МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

01.03.02 ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ И КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

БАКАЛАВРСКАЯ РАБОТА

на тему Реализация алгоритма шифрования данных с использованием эллиптических кривых

Студент Г. Ю. Березовский _____

Руководитель О.В. Лелонд _____

Допустить к защите

Заведующий кафедрой, к.т.н, доцент, А.В. Очеповский _____

«_____» _____20_____г.

Тольятти 2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав.кафедрой «Прикладная математика и информатика»
_____А.В.Очеповский

«____»_____2016 г.

ЗАДАНИЕ
на выполнение бакалаврской работы

Студент Березовский Глеб Юрьевич

1. Тема Реализация алгоритма шифрования данных с использованием эллиптических кривых

2. Срок сдачи студентом законченной бакалаврской работы
17.06. 2016

3. Исходные данные к бакалаврской работе: методы криптографической защиты информации, алгоритмы шифрования данных.

4. Содержание бакалаврской работы (перечень подлежащих разработке вопросов, разделов): изучение основ эллиптической криптографии; анализ алгоритмов шифрования; реализация алгоритма шифрования данных на основе эллиптических кривых.

5. Ориентировочный перечень графического и иллюстративного материала: презентация, включающая блок-схемы алгоритмов, диаграммы классов, показывающие структуру приложения.

6. Дата выдачи задания «11» января 2016 г.

Руководитель выпускной квалификационной работы	_____	О.В. Лелонд
Задание принял к исполнению	_____	Г.Ю. Березовский

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**
Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав.кафедрой «Прикладная
математика и информатика»
_____А.В.Очеповский

«____» _____2016 г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения бакалаврской работы

Студента Березовский Глеб Юрьевич
по теме Реализация алгоритма шифрования данных с использованием эллиптических кривых

Наименование раздела работы	Плановый срок выполнения раздела	Фактический срок выполнения раздела	Отметка о выполнении	Подпись руководителя
Изучение эллиптических кривых	1.03.2016	1.03.2016	выполнено	
Анализ методов криптографической защиты информации	11.03.2016	11.03.2016	выполнено	
Реализация алгоритма шифрования данных на основе эллиптических кривых	25.03.2016	25.03.2016	выполнено	
Тестирование приложения	1.04.2016	1.04.2016	выполнено	
Оформление пояснительной записки	6.04.2016	6.04.2016	выполнено	
Создание презентации	21.04.2016	21.04.2016	выполнено	
Предварительная	31.05.2016	31.05.2016	выполнено	

защита				
Проверка на наличие заимствований (плагиата) в системе «Антиплагиат ВУЗ»	17.06.2016	17.06.2016	выполнено	
Сдача на кафедру комплекта документов для защиты	17.06.2016	17.06.2016	выполнено	
Защита бакалаврской работы	29.06.2016	29.06.2016	выполнено	

Руководитель выпускной квалификационной работы _____ О.В. Лелонд

Задание принял к исполнению _____ Г.Ю. Березовский

АННОТАЦИЯ

к бакалаврской работе на тему «Реализация алгоритма шифрования данных с использованием эллиптических кривых»

Работа выполнена студентом Тольяттинского государственного университета Березовским Глебом Юрьевичем.

Цель работы – реализация алгоритма шифрования данных на основе эллиптических кривых.

Объектом исследования являются методы защиты информации от несанкционированного доступа.

Предмет исследования – алгоритмы шифрования на основе эллиптических кривых.

Для достижения поставленной цели в работе решаются следующие задачи:

1. Изучение основных методов защиты информации.
2. Анализ алгоритмов защиты данных.
3. Реализация алгоритма шифрования данных на основе эллиптических кривых.

Бакалаврская работа состоит из введения, трех глав, заключения.

В главе 1 рассматриваются теоретические аспекты эллиптической криптографии.

В главе 2 проводится анализ современных методов криптографической защиты информации и выбирается алгоритм для реализации.

В главе 3 описывается процесс реализации алгоритма шифрования данных. Проводится тестирование данного приложения.

В заключении подводятся итоги исследования, формируются окончательные выводы по рассматриваемой теме.

Выпускная квалификационная работа содержит пояснительную записку объемом 40 страниц, включая 7 рисунков, одну таблицу и 31 формулу, одно приложение и список литературы из 34 источников.

Оглавление

Введение 4

Глава 1. Сведения из эллиптической криптографии 7

1.1 Математические основы эллиптических кривых 7

1.2 Эллиптическая криптография 9

1.3 Стандарты для эллиптических кривых 11

1.4 Достоинства и недостатки эллиптической криптографии 11

Глава 2 Анализ криптографических методов защиты информации 14

2.1 Классификация криптографических методов защиты информации 14

2.2 Протокол Диффи-Хелмана 14

2.3 Схема Эль-Гамаля 15

2.3.1 Генерация открытого и закрытого ключей в схеме Эль-Гамаля 15

2.3.2 Шифрование в схеме Эль-Гамаля 16

2.3.3 Дешифрование в схеме Эль-Гамаля 17

2.4 Криптографическая программа PGP 18

2.5 Алгоритм цифровой подписи ЭЦП ГОСТ Р 34.10-2001 20

2.6 Алгоритм Чандрасекхара 21

2.7 Задача дискретного логарифмирования на эллиптических кривых и пути ее решения. 24

2.7.1 Задача дискретного логарифмирования на эллиптических кривых 24

2.7.2 Алгоритмы решения задачи дискретного логарифмирования на эллиптических кривых 24

Глава 3. Реализация алгоритма шифрования данных 28

3.1 Реализация алгоритма Чандрасекхара 28

3.1.1 Структура класса Main 30

3.1.2 Структура класса User_A 32

3.1.3 Структура класса User_B 33
3.3 Тестирование приложения 35
Заключение 36
Список используемой литературы 38
Приложение А. Реализация алгоритма 41

Введение

Жизнеспособность общества все больше зависит от развития информационной среды. Информация является важной частью в функционировании государственных и общественных институтов, в жизни каждого человека. В нынешних условиях сформировался новый вид трудовой деятельности, связанный с хранением, распространением и получением информации.

Информатизация ведет к созданию общего мирового информационного пространства, к унификации информационных технологий различных стран мира.

Новые технологии открывают большие возможности. Вместе с тем катастрофически возрастает цена потерь в случае нештатной ситуации или снижения надежности систем обработки и передачи информации. Можно наблюдать, что в настоящее время все более зримо проявляется зависимость экономики от надежности систем защиты информации.

В нынешних условиях защита информации становится все более актуальной и одновременно все более сложной проблемой. Это обусловлено как массовым применением методов автоматизированной обработки данных, так и широким распространением методов и средств несанкционированного доступа к информации. Поэтому важную роль в организации противодействия потенциальным угрозам занимает подход, при котором средства защиты информации используются комплексно, каждое в соответствии со своим предназначением.

Внедрение и активное использование современных информационных технологий существенно повысили уязвимость информации, циркулирующей в современных информационно-телекоммуникационных системах.

Несанкционированное искажение, копирование, уничтожение информации в настоящее время затрагивает не только процессы, относящиеся к сфере государственного управления, но и интересы физических лиц.

С каждым днём увеличивается объем информации и увеличивается её спрос, а значит, и растёт её ценность, поэтому возрастают требования к её защите.

Актуальность темы бакалаврской работы объясняется необходимостью, располагая небольшими вычислительными ресурсами, противостоять современным способам несанкционированного доступа к информации.

Целью бакалаврской работы является реализация алгоритма шифрования данных на основе эллиптических кривых.

Объект исследования – алгоритмы защиты информации от несанкционированного доступа.

Предметом исследования являются алгоритмы шифрования на основе эллиптических кривых.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- изучить основные методы защиты информации;
- провести анализ алгоритмов, основанных на эллиптической криптографии;
- реализовать алгоритм шифрования информации на высокоуровневом языке программирования;
- произвести тестирование алгоритма.

В процессе работы изучены современные методы, средства и способы защиты информации.

Работа состоит из введения, трех глав и заключения.

В первой главе рассматриваются теоретические аспекты эллиптической криптографии: определение, математические основы, достоинства и недостатки. Во второй главе производится анализ существующих алгоритмов защиты информации, рассматриваются варианты атак алгоритмов, основанных

на эллиптических кривых. В третьей главе реализуется алгоритм шифрования информации с помощью эллиптических кривых.

Глава 1. Сведения из эллиптической криптографии

1.1 Математические основы эллиптических кривых

Эллиптические кривые - кубические кривые на проективной плоскости, задаваемые уравнением 3-й степени с коэффициентами из поля K и «точкой на бесконечности» [8].

Уравнение кривой имеет вид

$$y^2 = x^3 + ax + b \ , \tag{1.1}$$

где a, b – действительные числа.

Примеры эллиптических кривых представлены на рисунке 1.1.

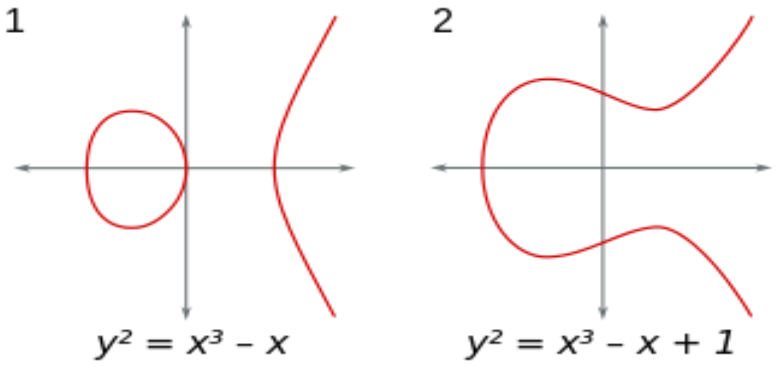

Рисунок 1.1 – Эллиптические кривые

Уравнение вида (1.1) называется уравнением Вейерштрасса.

