УДК 378.016:[519.6:004.43:003.26]

Рожков А. В., Руссу Р. Б.

ЭКСПЕРИМЕНТАЛЬНАЯ МАТЕМАТИКА И ЯЗЫК JULIA — ВЫЧИСЛЕНИЯ В КРИПТОГРАФИИ*

Александр Викторович Рожков

доктор физико-математических наук, профессор

great.ros.marine2@gmail.com

ФГБОУ ВО «Кубанский государственный университет», Россия, Краснодар

Руслан Борисович Руссу

магистрант факультета математики и компьютерных наук

rus.russu2013@yandex.ru

ФГБОУ ВО «Кубанский государственный университет», Россия, Краснодар

EXPERIMENTAL MATHEMATICS AND LANGUAGE JULIA – COMPUTING IN CRYPTOGRAPHY

Alexander Viktorovich Rozhkov

Kuban State University, Russia, Krasnodar

Ruslan Borisovich Russu

Kuban State University, Russia, Krasnodar

Аннотация. *Научно-методическая инициатива по обучению математике и информатике, реализуемая в КубГУ с 2015 г. Поддержана Благотворительным фондом Владимира Потанина. В данной статье исследуются криптографические алгоритмы.*

Abstract. *The scientific and methodical initiative of training in mathematics and informatics realized in KUBSU since 2015. Supported by the Vladimir Potanin Charitable Foundation. This article examines cryptographic algorithms.*

Ключевые слова: *Поля Галуа, примитивный элемент, язык программирования Julia, эллиптическая кривая, криптография.*

Keywords: *Galois fields, primitive element, Julia programming language, elliptic curve, cryptography.*

В рамках реализации проекта, поддержанного грантом фонда Владимира Потанина ГСГК-0072-21, разрабатывается ряд курсов для магистерской программы “Алгебраические методы защиты информации”, открытой в Кубанском государственном университет в 2013 г. Первые результаты изложены в [1]. В данной работе речь идет о курсе “Теоретико-числовые методы криптографии”. Основой курса является теория полей Галуа и эллиптические кривые [2].

Введение

Julia (текущая версия 1.7.1) — высокопроизводительный язык программирования с динамической типизацией, созданный для математических вычислений. Синтаксис языка схож с MATLAB и Python. Julia написан на Си, C++ и Scheme. В стандартный комплект входит JIT-компилятор LLVM, благодаря чему он не уступают в производительности компилируемому языку C/C++.

Язык имеет встроенную поддержку распределенных и параллельных вычислений. Более того, в код Julia можно включать модули и библиотеки, написанные на языках C/C++, FORTRAN, Python, Java.

Язык Julia имеет более 7 тыс. расширяющих пакетов и поэтому может использоваться как система компьютерной алгебры.

В области алгебры и криптографии базовые пакеты Nemo v0.28.0, AbstractAlgebra v0.23.0, GaloisFields v1.1.1, Primes v0.4.0, LinearAlgebra, Hecke, SymPy v1.1.3. Для построения графиков хорош мощный пакет Plots v1.25.7.

В Windows Julia по умолчанию ставится по адресу C:\Users\user\AppData\Local\Programs\Julia-1.7.1, а расширяющие ее пакеты по адресу C:\Users\user\.julia. При этом, даже если вы не установили ни одного пакета большинство пакетов уже будет на вашем компьютере — их объем больше 7 Gb. Причина в том, что многие пакеты между собой связаны перекрёстными ссылками.

Работа с пакетами. Запускаем Julia в терминале REPL. Нажимаем кнопку “]” и попадаем в менеджер пакетов `(@v1.7) pkg>`

Для добавления, удаления, тестирования пакета, обновления и выяснения статуса пакетов выполняем следующие команды

```
(@v1.7) pkg> add Nemo
(@v1.7) pkg> rm Nemo
(@v1.7) pkg> test Nemo
(@v1.7) pkg> up
(@v1.7) pkg> st
Status `C:\Users\rosav\.julia\environments\v1.7\Project.toml`
 [eb74ef6d] DarkCurves v0.2.0
 [8d0d7f98] GaloisFields v1.1.1
 [7073ff75] IJulia v1.23.2
 [2edaba10] Nemo v0.28.0
```

Менеджер помощи вызывается клавишей “?”

```
help?>
```

Чтобы работать в привычной среде браузера нужно набрать команды

```
julia> using IJulia
julia> notebook()
```

Вычисления в полях Галуа

Запускаем Julia в терминале REPL, можно это сделать и различных редакторах типа VS Code, и подключаем пакет Nemo

```
julia> using Nemo
Welcome to Nemo version 0.28.0
Nemo comes with absolutely no warranty whatsoever
```

У пакета Nemo есть одна методическая особенность. Полями Галуа в нем называются простые поля Галуа GF(p), которые реализуются как кольца вычетов по простому модулю.

```
julia> F=GF(7)
Galois field with characteristic 7
julia> F(3)^3
6
julia> F(3^3)
6
```

Произвольные поля Галуа называются конечными полями.

Конечное поле можно задать двумя способами.

Первый способ. Поле Галуа задается как поле разложения неизвестного нам многочлена. Мы задаем только характеристику поля и степень расширения простого поля:

```
julia> R, x = FiniteField(5, 2, "x")
(Finite field of degree 2 over F_5, x)
```

Но мы можем легко выяснить какой это многочлен

```
julia> x^2
x+3
```

Значит это многочлен $x^2 = x+3 \Rightarrow x^2 - x - 3 = x^2 + 4x + 2$.

