Сохранение последовательности шагов, выполнявшихся студентом при решении, реализует класс `Solver`. Этот класс содержит список, состоящий из названия шага и данных для его выполнения, заполненных студентом.

Для демонстрации работоспособности разработанной архитектуры программного тренажера было создано web-приложение для решения СЛАУ. При загрузке страницы с выбранным методом (на рисунке 1 метод Гаусса решения СЛАУ) вызывается генератор СЛАУ. Студенту в виде списка предоставляются доступные шаги. При выборе конкретного шага на странице отобразится набор полей для заполнения данными, необходимыми для его выполнения.

При нажатии на кнопку «Выполнить» заполненные данные отправляются на сервер, где производятся вычисления, и СЛАУ обновляется.

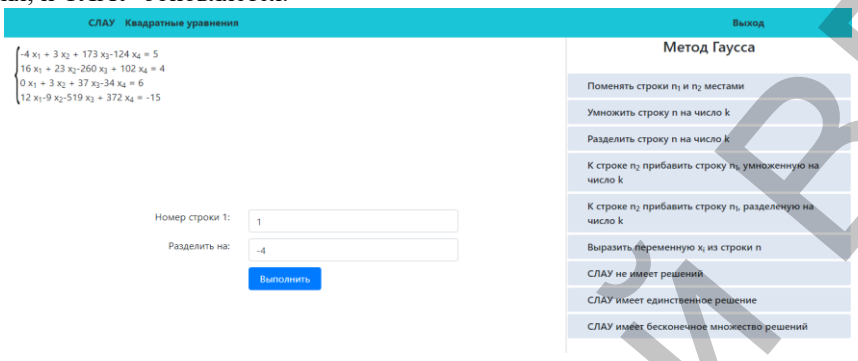

Рисунок 1 – Выполнение шага для решения задачи в web-приложении

Заключение. Такие мощные инструменты объектно-ориентированного языка Java как обобщения, интроспекция, рефлексия, позволяют динамически модифицировать программу во время выполнения, что предоставляет новые возможности при написании кода.

В результате была спроектирована архитектура программного обеспечения тренажера, которая позволяет гибко расширять приложения, добавляя новые задачи по различным дисциплинам учебных планов, проанализировать выполненные студентом действия и оценить уровень его знаний, умений и навыков.

1. Гослинг, Дж., Джой, Б., Стил, Г. Л., Брача, Г., Бакли, А. Язык программирования Java SE 8. Подробное описание. – Москва: Вильямс, 2015. – 672 с.
2. Коновалов, Я.Ю. Генератор контрольных заданий по высшей математике: опыт создания и применения / Я.Ю. Коновалов, С.К. Соболев // ФГБОУ ВПО «МГТУ им. Н.Э.Баумана». – 2015. – №4. – С. 1046-1055.

ИСПОЛЬЗОВАНИЕ ТЕОРИИ ГРУПП ТОЧЕК НА ЭЛЛИПТИЧЕСКОЙ КРИВОЙ ДЛЯ СОЗДАНИЯ ЭЛЕКТРОННОЙ ПОДПИСИ

Марчук К.С.,
студент 2 курса БГТУ, г. Минск, Республика Беларусь
Научный руководитель – Асмыкович И.К., канд. физ.-мат. наук, доцент

Долгое время люди копили информацию. Сначала это были рисунки на стенах, после – печатный или письменный текст, а в наше время мы имеем огромное количество информации в электронном виде. Ни для кого не секрет, что большая часть информации имеет ценность, особенно для юридических лиц. В нашем примере, информация нуждается в проверке принадлежности – подписи.

ECDSA (Elliptic Curve Digital Signature Algorithm) – криптографический алгоритм с открытым ключом для создания цифровой подписи, определённый в группе точек эллиптической кривой [1]. Подпись создается секретно, но может быть публично проверена. Это означает, что только один субъект может создать подпись сообщения, но любой может проверить её корректность.

До того, как ECC стала популярной, почти все алгоритмы с открытым ключом основывались на RSA, DSA и DH [2], альтернативных криптосистемах на основе модулярной арифметики. Они по-прежнему популярны, и часто используются вместе с ECC. Однако основы ECC всё ещё являются для большинства людей загадкой.

Цель – рассмотреть конкретный пример работы алгоритма подписи данных, основанного на теории эллиптических кривых.

Материал и методы. В Биткойне [3] используется вариант эллиптической криптографии secp256k1. Покажем, откуда берутся секретные и открытые ключи и как они связаны друг с другом. В ECDSA секретный ключ – это случайное целое число между 1 и значением порядка – количеством

элементов группы. Открытый же ключ получается из секретного при помощи операции скалярного умножения базовой точки на значение секретного ключа. Это показывает, что максимально возможное количество секретных ключей - конечно, и равно порядку - действительно большому числу.