В данной бакалаврской работе рассматриваются эллиптические кривые с характеристикой поля, равной 0.

Эллиптические кривые бывают сингулярные и несингулярные. Несингулярные кривые – это эллиптические кривые с отличным от нуля дискриминантом Δ

$$\Delta = (4a^3) + 27b^2 \neq 0 \ , \tag{1.2}$$

где a и b - коэффициент уравнения (1.1).

Использование сингулярных кривых в криптографии нецелесообразно, так как это снижает криптостойкость алгоритмов и протоколов [8].

На рисунке 1.2 представлен пример несингулярной кривой.

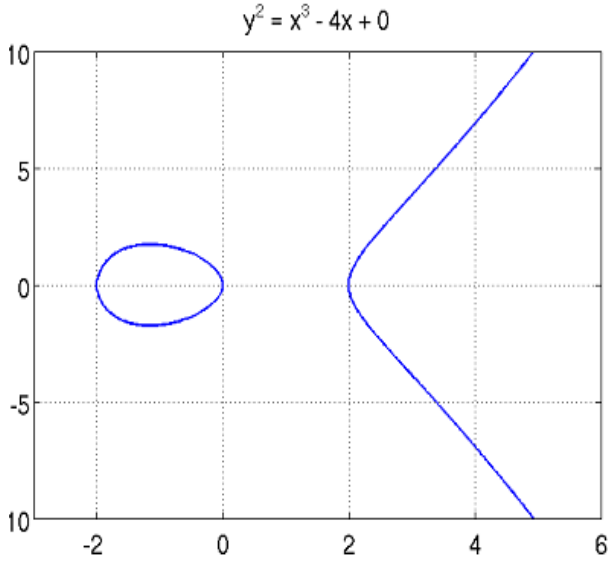

Рисунок 1.2 – Несингулярная кривая

Точки на эллиптической кривой можно складывать. Запишем сложение двух точек в виде формулы:

$$P + Q = -R, \tag{1.3}$$

где *P*, *Q*, *R* – точки эллиптической кривой.

Пусть (x_p, y_p) - координаты точки P, а (x_Q, y_Q) координаты точки Q.

Определим коэффициент α формулой:

$$\alpha = \frac{y_Q - y_P}{x_Q - x_P}. \tag{1.4}$$

Координаты точки $P + Q$ равны:

$$x_{P+Q} = \alpha^2 - x_P - x_Q, \tag{1.5}$$

$$y_{P+Q} = -y_P + (x_P - x_R). \tag{1.6}$$

Правило сложения точек продемонстрировано на рисунке 1.3

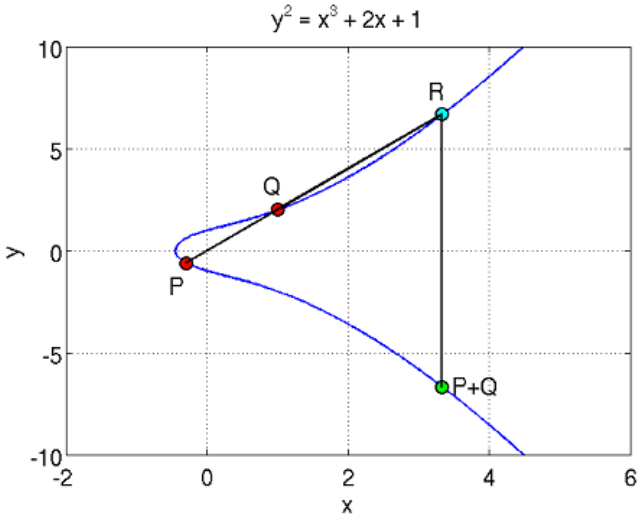

Рисунок 1.3 – Сложение точек

1.2 Эллиптическая криптография

Эллиптическая криптография – раздел криптографии, который основан на эллиптических кривых над конечными полями [8]. Эллиптические кривые было предложено использовать в криптосистемах в 1985 году Нилом Коблицем и Виктором Миллером.

Эллиптические кривые используются в алгоритмах цифровой подписи, обмена ключами и асимметричного шифрования.

Эллиптическая криптография применяется во многих системах:

- Bitcoin – это первая и самая известная из множества криптовалют, символ и флагман криптовалютного мира. ECDSA – акроним для алгоритма цифровой подписи с эллиптическими кривыми. Это процесс, который использует эллиптические кривые и конечные поля, чтобы “подписать” данные таким образом, что третьи лица могут легко проверить подлинность подписи, но при этом сам подписывающий оставляет за собой эксклюзивную возможность создавать подписи. В случае Bitcoin “данные”, которые подписываются, – это транзакция, которая передает право собственности на Bitcoin [25];
- SSH - сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и

туннелирование TCP-соединений. SSH использует эллиптическую криптографию для авторизации пользователей. Эксперты подсчитали, что из 12 миллионов хостов, поддерживающих SSH, в 10,3% случаев реализован ECDSA (алгоритм цифровой подписи, основанный на эллиптических кривых) для аутентификации и в 13,8% − ECDH (аналог протокола Диффи-Хеллмана с использованием эллиптической криптографии) для обмена ключами [8] ;

- TLS - криптографический протокол, обеспечивающий защищённую передачу данных между узлами в сети Интернет. Протокол TLS предназначен для предоставления трёх услуг всем приложениям, работающим над ним, - шифрование, аутентификация и сохранение [25] .

Эллиптические кривые находят широкое применение в алгоритмах шифрования данных:

- алгоритм обмена ключами позволяет двум сторонам получить общий секретный ключ с использованием незащищенного от прослушивания, но защищённого от подмены канала связи. Полученный ключ можно использовать для обмена сообщениями;
- алгоритм цифровой подписи нужен для аутентификации пользователей, передачи электронных документов и других данных. Если электронный документ подписан сертифицированной цифровой подпись, то он имеет юридическую значимость. Известные программы, работающие с электронной цифровой подписью, – КриптоПро, VipNet;
- алгоритм ассиметричного шифрования или алгоритм с открытым ключом. Это такой способ шифрования данных, при котором открытый ключ передается по открытому каналу (не скрывается) и используется для проверки электронной подписи и шифрования данных. Для дешифрования же и создания электронной подписи используется второй ключ, секретный.

1.3 Стандарты для эллиптических кривых

Для эллиптической криптографии используются не все эллиптические кривые. От выбора эллиптической кривой зависит криптостойкость и уязвимость алгоритмов. Эллиптическая кривая должна обладать нужными параметрами.

На территории Российской Федерации введены стандарты для формирования и проверки электронной подписи. В соответствии с этим была создана методика выбора эллиптических кривых, основанная на следующих ключевых моментах [24]:

1. Для повышения эффективности операций в базовых полях вычетов определяющие поля простые числа выбираются близкими к степеням двойки (в нашем случае к $2^{511}, 2^{512}$) [5, 11].
2. Выбираются эллиптические кривые с группой точек, имеющей порядок, равный простому числу [5, 11].
3. Рассматриваются только кривые с коэффициентом *a* в уравнении Вейерштрасса (1.1), равным -3.
4. Учитывается требование возможности определенной верификации "случайности" кривой [5, 11].

1.4 Достоинства и недостатки эллиптической криптографии

Основные плюсы эллиптической криптографии:

- гораздо меньшая длина ключа по сравнению с «классической» асимметричной криптографией [21];
- скорость работы эллиптических алгоритмов гораздо выше, чем у классических. Это объясняется как размерами поля, так и применением более близкой для компьютеров структуры бинарного конечного поля [21];
- из-за маленькой длины ключа и высокой скорости работы алгоритмы асимметричной криптографии на эллиптических кривых

могут использоваться в смарт-картах и других устройствах с ограниченными вычислительными ресурсами [21];

- для задачи дискретного логарифмирования на эллиптических кривых не существует субэкспоненциальных алгоритмов решения [21];

Основные минусы эллиптической криптографии:

- если появятся алгоритмы решения задач дискретного логарифмирования на эллиптических кривых, то это будет означать крах эллиптической криптографии [21];
- в эллиптической криптографии огромное количество тонкостей, начиная с выбора эллиптической кривой и заканчивая генерацией ключей. Все их необходимо учитывать. При неправильном выборе повышается уязвимость алгоритмов и повышается вероятность ошибки [21].

В работе [34] проводится сравнение производительности и использования памяти ECC(алгоритм на основе эллептических кривых) и RSA (криптографический алгоритм с открытым ключом). Вычисления производятся на двух процессорах – Chipcon CC1010 и Atmel ATmega128. Данные вычисления представлены на таблице 1.1.

Таблицы 1.1 – Сравнение алгоритмов.

Алгоритм	Atmega128@8Мгц			CC10 @ 14,7456 МГц		
	Время, с	Память для хранения данных данные, байтов	Память под код, байтов	Время, с	Память для хранения данных, байтов	Память под код, байтов
ECC-160	0.81	282	3682	4.58	180+86	2166
ECC-192	1.24	336	3972	7.56	216+102	2152
ECC-224	2.19	422	4812	11.98	259+114	2214
RSA-1024 шифрование	0.43	542	1073	> 4.48		
RSA-1024 расшифрование	10.99	930	6292	≈106.66		
RSA-2048 шифрование	1.94	1332	2854			
RSA-2048 расшифрование	83.26	1853	7736			

В данной главе были рассмотрены математические основы эллиптических кривых, описаны известные системы, в которых используются эллиптическая криптография, отмечены ее достоинства и недостатки.

На основании всего вышесказанного можно сделать вывод о том, что повсеместный переход на эллиптическую криптографию возможен, но не является необходимым.

