key.

:return: a PublicKey object

"""

der = rsa.pem.load\_pem(keyfile, "RSA PUBLIC KEY")

**return** cls.\_load\_pkcs1\_der(der)

**def** **\_save\_pkcs1\_pem**(self) -> bytes:

"""Saves a PKCS#1 PEM-encoded public key file.

:return: contents of a PEM-encoded file that contains the public key.

:rtype: bytes

"""

der = self.\_save\_pkcs1\_der()

**return** rsa.pem.save\_pem(der, "RSA PUBLIC KEY")

**@classmethod**

**def** **load\_pkcs1\_openssl\_pem**(cls, keyfile: bytes) -> "PublicKey":

"""Loads a PKCS#1.5 PEM-encoded public key file from OpenSSL.

These files can be recognised in that they start with BEGIN PUBLIC KEY

rather than BEGIN RSA PUBLIC KEY.

The contents of the file before the "-----BEGIN PUBLIC KEY-----" and

after the "-----END PUBLIC KEY-----" lines is ignored.

:param keyfile: contents of a PEM-encoded file that contains the public

key, from OpenSSL.

:type keyfile: bytes

:return: a PublicKey object

"""

der = rsa.pem.load\_pem(keyfile, "PUBLIC KEY")

**return** cls.load\_pkcs1\_openssl\_der(der)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *№ строки* | *Формат* | *Обозначения* | | | | *Наименование* | | | | *Кол-во листов* | | *Примечание* | |
|  |  |  | | | | *Документация общая* | | | |  | |  | |
|  |  |  | | | |  | | | |  | |  | |
| *1* | *А4* |  | | | | *Задание на практику* | | | | *1* | |  | |
| *2* | *А4* | *ПП.990000.000* | | | | *Отчет по учебной практике (включает 15 рисунков)* | | | | *29\** | |  | |
|  |  |  | | | |  | | | |  | |  | |
|  |  |  | | | | *Документация приложений* | | | |  | |  | |
|  |  |  | | | |  | | | |  | |  | |
| *3* | *А4* | *ПП.990000.000* | | | | *Приложение А-Техническое задание на программное средство* | | | | *4* | |  | |
| *4* | *А4* | *ПП.990000.000* | | | | *Приложение Б-Листинг кода* | | | | *8* | |  | |
|  |  |  | | | |  | | | |  | |  | |
|  |  |  | | | |  | | | |  | |  | |
|  |  |  | | | |  | | | |  | |  | |
|  |  |  | | | |  | | | |  | |  | |
|  |  |  | | | |  | | | |  | |  | |
|  |  |  | | | |  | | | |  | |  | |
|  |  |  | | | |  | | | |  | |  | |
|  |  |  | | | |  | | | |  | |  | |
|  |  |  | | | |  | | | |  | |  | |
|  |  |  | | | |  | | | |  | |  | |
|  |  |  |  |  | *ПП.990000.000* | | | | | | | | |
|  |  |  |  |  |
| *Изм* | *Лист* | *№ Докум.* | *Подп.* | *Дата* |
| *Разраб.* | | *Ковалев Д. П.* |  |  | *Отчет по практической подготовке при проведении производственной практики (ведомость отчета)* | | *Лит.* | | | | *Лист* | | *Листов* |
| *Руков.пр* | | *Бурякова О. С.* |  |  |  |  |  | | *42* | | *1* |
|  | |  |  |  | *ДГТУ*  *Группа ВКБ32* | | | | | | |
| Н. Контр | |  |  |  |
| *Утверд.* | | *Сафарьян О. А.* |  |  |

\*количество листов отчета, исключая приложения