Результаты и их обсуждение. Вычисление открытого ключа разбивается на ряд операций удвоения и сложения точек, начиная с базовой точки [1]. Напомню, сложение точек p + q определяется покомпонентно следующим образом [1]:

$c = (q_y - p_y) / (q_x - p_x)$, $r_x = c^2 - p_x - q_x$, $r_y = c\,(p_x - r_x) - p_y$

А операция «удвоения» точки p выглядит следующим образом:

$c = (3p_x^2 + a) / 2p_y, r_x = c^2 - 2p_x$, $r_y = c\,(p_x - r_x) - p_y$

Объем вычислений на реальном примере был бы невероятно сложным, но мы можем попробовать на примере с небольшими числами, чтобы увидеть, как это работает. Итак, возьмем уравнение кривой Биткойна, но будем использовать маленькие числа.

Уравнение кривой: $y^2 = x^3 + 7$

Модуль: 67 Базовая точка: (16, 4) Порядок: 79

Требуется выбрать секретный ключ из 79 доступных, возьмём 3. Чтобы найти публичный ключ, потребуется операция удвоения и сложения:

$c = (3 * 16^2 + 0) / (2 * 4) \bmod 67 \rightarrow c = 768 / 8 \bmod 67 \rightarrow 31 / 8 \bmod 67$

Но как нам выполнить операцию деления в контексте конечного поля, где результат должен быть целочисленным? Для этого мы должны умножить 31 на величину, обратную к 8. Это должно быть такое числа, при умножении на которое, мы получаем остаток 1 по заданному модулю, то есть: $8^{-1} = 42$

Далее все довольно просто:

$c = 31 * 42 \bmod 67 = 29$ $r_x = (29^2 – 2 * 16) \bmod 67 = 5$ $r_y = (29 * (16 – 5) – 4) \bmod 67 = 47$

Далее сложим эту точку с базовой p = (16, 4), q = (5, 47):

$c = (47 – 4)/(5 – 16) \bmod 67 = 57$ $r_x = 3249 – 16 – 5 \bmod 67 = 12$ $r_y = 57 * (16 – 12) – 4 \bmod 67 = 23$

Итак, мы определили, что для секретного ключа 3 публичным ключом будет **точка** (12, 23).

Подпись данных секретным ключом. Теперь у нас есть секретный и публичный ключ, которые мы можем использовать для подписи данных. Сами данные могут иметь любую длину. Для начала данные хэшируются, чтобы получить уникальное число, содержащее такое же количество битов (256), как и порядок кривой. Упрощая нашу задачу, мы пропустим шаг хеширования и подпишем данные z. Обозначим через G базовую точку, через n - порядок, а d - закрытый ключ. Алгоритм создания подписи выглядит следующим образом:

1. Выбрать некоторое целое k в пределах от 1 до n-1.
2. Рассчитать точку (x, y) = k * G, используя скалярное умножение.
3. Найти r = x mod n. Если r = 0, вернуться к шагу 1.
4. Найти s = (z + r * D) / k mod n. Если S = 0, вернуться к шагу 1.
5. Пара (r, s) является нашей подписью

Напомним, вместо деления числителя на знаменатель, мы числитель умножаем на обратную знаменателю величину. На шаге 1 важно, чтобы k не повторялось в разных подписях, и чтобы его не могла угадать третья сторона[2]. То есть k должен быть либо случайным, либо создан детерминированным процессом, который хранится в тайне. Иначе третья сторона получает возможность найти секретный ключ, начиная с шага 4, так как s, z, r, k и n всем известны.

Давайте выберем в качестве наших данных число 21 и подпишем его секретным ключом 2.

z = 17 (данные), n = 79 (порядок), G = (16, 4) (базовая точка), d = 3 (секретный ключ)

1. Выберем случайное число:

$k = \text{rand}(1, n - 1) \rightarrow$ $k = \text{rand}(1, 79 - 1) \rightarrow k = 2$

2. Рассчитаем точку.

Это делается таким же образом, как ранее при вычислении публичного ключа - для краткости опустим подробную арифметику сложения и удвоения.

$(x, y) = 2G = (16, 4) + (16, 4) = (5, 47) \rightarrow x = 5, y = 47$

3. Находим *r:*

$r = x \bmod n \rightarrow r = 5 \bmod 79 = 5$

4. Находим *s*:

$s = (z + r * d) / k \bmod n \rightarrow s = (21 + 5 * 3) / 2 \bmod 79 = 18 \bmod 79 = 18$

5. Теперь наша подпись – это пара (*r, s*) = (5, 18). Как и секретные и публичные ключи, подпись обычно представляется в виде шестнадцатеричной строки.

Заключение. В тезисах показана работа алгоритма ECDSA на примере малых порядков. Отметим, что надёжность алгоритмов, основанных на теории эллиптических кривых, гораздо выше, при таком же размера ключа, как и в алгоритме RSA.

1. Острик, В.В. Алгебраическая геометрия и теория чисел: рациональные и эллиптические кривые / В.В. Острик, М.А. Цфасман - М.: МЦНМО, 2001. – 48 с.
2. Википедия, свободная энциклопедия (https://www.wikipedia.org)
3. Биткойн (https://ru.wikipedia.org/wiki/Биткойн)