Глава 2 Анализ криптографических методов защиты информации

2.1 Классификация криптографических методов защиты информации

Криптографические методы защиты информации входят в систему технических методов защиты информации [20]. Данные методы подразделяются на следующие группы:

1. Методы хеширования данных, предназначенные для проверки целостности электронных документов без чтения самой информации в этих документах [21].
2. Методы шифрования информации с помощью криптографического преобразования, превращающего начальный текст в нечитаемую последовательность символов [21].
3. Методы построения электронной цифровой подписи, предназначенные для защиты информации от изменения, связывания источника информации (автора) с самой информацией [21].

Основные методы шифрования и построения электронной цифровой подписи заключаются в перестановках и сдвигах исходного текста, гаммировании (наложении одного текста на другой), подстановке (отображениях одних алфавитов на другие) или комбинации данных методов. [20].

2.2 Протокол Диффи-Хелмана

Обмен ключами с использованием эллиптических кривых может быть выполнен следующим образом. Сначала выбирается простое число $p \approx 180$ и параметры a и b для уравнения эллиптической кривой. Это задает множество $E_p(a,b)$ точек кривой. Затем в $E_p(a,b)$ выбирается генерирующая точка $G = (x_1 . y_1)$. При выборе G важно, чтобы наименьшее значение n, при котором $n \times G = 0$, оказалось очень большим простым числом (порядка $2^{511}, 2^{512}$).

Параметры $E_p(a,b)$ и G криптосистемы являются параметрами, известными всем участникам обмена информацией.

Обмен ключами между пользователями A и B производится по следующей схеме:

1. Участник A выбирает целое число n_A, меньшее n. Это число является закрытым ключом участника A. Затем участник A вычисляет открытый ключ $P_A = n_a \times G$, который представляет собой некоторую точку на $E_p(a,b)$.

2. Точно так же участник B выбирает закрытый ключ n_B и вычисляет открытый ключ P_A.

3. Участники обмениваются открытыми ключами, после чего вычисляют общий секретный ключ K. Участник A: $K = n_a \times P_B$. Участник B: $K = n_B \times P_A$.

Следует заметить, что общий секретный ключ представляет собой пару чисел. Если данный ключ предполагается использовать в качестве сеансового ключа для алгоритма симметричного шифрования, то из этой пары необходимо создать одно значение.

2.3 Схема Эль-Гамаля

Схема Эль-Гамаля — это стандарт на электронную цифровую подпись. Он сравнительно прост для рассмотрения.

2.3.1 Генерация открытого и закрытого ключей в схеме Эль-Гамаля

Схема Эль-Гамаля основывается на задаче возведения в степень. Обычно выбирается большое простое число p и рассматриваются операции в поле (или в мультипликативной группе) по модулю p. Выбирается случайное число g, которое является генератором (генератор мультипликативной группы — это элемент, возводя который во все степени можно получить все элементы

группы). В данном случае все числа, которые взаимно просты с p, будут являться генераторами. Далее вычисляется число y:

$$y = g^x \bmod p, \tag{2.1}$$

где x – случайное число из диапазона от 1 до $(p - 1)$.

Тогда открытым ключом будет являться набор (y, g, p) а закрытым — (x, g, p).

Сложность восстановления закрытого ключа по открытому каналу связана с так называемой задачей дискретного логарифма. Допустим, злоумышленнику известен открытый ключ (y, g, p):

$$g^x \bmod p \tag{2.2}$$

Чтобы восстановить x из этого выражения, нужно взять дискретный логарифм:

$$x = \log_g y \mod p \tag{2.3}$$

Данная задача является настолько же сложной, насколько и задача факторизации. Не существует эффективных полиномиальных алгоритмов для вычисления числа x, хотя существуют субэкспоненциальные алгоритмы и алгоритмы для квантового компьютера, которые, возможно, способны решить эту задачу.

2.3.2 Шифрование в схеме Эль-Гамаля

Схема шифрования осуществляется следующим образом.

Выбирается случайное число k, находящееся в пределах от 1 до $(p - 1)$, взаимно простое с $(p - 1)$, после чего вычисляется число a:

$$a = g^k \bmod p \tag{2.4}$$

Далее вычисляется число b:

$$b = y^k M \bmod p, \tag{2.5}$$

где M - сообщение.

Числа, a и b образуют подпись. Для систем с открытыми ключами подпись сообщения не передается по каналу связи, так как размер ее слишком большой. Предварительно вычисляется хеш-функция $h = H(M)$ от сообщения, и подпись вычисляется уже от этой хеш-функции. Хеш-функция имеет фиксированные размеры, и за счёт этого электронно-цифровая подпись будет достаточно малой. После этого происходит передача сообщения и подписи вычисленной хэш-функции.

В случае с шифрованием выбирается случайный ключ, который называется сессионным ключом S. Этот сессионный ключ зашифровывается схемой Эль-Гамаля, а закрытый текст получается путём шифрования сообщения M на выбранном случайном ключе. Тогда передаваться будут зашифрованный сессионный ключ и зашифрованное сообщение. При этом, если сессионный ключ имеет размер 128 бит, то размер зашифрованного ключа будет не меньше 256 бит.

Так как число p выбирается не меньше 1000 бит длиной, то размер зашифрованного сообщения будет не меньше 2000 бит. В схеме Эль-Гамаля всегда размер зашифрованного сообщения в 2 раза больше, чем размер исходного.

При каждом шифровании получается новый результат. В RSA одинаковые сообщения при шифровании преобразовывались в одну и ту же последовательность символов. В схеме Эль-Гамаля так не получится: одинаковые сообщения, если им соответствуют разные числа k, будут давать разные числа a и b на выходе.

2.3.3 Дешифрование в схеме Эль-Гамаля

В данной схеме дешифрование происходит быстро:

$$M = \frac{b}{a} \bmod p, \tag{2.6}$$

где M – символ сообщения,

a, b – числа, выбранные при шифровании,

p – простое число.

Эта криптосистема очень проста с математической точки зрения. Восстановление сообщения без знания ключа происходит следующим образом: злоумышленник знает открытый ключ (g, p, y) и числа a и b, которые были переданы по каналу связи. Для восстановления нужно найти

$$M = b y^{-k} \bmod p \ , \tag{2.7}$$

где

$$k = \log_g a \bmod p . \tag{2.8}$$

Для нахождения k нужно вычислить дискретный логарифм. Оказывается, задача дискретного логарифмирования, возникающая при восстановлении закрытого ключа из открытого и восстановления сообщения из переданного текста, является достаточно сложной.

2.4 Криптографическая программа PGP

PGP – это криптографическая программа с высокой степенью надежности. Она позволяет обмениваться информацией в режиме полной конфиденциальности. Данная программа была впервые написана Филом Циммерманом примерно в 1995-м году. Он собирался её продать, но в тот момент правительство США решило внедрить на законодательном уровне так называемую инициативу Clipper. Эта инициатива обязывала всех производителей криптографического аппаратного и программного обеспечения внедрять в свои системы backdoor — специальную микросхему от правительства США. Данная микросхема обладала таким шифром, который очень легко вскрыть при наличии специального кода, депонированного в бюро шифров и находящихся в специальном хранилище. Предполагалось, что доступ к этому хранилищу выдается только по судебному распоряжению. После этого PGP стало открытой программой. Исходный код программы был опубликован в Интернете. Главным преимуществом данной программы является то, что при обмене конфиденциальной информацией не нужно передавать тайные ключи,

так как принцип работы данной программы – публичная криптография или обмене открытым ключами.

В программе PGP используется принцип двух взаимосвязанных ключей — открытого и закрытого. Открытый ключ находится в открытом доступе, он передается тем пользователям, с которыми вы хотите обмениваться конфиденциальной информацией, а к закрытому ключу есть доступ только у вас.

Алгоритм работы PGP:

1. Сжимается текст, что увеличивает скорость и надежность шифрования.
2. Генерируется сессионный ключ, который является случайным числом.
3. Шифруется информация с помощью сессионного ключа.
4. Шифруется сессионный ключ с помощью открытого ключа.
5. Передается сообщение, которое содержит зашифрованную информацию и зашифрованный ключ.
6. Программа PGP получает зашифрованную информацию.
7. Дешифруется сессионный ключ с помощью закрытого ключа.
8. Расшифровывается информация с помощью сессионного ключа.

Ключи, используемые в программе PGP, находятся на жестком диске вашего компьютера в зашифрованном виде в виде двух файлов. Эти файлы называются кольцами. При потере ключей вы не сможете работать.

Преимуществом данной программы является применение «хэш-функции», которая анализирует изменение информации. При любом изменении информации «хэш функция» оповестит нас. С помощью «хэш функции» и закрытого ключа информация подписывается электронной подписью. При получении информации получателем PGP проверяет подпись.

2.5 Алгоритм цифровой подписи ЭЦП ГОСТ Р 34.10-2001

На эллиптических кривых построен алгоритм проверки ЭЦП ГОСТ Р 34.10 – 2001, являлся стандартом РФ в области ЭЦП. Схема этого алгоритма приведена на рис. 2.1.

