# Практические задания по курсу «Криптосистемы с открытым ключом» (весенний семестр 2019–2020 учебного года)

И. В. Чижов

23 марта 2020 г.

## 1. Кодер паролей для их защиты при хранении в записных книжках - passcoder.

Требуется спроектировать и реализовать систему защиты паролей при их хранении в записных книжках или блокнотах.

### 1.1. Требования к программе.

1) Должны поддерживаться три режима работы: режим регистрации пользователя, режим шифрования пароля, режим расшифрования пароля;

2) В режиме регистрации пользователя программа должны генерировать открытый и секретный ключи для шифрования паролей криптосистемой Рабина. Открытый ключ должен быть подписан ЭЦП RSA на секретном ключе программы.

3) Предусмотреть процедуру смены ключа подписи программы.

4) Предусмотреть защиту секретного ключа пользователя паролем.

5) В режиме шифрования пароля запускается интерактивный сеанс, в котором программа запрашивает путь к файлу с открытым ключом и просит ввести пароль. В результате программа выводит пароль, зашифрованный на открытом ключе пользователя криптосистемой Рабина.

6) Шифр-текст пароля выводится в кодировке base32, в которой удобно хранить его в записной книге или в блокноте.

7) В режиме расшифрования пароля программа запрашивает путь к файлу с секретным ключом и шифр-текст пароля в кодировке base32. На выходе программа выводит на экран пароль. В случае, если секретный ключ не соответствует шифр-тексту, то необходимо вывести сообщение об этом пользователю.

8) (ОПЦИЯ). Реализовать хранение зашифрованных паролей в текстовом файле следующего формата. Каждая запись — с новой строки. Запись содержит три поля: (<Порядковый номер записи>, <Описание пароля, например «Пароль от почты gmail»>, <шифр-текст пароля в кодировке base32>). Требуется реализовать возможность отобразить пользователю файл паролей и выбор пользователем пароля для расшифрования с помощью указания порядкового номера пароля.

9) Предпочтительный язык программирования — python. В случае использования другого языка нужно создать простую сборочную системы на основе make-файлов для трёх операционных системы: Windows 10, MacOS 10.15+, Linux.

## 1.2. Требования к выполнению задания и принцип его оценивания.

1) Все технические решения нужно обосновать с точки зрения надёжности и стойкости к известным автору атакам (вклад в оценку — 45%). Предполагается, что к программе будет приложен файл пояснительной записки с обоснованием выбора технических решений: в первую очередь протоколов и форматов хранения ключа, шифр-текста. Объём пояснительной записки — достаточный для обоснования решений. Параметры криптосистем подбираются в процессе технического проектирования авторами исходя из эргономики и обеспечения стойкости в 80 битов. Выбор параметров обосновать в пояснительной записки.

2) Оценивается правильность работы программы (вклад в оценку — 50%) и её эргономика (вклад в оценку — 5%).

3) Плагиат кода или пояснительной записки — обнуляет выполнение задачи. И она не засчитывается. Плагиатом не является любое заимствование с указанием его авторства. Заимствованный участок кода не включается в оценивание. Например, Вы не смогли реализовать функцию и заимствовали её у друга, указав это в программе. Реализация этой функции исключается из вклада в оценку правильности работы программы на основе «важности» (принципиальности) этой

функции в программе. Важность вещь разумно-субъективная, определяемая лектором курса. Так, если Вы заимствовали реализацию криптосистемы. То, ясно, что это важная функция, т. к. она проверяет знания по теме курса. И её вклад может быть до 20% вклада кода программы в оценку. В этом случае, этот вклад вычитается из общего вклада в 50% и получается 30% вклада программы. Если же была заимствована функция реализации интерфейса командной строки, то она не относится к тем функциям, которые призваны проверить знания в области криптографии с открытым ключом, поэтому её принципиальность может быть оценена не более, чем в 5%.

4) Для получения оценки «отлично» нужно выполнить задание более, чем на 79%, оценка «хорошо» ставится за выполнение на 65% – 79%, за 50% – 65% — тройка, и при выполнении задания менее, чем на 50% ставится — «неудовлетворительно».

# 2. Анализатор надёжности выбора параметров криптосистемы RSA: RSASecAnalyzer

Требуется реализовать систему анализа корректности выбранных параметров в криптосистеме RSA.

## 2.1. Требования к программе.

1) На вход программе должен подаваться открытый ключ криптосистемы RSA: $(e, N)$

2) Программа должна применить к открытому ключу следующие известные атаки: атаки факторизации модуля $(p-1)$-методом Полларда, $\rho$-методом Полларда, атаку восстановления шифр–текста, использующую маленький модуль шифрования, атаку Винера, атаку факторизации, использующую итерацию процесса шифрования.

3) По каждой атаке пользователю в файл выводится отчёт о применимости этой атаки к открытому ключу криптосистемы RSA, который был подан на вход пользователю; по каждой атаке выводятся подробно значения выбора различных параметров, с которыми проводилось тестирование.

4) Необходимо разработать временные ограничения проведения каждой атаки, т.е. программа не должна работать бесконечно долго в процессе анализа; временные ограничения должны изменяться пользовате-

лем до запуска анализа; параметры методов должны подстраиваться под эти временные ограничения.