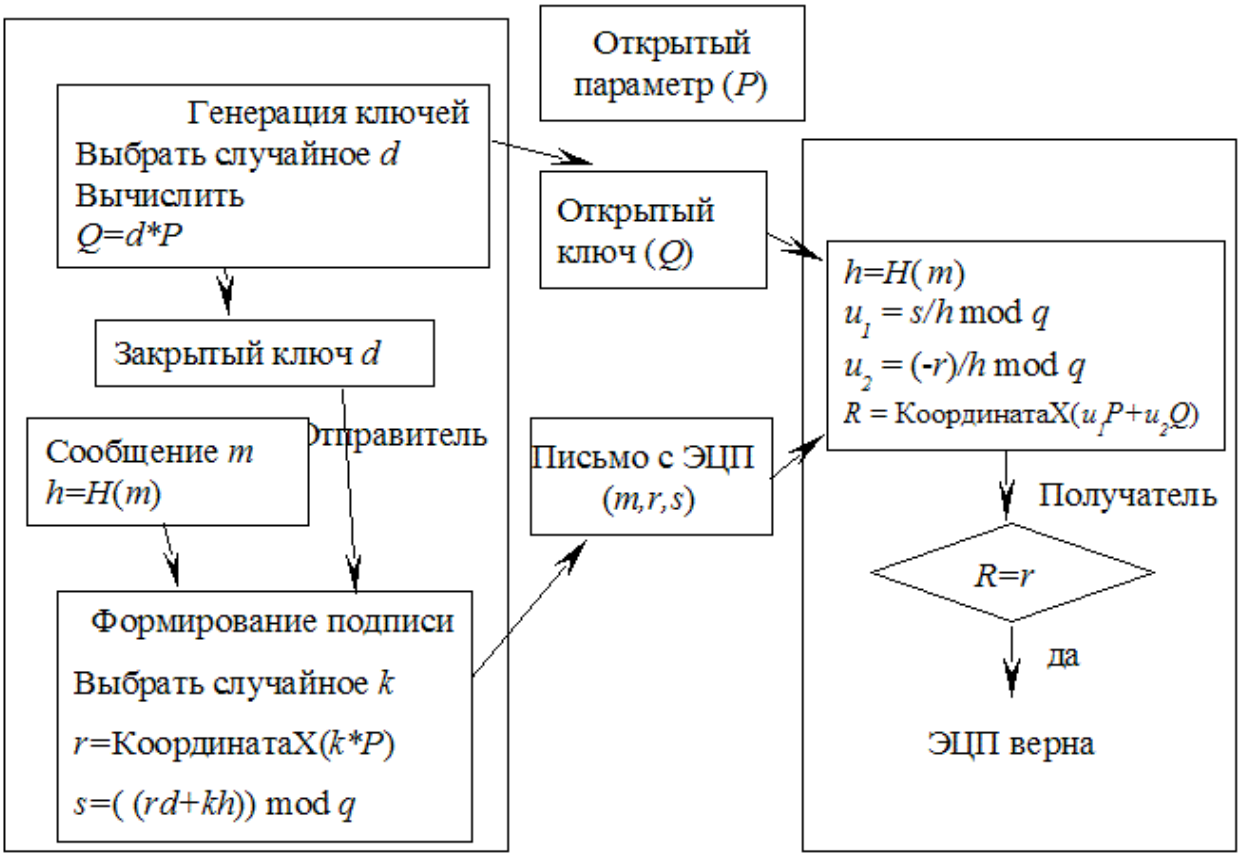

Рисунок 2.1– Схема алгоритма ЭЦП ГОСТ Р 34.10-2001

Основным достоинством криптосистем на основе эллиптических кривых является то, что они обеспечивают надежность, адекватную классическим криптосистемам RSA или Эль-Гемаль, на существенно меньших по длине ключах, что положительно отражается на времени кодирования и декодирования. Криптосистемы цифровой подписи на основе эллиптических кривых с длиной ключа 160 бит имеют одинаковую стойкость с криптосистемами DSA и Эль-Гемаля с длиной ключа 1024 бита.

Электронная цифровая подпись – это эффективное средство защиты информации от модификации, которое переносит свойства реальной подписи

под документом в область электронного документооборота. В основу ЭЦП положены такие криптографические методы, как асимметричное шифрование и хэш-функции.

2.6 Алгоритм Чандрасекхара

Алгоритм Чандрасекхара является ассиметричным алгоритмом. Алгоритм выполняется следующим образом. Сначала выбирается уравнение эллиптической кривой $E_p(a,b)$, где p – простое число. Затем находятся все точки, принадлежащие кривой и составляется таблица *N*, в которой каждой точке присваивается символ. После этого выбирается случайная точка *C* на эллиптической кривой. Параметры $E_p(a,b)$ и C являются параметрами, известными всем участникам, таблица *N* известна только тем участникам, которые обмениваются информацией.

Обмен информацией между пользователями A и B производится по следующей схеме.

Участник A:

- выбирает случайное число a, где $a > p$;
- выбирает точку *A*, принадлежащую $E_p(a,b)$;
- вычисляет $A_1 = a(C + A)$;
- вычисляет $A_2 = A \times a$.

Точки *A1* и *A2* являются открытыми ключами, точка *A* – закрытой.

Участник B:

- выбирает случайное число b, где $b > p$;
- выбирает точку *B*, принадлежащую $E_p(a,b)$;
- вычисляет $B_1 = b(C + B)$;
- вычисляет $B_2 = B \times b$.

Точки *B1* и *B2* являются открытыми ключами, точка *B* – закрытой.

Участник A:

- вычисляет $A_b = a \times B_2$.

Точка A_b является открытым ключом.

Участник B:

- вычисляет $B_a = b \times A_2$.

Точка B_a является открытым ключом.

После вычисления параметров происходит шифрование информации участником *B*. Каждый символ сообщения *M* кодируется двумя символами из таблицы *N:*

- выбирается случайное число γ;
- вычисляется $E_1 = \gamma \times C$;
- вычисляется $E_2 = M + (b + y) A_1 - \gamma A_2 + A_b$.

Для расшифровки информации участником *A* используется формула

$$M = E_2 - (a \times E_1 + a \times B_1 + B_a). \quad (2.9)$$

Безопасность криптографии, основанной на эллиптических кривых, зависит от значения *p*, где *p* - это порядок конечного поля, и случайной точки C на эллиптической кривой. Эллиптические параметры кривой, использующиеся в криптографии, должны быть тщательно подобраны, для того чтобы противостоять всем известным атакам в эллиптической криптографии. Алгоритм Чандрасекхара обеспечивает конфиденциальность информации. При кодировке информации он кодирует каждый символ сообщения двумя символами, и одинаковые символы сообщения могут быть закодированы по-разному, так как данный алгоритм при шифровании каждого символа использует случайное число. Из-за этого линейный криптоанализ алгоритма является весьма затруднительным. Метод шифрования, предложенный Чандрасекхаром, обеспечивает достаточную защиту от несанкционированного доступа при относительно низких вычислительных мощностях.

Описание алгоритма представлено на рисунке 2.2.

Рисунок 2.2 – Схема алгоритма Чандрасекхара

2.7 Задача дискретного логарифмирования на эллиптических кривых и пути ее решения.

2.7.1 Задача дискретного логарифмирования на эллиптических кривых

Для взлома алгоритмов, основанных на эллиптических кривых, нужно решить задачу дискретного логарифмирования на эллиптических кривых.

Задача логарифмирования на эллиптической кривой формулируется следующим образом: даны две точки *R, Q* некоторой эллиптической кривой, необходимо найти число k такое, что при умножении на него точки Q результатом получим точку R:

$$R = kQ, \quad R,Q \in E_p(a,b) \tag{2.10}$$

Для обеспечения надежной безопасности задача дискретного логарифмирования должна быть максимально сложной. Отметим, что сложность дискретного логарифмирования в значительной степени зависит от параметров задачи (уравнения эллиптической кривой, порядка циклической группы и др.).

Для повышения сложности задачи дискретного логарифмирования порядок *r* конечного поля должен быть большим простым числом. В противном случае можно найти «проекции» логарифма в циклических подгруппах простых порядков и восстановить логарифм по этим «проекциям» по китайской теореме об остатках.

Алгоритмы дискретного логарифмирования можно разделить на универсальные, не зависящие от эллиптической кривой, и специальные, которые используются с определенными эллиптическими кривыми.

2.7.2 Алгоритмы решения задачи дискретного логарифмирования на эллиптических кривых

Существует различные алгоритмы решения:

- алгоритм Полига-Хеллмана — алгоритм вычисления дискретного логарифма. Предположим, что n — количество точек эллиптической кривой. Пусть число n раскладывается на простые

множители $p_1, p_2, \ldots . p_n$. Суть метода в том, чтобы найти дискретные логарифмы по модулю чисел p_i, а затем получить общее решение с помощью «китайской теоремы об остатках». Атака позволяет свести проблему дискретного логарифмирования в поле порядка n к той же задаче, но в поле с гораздо меньшим порядком p. Для того чтобы противостоять атаке, необходимо просто выбирать эллиптические кривые, количество точек которых делится на очень большое простое число $q \approx n$ [8];

- алгоритм Шенкса, более известный как «шаги младенца/шаги гиганта». Для группы размером n формируется соответствующая таблица, затем по этой таблице происходит поиск нужного элемента. Сложность алгоритма $O(\sqrt{q})$, где q - количество точек на кривой (или порядок поля) [8] ;
- алгоритм Полларда - универсальный алгоритм дискретного логарифмирования на эллиптической кривой. Он обладает временной сложностью $O(\sqrt{q})$ и емкостной сложностью $O(\sqrt{\log q})$ и не допускает распараллеливания;
- алгоритм встречи на случайном дереве – универсальный алгоритм, обладает временной и емкостной сложностью $O(\sqrt{q \log q})$, но зато допускает распараллеливание для произвольного числа параллельно работающих процессоров;
- субэкспоненциальные методы решения для сингулярных и суперсингулярных кривых. Особые свойства таких кривых позволяют свести задачу дискретного логарифмирования на эллиптической кривой к задаче дискретного логарифмирования в конечном поле. Соответственно, для такого класса кривых стандартные ключи размером в 160-320 бит будут фатально уязвимы, что позволит злоумышленникам вскрыть секретный ключ за относительно небольшое время [9].

Системам с открытыми ключами свойственны две больших уязвимости. Во-первых, криптосистемы основаны на математических задачах, для которых неизвестно, существует решение или нет. Например, в случае с криптосистемами на открытых ключах пока не найдено эффективных решений, но это не означает, что их нет. Также существуют различные сложные проблемы: проблема факторизации, проблема дискретного логарифмирования, проблема дискретного логарифмирования в произвольной мультипликативной группе [15] .

Вторая проблема связана с необходимостью надёжного открытого канала, в котором злоумышленник не имел бы возможности что-то поменять (идеальный канал). В случае криптосистемы на закрытых ключах факт того, что получатель смог расшифровать сообщение, означает, что шифровкой занимался тот, кто обладает секретным ключом. В случае же криптосистем на открытых ключах тот факт, что у получателя получилось расшифровать сообщение, ничего не значит, потому что зашифровать сообщение может кто угодно [15] .

Отметим два важных момента. Во-первых, сложность задачи дискретного логарифмирования на эллиптической кривой не снижается в течение 15 лет с момента создания первой криптосистемы на эллиптических кривых, чего нельзя сказать о сложности задачи дискретного логарифмирования в мультипликативной группе конечного поля.

Во-вторых, логарифмирование на эллиптической кривой метод Полларда или встречи на случайном дереве не допускает предвычислений, уменьшающих сложность задачи, если хотя бы одна из точек P и Q неизвестна. Поэтому время эксплуатации информационной системы может быть продлено заменой персональных секретных и соответствующих открытых ключей. Логарифмирование в конечном поле допускает предвычисления, не зависящие от точек P и Q, то есть замена персонального ключа практически не может продлить срок безопасной эксплуатации системы [8]. Поэтому сложность логарифмирования в группе определяет не время действия персонального

ключа подписи (или подписанного сообщения), а время жизни информационной системы в целом.

В данной главе был произведен анализ алгоритмов шифрования данных. Рассмотрены варианты атак на алгоритмы, основанные на эллиптических кривых.

В результате анализа алгоритмов решения задачи дискретного логарифмирования было выяснено, что взломать алгоритм шифрования на основе эллиптических кривых довольно сложно, если подобраны правильные параметры.

На основе анализа алгоритмов шифрования данных был выбран алгоритм для реализации на высокоуровневом языке программирования, описание реализации алгоритма будет приведено в следующей главе.

Алгоритм был создан ученым индийским ученым Чандрасекхаром. При анализе данного алгоритма были выявлены следующие особенности:

- данный алгоритм является ассиметричным;
- алгоритм обеспечивает шифрование данных и гарантирует безопасную передачу данных;
- каждый участник обмена информацией публикует конкретный открытый ключ для связи с другим конкретным участником, что позволяет обмениваться конфиденциальной информацией в общей беседе;
- каждый символ сообщения зашифрован в виде пары точек на эллиптической кривой. Здесь случайное число γ используется в шифровании каждого символа сообщения, и γ отличается для шифрования каждого символа. Именно поэтому одни и те же символы в пространстве сообщения шифруются разными символами. Из-за этого линейный криптоанализ является весьма затруднительным.

Глава 3. Реализация алгоритма шифрования данных

3.1 Реализация алгоритма Чандрасекхара

Приложение, реализующее данный алгоритм, состоит из четырех классов:

- Main;
- User_A;
- User_B;
- Stroka_encoding.

Принимая в расчет, что наша реализация алгоритма должна работать на многих системах, используя малые вычислительны возможности. Из этого следует что язык программирования, на котором будет реализован алгоритм должен быть:

- прост в изучении;
- обладать способностью кроссплатформенности.

Для реализации алгоритма шифрования данных выбран язык программирования Java.

Приложение было разработано в свободной интегрированной среде разработки под названием Eclipse.

Подробно структуру приложения, реализующее алгоритм Чандрасекхара можно увидеть на рисунке 3.1.

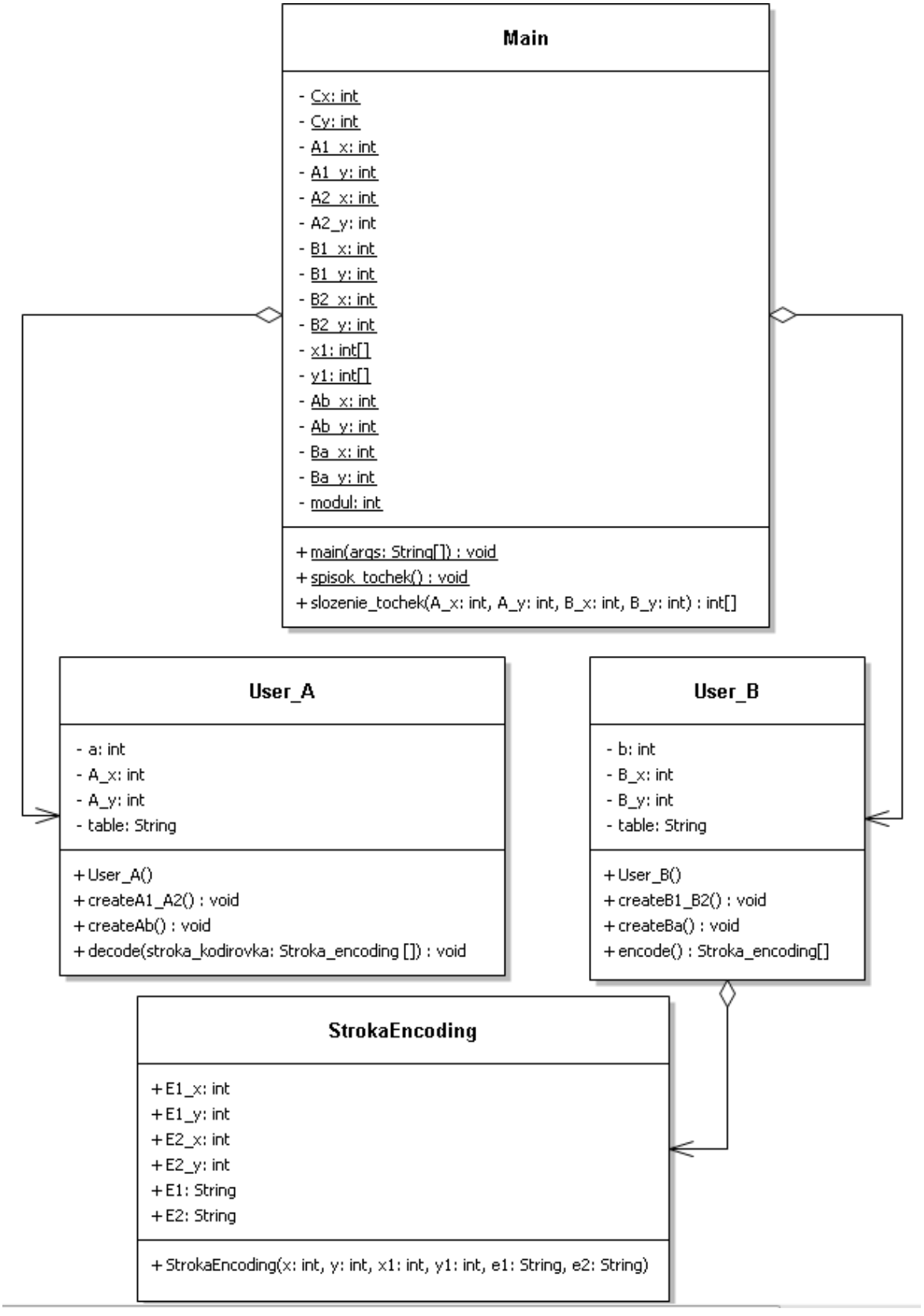

Рисунок 3.1 – Структура приложения

3.1.1 Структура класса Main

Класс Main – это тело основной программы. В данном классе объявляются глобальные переменные, определяются основные функции и процедуры.

Глобальные переменные являются открытыми ключами алгоритма Чандрасекхара.

Глобальные переменные:

public static int Cx – координата x случайной точки на эллиптической кривой.

public static int Cy - – координата y случайной точки на эллиптической кривой.

public static int A1_x – координата x точки эллиптической кривой, определяемая участником A.

public static int A1_y - координата y точки эллиптической кривой, определяемая участником A.

public static int A2_x – координата x второй точки эллиптической кривой, определяемая участником A.

public static int A2_y - координата y второй точки эллиптической кривой, определяемая участником A.

public static int B1_x - координата x точки эллиптической кривой, определяемая участником B.

public static int B1_y - координата y точки эллиптической кривой, определяемая участником B.

public static int B2_x - координата x второй точки эллиптической кривой, определяемая участником B.

public static int B2_y - координата y второй точки эллиптической кривой, определяемая участником B.

public static int[] x1 – массив абсцисс точек принадлежащих, эллиптической кривой.

public static int[] y1 - массив ординат точек принадлежащих, эллиптической кривой.

public static int kol – количество точек удовлетворяющие, уравнению эллиптической кривой.

public static int Ab_x - координата x третьей точки эллиптической кривой, определяемая участником A.

public static int Ab_y - координата y третьей точки эллиптической кривой, определяемая участником A.

public static int Ba_x - координата x третьей точки эллиптической кривой, определяемая участником B.

public static int Ba_y - координата x третьей точки эллиптической кривой, определяемая участником B.

public static int modul - модуль эллиптической кривой.