5) Необходимо также реализовать режим тестирования, который по размеру задачи и списку «применимости» атак, генерирует заведомо слабые секретные ключи, которые могут быть взломаны с использованием списка атак. Пример:
**Вход:** 100, [«$(p-1)$-метод», «атака Винера»]
**Выход:** открытый ключ $(e, N)$ и секретный ключ $d$, где $N$ – имеет размер 200 битов, причём по открытому ключу достаточно легко восстановить секретный ключ, если применить $(p-1)$-метод или атаку Винера.

6) Предпочтительный язык программирования — python. В случае использования другого языка нужно создать простую сборочную системы на основе make-файлов для трёх операционных системы: Windows 10, MacOS 10.15+, Linux.

## 2.2. Требования к выполнению задания и принцип его оценивания.

1) Оценивается правильность работы программы (вклад в оценку — 75%) и её эргономика (вклад в оценку — 25%).

2) Плагиат кода — обнуляет выполнение задачи. И она не засчитывается. Плагиатом не является любое заимствование с указанием его авторства. Заимствованный участок кода не включается в оценивание. Например, Вы не смогли реализовать функцию и заимствовали её у друга, указав это в программе. Реализация этой функции исключается из вклада в оценку правильности работы программы на основе «важности» (принципиальности) этой функции в программе. Важность вещь разумно-субъективная, определяемая лектором курса. Так, если Вы заимствовали реализацию метода Полларда. То, ясно, что это важная функция, т. к. она проверяет знания по теме курса. И её вклад может быть до 20% вклада кода программы в оценку. В этом случае, этот вклад вычитается из общего вклада в 50% и получается 30% вклада программы. Если же была заимствована функция реализации интерфейса командной строки, то она не относится к тем функциям, которые призваны проверить знания в области криптографии с открытым ключом, поэтому её принципиальность может быть оценена не более, чем в 5%.

3) Для получения оценки «отлично» нужно выполнить задание более, чем на 79%, оценка «хорошо» ставится за выполнение на 65% – 79%,

за 50% – 65% — тройка, и при выполнении задания менее, чем на 50% ставится — «неудовлетворительно».

# 3. Анализатор надёжности выбора параметров криптосистемы Эль-Гамаля: ElGamalSecAnalyzer

Требуется реализовать систему анализа корректности выбранных параметров в криптосистеме Эль-Гамаля, построенной на основе мультипликативной группы простого поля $GF^{\star}(p)$.

## 3.1. Требования к программе.

1) На вход программе должен подаваться открытый ключ криптосистемы Эль-Гамаля: $(y = g^x \mod p, g, p)$

2) Программа должна применить к открытому ключу следующие известные атаки: метод «большой шаг – малый шаг», $\rho$-методом Полларда, в котором цикл ищется методом Госпера, метод Полига–Хэллмана.

3) По каждой атаке пользователю в файл выводится отчёт о применимости этой атаки к открытому ключу криптосистемы Эль-Гамаля, который был подан на вход пользователю; по каждой атаке выводятся подробно значения выбора различных параметров, с которыми проводилось тестирование.

4) Необходимо разработать временные ограничения проведения каждой атаки, т. е. программа не должна работать бесконечно долго в процессе анализа; временные ограничения должны изменяться пользователем до запуска анализа; параметры методов должны подстраиваться под эти временные ограничения.

5) Необходимо также реализовать режим тестирования, который по размеру задачи генерирует заведомо слабые секретные ключи, которые могут быть взломаны с использованием метода Полига-Хэллмана.

6) Предпочтительный язык программирования — python. В случае использования другого языка нужно создать простую сборочную системы на основе make-файлов для трёх операционных системы: Windows 10, MacOS 10.15+, Linux.

## 3.2. Требования к выполнению задания и принцип его оценивания.

Требования полностью совпадают с требованиями раздела 2.2.

# 4. Криптоанализ криптосистемы Меркля–Хэллмана: MHcrack.

Требуется реализовать систему взлома криптосистемы Меркля-Хэллмана.

## 4.1. Требования к программе.

1) Программа должна поддерживать работу в трёх режимах: генерация ключей, шифрование, дешифрование (взлом)

2) Режим генерации ключей должен по значению параметра стойкости выдавать секретный и открытый ключи необходимого размера;

3) Ключи должны записывать в файл; Форматы файлов остаются на усмотрение разработчика.

4) В режиме шифрования на вход программе подаётся путь к файлу ключей и строка открытого текста (UTF8-последовательность символов). Предусмотреть как интерактивный метод ввода параметров, так и через аргументы программы. Выходом в этом режиме является файл, содержащий шифр-текст входной последовательности в кодировке base32.

5) В режиме взлома на вход криптосистемы подаётся открытый ключ и шифр-текст (последовательность символов в кодировке base32). Далее программа с использованием LLL-алгоритма строит приведённый базис специальной решётки и выдаёт возможное значение открытого текста в UTF8-кодировке.

6) Предпочтительный язык программирования — python. В случае использования другого языка нужно создать простую сборочную системы на основе make-файлов для трёх операционных системы: Windows 10, MacOS 10.15+, Linux.

## 4.2. Требования к выполнению задания и принцип его оценивания.

Требования полностью совпадают с требованиями раздела 2.2.