Основные функции и процедуры класса Main.

spisok_tochek(). Данная процедура с помощью математических действий определяет, какие точки удовлетворяют уравнению эллиптической кривой, и добавляет координаты данных точек в массивы *x1[]* и *y1[]*.

slozenie_tochek(). Эта функция складывает две точки *P(x, y)* и *Q(x1.y1)* на эллиптической кривой и получает координаты третьей точки *R(x2,y2)*, используя данные формулы:

$$P \neq Q,$$

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \ , \tag{3.1}$$

$$y_3 = \frac{y_2 - y_1}{x_2 - x_1} - \times (x_1 - x_3) - y_1 \ , \tag{3.2}$$

$$P = Q \ ,$$

$$x_3 = \left(\frac{3{x_1}^2 - a}{2y_1}\right)^2 - 2x_1, \tag{3.3}$$

$$x_3 = \left(\frac{3x_1^2 - a}{2y_1}\right)^2 \times (x_1 - x_3) - y_1 \ . \qquad (3.4)$$

void main(String[] args) – главная процедура, с которой начинается запуск приложения. Данная процедура создает объекты класса User_A, User_B, вызывает процедуру spisok_tochek() и функцию slozenie_tochek() .Она выполняет основные функции приложения.

3.1.2 Структура класса User_A

Класс User_A выполняет функции по вычислению параметров, которые необходимы для обмена зашифрованной информацией, и расшифровывает закодированные сообщения. Данный класс содержит переменные и процедуры для работы с этим классом. Переменные данного класса являются секретной частью ключа и недоступны другим пользователям.

Переменные данного класса:

int a – случайное число, которое меньше порядка конечного поля.

int A_x – координата x точки на эллиптической кривой, определяемая пользователем A.

int A_y – координаты y точки на эллиптической кривой, определяемая пользователем A.

String table – кодовая таблица, необходимая для раскодировании информации.

Процедуры класса User_A.

createA1_A2() – процедура, которая определяет глобальные переменные A1_x, A1_y, A2_x, A2_y с помощью формул

$$A_1 = a(C + A) \ , \qquad (3.4)$$

где A_1 - точка на эллиптической кривой, определяемая участником A,

C – точка эллиптической кривой,

a – случайное число, определяемое участником A,

$$A_2 = aA. \qquad (3.5)$$

createAb() – процедура, определяющая глобальные переменные Ab_x, Ab_y с помощью формулы

$$A_в = a\,B_2, \tag{3.6}$$

где $A_в$ - точка на эллиптической кривой, определяемая участником *A*,

a – случайное число, определяемое участником *A*,

B_2 – точка на эллиптической кривой, определяемая участником *B*.

decode() – процедура, которая расшифровывает информацию, входным параметром является массив зашифрованных символов. Процедура определяет точки на эллиптической кривой, к которым привязываются символы с помощью переменной String table. После нахождения данных точек процедура расшифровывает сообщение, используя найденные точки и глобальные переменные, которые являются открытой частью ключа. После проведения математических операций процедура выводит расшифрованную строчку.

Данный класс выполняет функции пользователя *A* в алгоритме Чандрасекхара.

3.1.3 Структура класса User_B

Данный класс определяет методы и переменные, которые необходимы для кодирования и раскодирования информации.

Переменные класса:

int b – случайное число, которое меньше порядка конечного поля.

int B_x – координата x точки на эллиптической кривой, определяемая пользователем B.

int B_y – координата y точки на эллиптической кривой, определяемая пользователем B.

String table – кодовая таблица, необходимая для кодирования информации.

Данные переменные являются секретным ключом пользователя B.

Процедуры класса B.

createB1_B2() – процедура, которая определяет глобальные переменные B1_x, B1_y, B2_x, B2_y с помощью формул

$$A_1 = b(C + B) \quad , \tag{3.4}$$

где A_1 –точка на эллиптической кривой, определяемая участником A,

b – случайное число,

B – точка на эллиптической кривой, определяемая участником B,

$$A_2 = bB \quad , \tag{3.5}$$

где A_2 – точка на эллиптической кривой, определяемая участником A.

createAb() –процедура, определющая глобальные переменные Ab_x, Ab_y с помощью формулы

$$B_A = b\,A_2, \tag{3.6}$$

где B_A – точка эллиптической кривой.

decode() – функция, которая шифрует сообщение. Сначала вводится строка для кодировки сообщения String stroka. Каждый символ сообщения шифруется двумя точками на эллиптической кривой. После этим двум точкам присваиваются символы из кодовой таблицы String table. Для сохранения зашифрованной строки функция создает массив объектов класса Stroka_encoding, в объекте хранятся точки эллиптической кривой и зашифрованные символы. Шифруется символы с помощью формул

$$E_1 = \gamma C \ , \tag{3.7}$$

где E_1 – точка эллиптической кривой,

C – точка эллиптической кривой,

$$E_2 = M + (b + \gamma)\,A_1 - \gamma\,A_2 + A_B, \tag{3.8}$$

где E_2 – точка эллиптической кривой,

M – точка эллиптической кривой.

Функция decode() возвращает массив зашифрованных символов в класс Main.

3.3 Тестирование приложения

Тестирования приложения является важной частью реализации алгоритма. Данный алгоритм тестировался со следующими предопределенными параметрами.

1. Эллиптическая кривая E, задавая уравнением

$$y^2 = x^3 + 2 \times x + 9 \bmod 37 \tag{3.9}$$

2. Точка $C(9,4)$ на эллиптической кривой E.

На рисунке 3.2 выводятся результаты работы алгоритма.

Рисунок 3.2 – Консоль приложения

В ходе работы приложения выводится три строки: исходная, зашифрованная, расшифрованная. В результате реализации алгоритма Чандрасекхара появляется файл формата jar, который можно запускать как на мобильных устройствах, так и на компьютере.

В данной главе описывается структура приложения, которое реализует алгоритм шифрования данных на основе эллиптических кривых над конечным полем. Также описывается процесс тестирования приложения с помощью предопределенных параметров. Разработанное приложение используют только стандартные библиотеки.

Заключение

В ходе выполнения бакалаврской работы был рассмотрен алгоритм Чандрасекхара, его теоретические основы, атаки, проводимые на алгоритм и методы противостояния им, проанализированы достоинства и недостатки алгоритма, а также осуществлена реализация данного алгоритма на языке Java.

Алгоритм Чандрасекхара является достаточно надежным методом обмена информацией через незащищенные каналы передачи информации. Так как алгоритм использует малые вычислительные возможности, то сфера его применения очень большая. За долгие годы использования эллиптической криптографии было разработано множество атак, но данный алгоритм при правильно подобранных параметрах защитит информацию. Безопасное шифрование информации является важным аспектом защиты данных. Поэтому алгоритм Чандрасекхара востребован и в наше время.

В работе были рассмотрены теоретические основы алгоритма Чандрасекхара, пошагово описан ход его выполнения.

Также были рассмотрены алгоритмы шифрования данных, которые широко используются для декодирования информации в современном мире.

Кроме того, были проанализированы и сами атаки на алгоритмы, основанные на эллиптической криптографии. Главное внимание было уделено подбору правильных параметров кривой. Были рассмотрены такие варианты атак, как алгоритм Полига-Хеллмана и алгоритм Шенкса. Эти атаки направлены на уменьшение сложности алгоритма.

Была отмечена возможность обмена секретной информацией более чем двумя участниками с помощью алгоритма Чандрасекхара. Это позволяет неограниченному числу пользователей безопасно обмениваться секретной информацией.

В ходе исследования было выяснено, что при правильно подборе эллиптической кривой увеличивается криптостойкость алгоритмов шифрования.

Помимо рассмотрения теоретических аспектов был реализован алгоритм Чандрасекхара. Реализация была осуществлена на языке программирования Java. Проведена проверка реализации на работоспособность. Приложение, которое реализует данный алгоритм, выполняет функции шифрования и дешифрования информации.

Данное приложение было протестировано с помощью предопределенных параметров.

Разработанное приложение может быть улучшено путем:

- добавления интерфейса для пользователя;
- добавления возможности обмениваться конфиденциальной информацией со многими пользователями;
- добавления возможности выбора эллиптической кривой и параметров для шифрования информации;
- сохранения результатов шифрования в удобном формате для передачи пользователям;
- повышения скорости работы приложения с помощью изменения метода сложения точек.

Данное приложение работает при малых вычислительных возможностях электронно-вычислительных устройств. Поэтому приложение может работать как на персональных компьютерах, так и на мобильных устройствах.

Список используемой литературы

1. ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание документа.

2. ГОСТ 7.32-2001. Отчет о научно-исследовательской работе. Структура и правила оформления

3. ГОСТ 7.82-2001. Библиографическая запись. Библиографическое описание электронных ресурсов.

4. ГОСТ Р 7.05-2008 Библиографическая ссылка.

5. Бабаш, А. В. Информационная безопасность и защита информации: учебное пособие/ А. В. Бабаш, П.Н. Башлы, Е. К. Баранова. — М.: РИОР, 2013. — 222 с.

6. Гомес, Ж. Математики, шпионы и хакеры. Кодирование и криптография /Ж. Гомес. — М.: Де Агостини, 2014. —144 с.

7. Грушо, А. А. Теоретические основы компьютерной безопасности: учеб. пособие для вузов / А. А. Грушо. — М.: Академия, 2009. — 272 с.

8. Жданов, О. Н. Применение эллиптических кривых в криптографии: учеб. пособие / О. Н. Жданов, Т. А. Чалкин. — Красноярск: СибГАУ, 2011 — 65 с.

9. Жуков, А. Е. Системы блочного шифрования: учеб. пособие по курсу «Криптографические методы защиты информации» / А.Е. Жуков — М.: Издательство МГТУ им. Н.Э. Баумана, 2013. — 79с.

10. Загинайлов, Ю. Н. Основы информационной безопасности: курс визуальны лекциий: направление подготовки "Информационная безопасность", специальность "Экономическая безопасность"/ Ю.Н. Загинайлов. — М.: Директ-Медиа, 2015 -105 с.

11. Иванов, М. А. Криптографические методы защиты и информации в компьютерных системах и сетях: учеб. пособие / М. А. Иванов, И. В. Чугунков. — М.: МИФИ, 2012 — 400 с.

12. Свинарев, Н.А. Инструментальный контроль и защита информации: учеб. пособие / Н.А. Свинарев, О.В. Ланкин, А.П Данилкин, и др. — Воронеж: ВГУИТ, 2013. — 192с.

13. Кияев, В. И. Безопасность информационных систем: учеб. пособие / В. И. Кияев, Граничин О.Н. — М.: ИНТУИТ, 2016 — 192 с.

14. Лапонина, О. Р. Протоколы безопасного сетевого взаимодействия 2-е издание, исправленное / О. Р. Лапонина. — М.: ИНТУИТ, 2016 — 462 с.

15. Маховенко, Е. Б. Теоретико-числовые методы в криптографии: учеб. пособие / Е. Б. Маховенко. — М.: Гелиос АРВ, 2006. — 320 с.

16. Набебин, А. А. Модулярная арифметика и криптография: учеб. пособие / А. А. Набебин. — М.: Издательный дом МЭИ, 2007.

17. Панасенко, С.П. Алгоритмы шифрования: специальный справочник / С.П. Панасенко. — СПб.: БХВ-Петербург, 2009 — 576 с.

18. Петров, А. А. Компьютерная безопасность. Криптографические методы защиты / А. А. Петров — М.: ДМК, 2010. — 448 с.

19. Романьков, В. А. Введение в криптографию. Курс лекций. 2-е издание, исправленное и дополненное: учебное пособие / В.А. Романьков — М.: Форум, 2012. — 240с.

20. Рубцова, Р. Г. Математические основы защиты информации: учеб. пособие / Р. Г. Рубцова, Ш. Т. Ишмухаметов. — Казань: Казанский федеральный университет, 2012 — 138с.

21. Рябко, Б. Я. Криптографические методы защиты информации: учеб. пособие / Б. Я. Рябко, А. Н. Фионов. — 2-е изд., стер. — М.: Горячая линия – Телеком, 2012. — 229 с.

22. Фороузан, Б. А. Математика криптографии и теория шифрования: учеб. пособие / Б. А. Фороузан. — М.: ИНТУИТ, 2016 — 510 с.

23. Ященко, В. В. Введение в криптографию / В. В. Ященко, Н. П. Варновский, Ю. В. Нестеренко — 4-е изд., доп. М.: МЦНМО, 2012. — 348 с.

24. Нестерова, Л. Ю., Карпенкова Н. В. Создание криптографии с помощью модулярной математики // Молодой ученый. — 2014. — №21.1. — с. 237-240.

25. Gilson, D. Blockchain.info issues refunds to Bitcoin theft victims / D. Gilson, 2013. [Электронный ресурс]: http://www.coindesk.com/blockchain-info-issues-refunds-to-bitcoin-theft-victims/

26. Ch. Suneetha. “Secure key transport in symmetric cryptographic protocols using elliptic curves over finite fields” / Ch. Suneetha, D. Sravana Kumar, A. Chandrasekhar , 2011. [Электронный ресурс]: http://www.academia.edu/19574633/

27. Chandrasekhara, K.R. “Elliptic Curve based authenticated session Key establishment protocol for High Security Applications in Constrained Network environment” / K.R. Chandrasekhara, M.P. Pillai1 and Sebastian, 2010. [Электронный ресурс]: http://www.arxiv.org/pdf/1202.1895

28. Griffiths, I. Programming Java: Building Windows 8, Web, and Desktop Applications for the .NET 4.5 Framework / I. Griffiths, 2012. – [Электронный ресурс]: https://giants.ict.griffith.edu.au./JPL/

29. Herlihy, M. The Art of Multiprocessor Programming. / M. Herlihy, N. Shavit. — Revised 1st Edition. — Morgan Kaufmann, 2012. — 537 p.

30. Ingalls, J. M. Interior Ballistics / J.M. Ingalls. — Nabu Press. — 2014. — 258 p.

31. Michaelis, K. Randomly failed! The state of randomness in current Java implementations / K. Michaelis, C. Meyer, and J. Schwenk, In E. Dawson, editor // CT-RSA. — volume 7779 of LNCS. – 2013. – pp. 129 —144.

32. Georgiev, M. The most dangerous code in the world: Validating SSL certificates in non-browser software / M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov, In T. Yu, G. Danezis, and V. D. Gligor, editors // ACM Conference on Computer and Communications Security. — 2012. — pp. 38 – 49.

33. Pelland, P. Moving to Microsoft Visual Studio 2010 / P. Poland, K. Haines – Microsoft Press, 2011. — 336 p.

34. Nils, G. “Comparing Elliptic Curve Cryptography and RSA on 8-bit CPus”/ Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle // 6th International Workshop Cambridge, MA, USA. — 2004. — 119 p.

Приложение А

Реализация алгоритма

```
package diploms;

import java.lang.Math;
import java.util.function.Function;

public class Main {
	public static int Cx = 9;
	public static int Cy = 4;
	public static int A1_x = 0;
	public static int A1_y = 0;
	public static int A2_x = 0;
	public static int A2_y = 0;
	public static int B1_x = 0;
	public static int B1_y = 0;
	public static int B2_x = 0;
	public static int B2_y = 0;
	public static int[] x1 = new int[100];
	public static int[] y1 = new int[100];
	public static int kol = 0;
	public static int Ab_x;
	public static int Ab_y;
	public static int Ba_x;
	public static int Ba_y;
	public static int modul =37;


	public static void main(String[] args) {

	//(y^2 = x^3+2x+9)mod37;



		spisok_tochek();
		User_A  Alice = new User_A();
		Alice.createA1_A2();
		User_B  Bob = new User_B();
		Bob.createB1_B2();
		Alice.createAb();
		Bob.createBa();
		Bob.table_sibmol();
		Stroka_encoding[] stroka_kodirovka;
		stroka_kodirovka = Bob.encodeString();



		for(int i = 0; i < stroka_kodirovka.length;i++)
		{

			System.out.print( stroka_kodirovka[i].E1);
			System.out.print(stroka_kodirovka[i].E2);
		}
		System.out.println("");
		Alice.decode(stroka_kodirovka);
		/*
```

```
*/



}

private static void spisok_tochek()
{
	int lev_y;
	int prav_x;
	int ostatok = 37;
	int kol = 0;
	for(int y = 1 ; y < 37 ;y++)
	{
		lev_y =(int) Math.pow(y,2);
		lev_y = lev_y % 37;
		for(int x = 0 ; x <37 ;x++)
		{
			prav_x = (int) Math.pow(x,3) + 2*x + 9;
			prav_x = prav_x % 37;
			if(lev_y == prav_x)
			{
				x1[kol] = x;
				y1[kol] = y;
				kol++;
			}
		}
	}
	/*
	for(int x = 0 ; x < kol ;x++)
	{
	System.out.print("x=" + x1[x] + " ");
	System.out.println("y=" + y1[x]);
	}
	*/
}

public static int[] slozenie_tochek (int A_x, int A_y, int B_x , int B_y )
{
	int[] koordinat = new int[2];


	int del = 1;
	int chet = 0;
	int delitel = 0;
	int sr=0;
	int s =0;


	if((A_x-B_x) != 0)
	{	delitel = A_x - B_x;
	if (delitel < 0) delitel= delitel +37;
	}

	else delitel = A_y*2;

		//(8*x)%37 =1
	while (del%delitel != 0)
```

```
		{
			chet++;
			del =(37*chet)+1;
		}


		if((A_x-B_x) != 0)
		{
			sr = (A_y-B_y);
			if (sr < 0) sr= sr +37;
			s = ( (sr)*(del/delitel)) %37;
			if(koordinat[0] < 0) s = s + 37;


		}else
		{
			 sr = (int) (3* Math.pow(A_x, 2)) + 2;
			 s = ( (sr)*(del/delitel)) %37;
		}
		koordinat[0] = ((int) (Math.pow(s, 2)) - (A_x +B_x))%37;
		if(koordinat[0] < 0) koordinat[0] = (koordinat[0] %37) + 37;
		koordinat[1]= (s*(A_x - koordinat[0]) - A_y)%37;
		if(koordinat[1] < 0) koordinat[1] = (koordinat[1] %37) + 37;

		//	System.out.print("x=" + koordinat[0] + " ");
		//	System.out.print("y=" + koordinat[1] + " ");

		return koordinat;


	}

}


package diploms;

import java.sql.Struct;
import java.util.Random;
import java.lang.Math;
import java.util.function.Function;

import javax.sound.sampled.AudioFormat.Encoding;

public class User_A {

	int a;   // a < 37
	int A_x;
	int A_y;
	String table = "abcdefghijklmnopqrstuvwxyz _123456789ABCEFGHIJKLMNOPQRSTUVWXYZ";
	String table2 = new String("abcdefghijklmnopqrstuvwxyz
_m2345t789ABCDEFGHIJKLMNOPQRSTUVWXYZ");
	char[] sumbol = table2.toCharArray();
	String stroka ="attack";
	char[] sumbol_stroka = stroka.toCharArray();



	User_A()
	{//secret keys
		Random a_Random = new Random();
```

```
		Random A_Random = new Random();

		/*
		int A_rdm =A_Random.nextInt(Main.kol);
		this.a = a_Random.nextInt(37);
		this.A_x = Main.x1[A_rdm];
		this.A_y =  Main.y1[A_rdm];
		*/

		this.a = 5;
		this.A_x = 10;
		this.A_y = 20;

	}


	public void createA1_A2()
	{
		int[] A1_rezult = new int[2];

		int[] rezult_A1 = new int[2];
		int[] rezult_A2 = new int[2];
		A1_rezult = Main.slozenie_tochek(Main.Cx, Main.Cy, A_x, A_y);
		rezult_A1 = A1_rezult;
		rezult_A2[0] =A_x;
		rezult_A2[1] =A_y;
		int chet = a - 1;
		//Находим A1
		while (chet > 0)
		{
			chet --;
		rezult_A1 = Main.slozenie_tochek(rezult_A1[0], rezult_A1[1], A1_rezult[0],
A1_rezult[1]);
		rezult_A2 = Main.slozenie_tochek(rezult_A2[0], rezult_A2[1], A_x, A_y);
		}

		Main.A1_x = rezult_A1[0];
		Main.A1_y = rezult_A1[1];

		Main.A2_x = rezult_A2[0];
		Main.A2_y = rezult_A2[1];




	}

	public void createAb()
	{
		//Находим Ab
		int[] Ab_rezult = new int[2];
		Ab_rezult[0] = Main.B2_x;
		Ab_rezult[1] = Main.B2_y;
		int chet = a - 1;
		while (chet > 0)
			{
				chet --;
				Ab_rezult = Main.slozenie_tochek(Ab_rezult[0], Ab_rezult[1],
Main.B2_x, Main.B2_y);
			}
		Main.Ab_x = Ab_rezult[0];
```

```
		Main.Ab_y = Ab_rezult[1];
	}

	public String proverka(String s)
	{
		if (s == "6")
		{
			return "t";
		}

		return  s;
	}


	public void decode(Stroka_encoding [] stroka_kodirovka)
	{
		for (int k =0 ; k < stroka_kodirovka.length; k++)
		{



		   int[] E1_rezult = new int[2];
		   E1_rezult[0] = stroka_kodirovka[k].E1_x;
		   E1_rezult[1] = stroka_kodirovka[k].E1_y;
		      int chet = a-1;
		      while (chet > 0)
				{
					chet --;
					E1_rezult =
Main.slozenie_tochek(E1_rezult[0],E1_rezult[1],stroka_kodirovka[k].E1_x,
stroka_kodirovka[k].E1_y);
				}



		    int[] B1_rezult = new int[2];
		    B1_rezult[0] = Main.B1_x;
		    B1_rezult[1] = Main.B1_y;
			int chet2 = a-1;
			while (chet2 > 0)
				{
					chet2 --;
					B1_rezult =
Main.slozenie_tochek(B1_rezult[0],B1_rezult[1],Main.B1_x, Main.B1_y);
				}


			E1_rezult =
Main.slozenie_tochek(E1_rezult[0],E1_rezult[1],B1_rezult[0],B1_rezult[1]);

			E1_rezult =
Main.slozenie_tochek(E1_rezult[0],E1_rezult[1],Main.Ba_x,Main.Ba_y);

			int[] Ba_rezult = new int[2];
			 Ba_rezult[0] = E1_rezult[0];
			 Ba_rezult[1] = E1_rezult[1];
			int chet3 =51;
				while (chet3 > 0)
					{
						chet3 --;
```

```
                                                Ba_rezult =
Main.slozenie_tochek(Ba_rezult[0],Ba_rezult[1],E1_rezult[0],E1_rezult[1]);
                                        }
                                    E1_rezult =
Main.slozenie_tochek(stroka_kodirovka[k].E2_x,stroka_kodirovka[k].E2_y,Ba_rezult[0],Ba_
rezult[1]);



                                    for(int r = 0; r < Main.x1.length;r++)
                                    {
                        if(E1_rezult[0] == Main.x1[r] && E1_rezult[1] == Main.y1[r])
                                        {
                                        String vivod = "" +sumbol[r];
                                        vivod = proverka(vivod);
                                        System.out.print(vivod);
                                        }
                                    }

                }
        }



}



package diploms;

import java.util.Random;

public class User_B {
        int b;  // b< 37
        int B_x;
        int B_y;
        String table = "abcdefghijklmnopqrstuvwxyz _123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";

        String table2 = new String("abcdefghijklmnopqrstuvwxyz
_123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ");
        char[] sumbol = table2.toCharArray();
        String stroka ="i lied at home";
        char[] sumbol_stroka = stroka.toCharArray();



        User_B()
        {//secret keys
                Random b_Random = new Random();
                Random B_Random = new Random();



                this.b = 7;
                this.B_x = Main.x1[6];
                this.B_y =  Main.y1[6];



        }
```

```
	public void createB1_B2()
	{
		int[] B1_rezult = new int[2];
		int[] rezult_B1 = new int[2];
		int[] rezult_B2 = new int[2];
		B1_rezult = Main.slozenie_tochek(Main.Cx, Main.Cy, B_x, B_y);
		rezult_B1 = B1_rezult;
		rezult_B2[0] =B_x;
		rezult_B2[1] =B_y;
		int chet = b - 1;
		int t =1;
		//Находим B1
		while (chet > 0)
		{
			t++;
			chet --;
		rezult_B1 = Main.slozenie_tochek(rezult_B1[0], rezult_B1[1], B1_rezult[0],
B1_rezult[1]);
		rezult_B2 = Main.slozenie_tochek(rezult_B2[0], rezult_B2[1], B_x, B_y);
		}

		Main.B1_x = rezult_B1[0];
		Main.B1_y = rezult_B1[1];

		Main.B2_x = rezult_B2[0];
		Main.B2_y = rezult_B2[1];


	}
	public void createBa()
	{
		//Находим Ab
		int[] Ba_rezult = new int[2];
		Ba_rezult[0] = Main.A2_x;
		Ba_rezult[1] = Main.A2_y;
		int chet = b - 1;
		while (chet > 0)
			{
				chet --;
Ba_rezult = Main.slozenie_tochek(Ba_rezult[0], Ba_rezult[1], Main.A2_x, Main.A2_y);
			}

		Main.Ba_x = Ba_rezult[0];
		Main.Ba_y = Ba_rezult[1];

	}


	public void table_sibmol()
	{



		for(int i=0; i<+sumbol_stroka.length; i++)
		{
		//   System.out.println(sumbol[i]);
		}
		//System.out.println(sumbol.length);
```

```
	}

	public Stroka_encoding[] encodeString()
	{
		System.out.println( stroka );
		Stroka_encoding[] simbol_strok =new Stroka_encoding[sumbol_stroka.length]
;

		for(int i=0; i < sumbol_stroka.length; i++)
		{
			for(int j=0; j < sumbol.length; j++)
			{

			   if(sumbol_stroka[i] == sumbol[j])
			   {
				int q = 23;
				int[] E1_rezult = new int[2];
				    E1_rezult[0] = Main.Cx;
				    E1_rezult[1] = Main.Cy;
				    int chet = q - 1;
				    while (chet > 0)
					{
						chet --;
E1_rezult = Main.slozenie_tochek(E1_rezult[0], E1_rezult[1], Main.Cx, Main.Cy);
					}



				   //E2
				    int[] E2_rezult = new int[2];
				    E2_rezult[0] = Main.A1_x;
				    E2_rezult[1] = Main.A1_y;
				    chet = (q+b)-2; //+++
				    while (chet > 0)
					{
						chet --;
						E2_rezult =
Main.slozenie_tochek(E2_rezult[0], E2_rezult[1], Main.A1_x, Main.A1_y);
					}
				    E2_rezult =
Main.slozenie_tochek(Main.x1[j],Main.y1[j], E2_rezult[0], E2_rezult[1]);
				    int[] q1 = new int[2];
				    q1[0] = Main.A2_x;
				    q1[1] = Main.A2_y;
				    chet =(37-q)-1;
				    while (chet > 0)
					{
						chet --;
			q1 = Main.slozenie_tochek(q1[0], q1[1], Main.A2_x, Main.A2_y);
					}
				    E2_rezult =
Main.slozenie_tochek(E2_rezult[0],E2_rezult[1], q1[0],q1[1]);
				    E2_rezult =
Main.slozenie_tochek(E2_rezult[0],E2_rezult[1], Main.Ab_x,Main.Ab_y);
	//		System.out.print("E1_x=" + E1_rezult[0] + " ");


				    String c ="";
				  String d="";
				 for(int x = 0 ; x < Main.x1.length ;x++)
				  {
```

```
                    if(E1_rezult[0] == Main.x1[x] && E1_rezult[1] == Main.y1[x])
                                {
                                      c = ""+sumbol[x];
                                      if(c != "" && d != "")
                                            {

                                                  simbol_strok[i] = new
Stroka_encoding (E1_rezult[0],E1_rezult[1],E2_rezult[0],E2_rezult[1],c,d);
                                                    c ="";
                                              d="";
                                            }
                                      }
                                      if(E2_rezult[0] == Main.x1[x] && E2_rezult[1] ==
Main.y1[x])
                                      {
                                            d = ""+sumbol[x];
                                            if(c != "" && d != "")
                                            {

                                                  simbol_strok[i] = new
Stroka_encoding (E1_rezult[0],E1_rezult[1],E2_rezult[0],E2_rezult[1],c,d);
                                                    c ="";
                                              d="";
                                            }
                                      }

                                }
                           }
                    }

             }



      return simbol_strok;
      }



}
```

```
package diploms;

public class Stroka_encoding {

      public int E1_x;
      public int E1_y;
      public int E2_x;
      public int E2_y;
      public String E1;
      public String E2;
```

```
	public Stroka_encoding(int x , int y, int x1, int y1,String e1,String e2)
	{
		this.E1_x = x;
		this.E1_y = y;
		this.E2_x = x1;
		this.E2_y = y1;
		this.E1 =e1;
		this.E2=e2;
}
```