Второй способ. Вначале создадим кольцо T многочленов над полем вычетов, а потом зададим расширение простого поля Галуа, как поля разложения U нашего конкретного многочлена

```
julia> T, t = PolynomialRing(ResidueRing(ZZ, 5), "t")
(Univariate Polynomial Ring in t over Integers modulo 5, t)
julia> U, z = FiniteField(t^2+t + 1, "z")
(Finite field of degree 2 over F_5, z)
```

Модельные задачи

Задача №1. *Для многочлена* $x^{17} - x + 3$ *над полем GF(199) найти корни.*

Решим задачу, не используя конечные поля.

```
julia> function f(p)
 for i in 1:p
 if (i^17-i+3)%p == 0
 print(i,",")
 end
 end
 end
f (generic function with 1 method)
julia> f(199)
25,199,
```

Получилось, что 199, т.е. 0 в поле GF(199), является корнем, что неверно. В чем причина? В разрядности целых чисел

```
julia> 199^17 - это Int64 19-значные числа
5245870108793248583
julia> BigInt(199)^BigInt(17) - допустимы миллионы знаков
1203655761401433389534544312357434563399
```

Немного изменим нашу программку

```
julia> function f(p)
 for i in 1:p
 if (BigInt(i)^BigInt(17)-i+3)%p == 0
```

```
 print(i,",")
 end
 end
 end
f (generic function with 1 method)
julia> f(199)
28,149,
```

Корни получились другие.

Теперь подключим поля Галуа

```
using Nemo
julia> function f(p)
 for i in GF(p)
 if i^17-i+3 == 0
 print(i,",")
 end
 end
 end
f (generic function with 1 method)
julia> f(199)
28,149,
```

Теперь корни совпали.

Задача №2. Найти примитивный элемент поля $GF(5^2)$, заданного как поле разложения многочлена $f(\mathrm{x}) = x^2 + x + 1, GF(5)$.

Решение. Используем второй способ задания поля Галуа. Вначале создадим кольцо T многочленов над поле вычетов, а потом зададим расширение простого поля Галуа, как поля разложения U нашего многочлена

```
julia> T, t = PolynomialRing(ResidueRing(ZZ, 5), "t")
(Univariate Polynomial Ring in t over Integers modulo 5, t)
julia> U, z = FiniteField(t^2+t + 1, "z")
(Finite field of degree 2 over F_5, z)
```

Проверим является ли элемент z примитивным. К сожалению, нет:

```
julia> z^8
4*z+4
julia> z^12
1
```

Проверим элемент y = z+1. Тоже не подходит:

```
julia> y=z+1
z+1
julia> y^8
z
julia> y^12
```

```
1
```

Проверим y = z+2. “Упорство и труд – все перетрут!” — подходит

```
julia> y=z+2
z+2
julia> y^8
4*z+4
julia> y^12
4
```

Итак, y = z+2 — примитивный элемент нашего поля.

Задачи 3. *Составить таблицу степеней примитивного элемента и таблицу логарифма Якоби.*

Решение. Вычислим все 24 степени элемента y:

```
julia> y=z+2
z+2
julia> for i in 1:24
 print(y^i, ",")
 end
z+2,3*z+3,z+3,4*z,4*z+1,3,3*z+1,4*z+4,3*z+4,2*z,2*z+3,4,4*z+3,2*z+2,4*z+2,z,
z+4,2,2*z+4,z+1,2*z+1,3*z,3*z+2,1
```

Составим объединенную таблицу — первая строка — показатели степеней примитивного элемента y, вторая строка — значение его степеней, третья -логарифм Якоби, который строится просто просмотром первых двух строк. Например, как найти L(10), по определению логарифма $1+y^{10}=y^{L(10)}$. Находим $y^{10}=2z \Rightarrow y^{10}+1=2z+1=y^{21}$, таким образом L(10)=21. Отметим, т.к. 0 не является степенью примитивного элемента, то в 12-столбце вместо значения логарифма стоит символ запрета или останова в машинах Тьюринга — знак #.

В итоге получаем:

Таблица 1 — Объединенная таблица степеней примитивного элемента и логарифма Якоби

i	**y^i**	**L(i)**
1	z+2	3
2	3z+3	9
3	z+3	17
4	4z	5
5	4z+1	15

6	3	12
7	3z+1	23
8	4z+4	4
9	3z+4	22
10	2z	21
11	2z+3	19
12	4	#
13	4z+3	8
14	2z+2	11
15	4z+2	13
16	z	20
17	Z+4	16
18	2	6
19	2z+4	10
20	Z+1	1
21	2z+1	14
22	3z	7
23	3z+2	2
24	1	18

Применение языка Julia при изучении разделов алгебры, требующих больших вычислений естественно и продуктивно.

Эллиптическая кривая

Задача №4. Вычислить количество точек на кривой L, заданной уравнением $y^2 = x^3 + 3x + 8$ над полем GF(199).

Теорема. (Хассе) *Если эллиптическая кривая L задана над полем, содержащим q элементов, то число точек на ней удовлетворяет неравенству*

$$|q+1-\#L| \le 2\sqrt{q}.$$

В нашем случае у кривой точек будет от 172 до 228.

```
julia> using Nemo
Welcome to Nemo version 0.28.0
Nemo comes with absolutely no warranty whatsoever
julia> function ros(p)
 F=GF(p)
 t=0
 for i in F
 for j in F
  s= j^2
```

```
    s1= i^3+3*i+8
    if s == s1
    t = t+1
    print("(",i, ",",j, "),")
    end
   end
   end
   print(t)
   end
  ros(199)
```

(0,±40),(6,±21),(10,±21),(11,±24),(12,±58),(14,±40),(15,±29),(16,±42),(17,±14),(18,±83),(19,±77),(21,±24),(22,±37),(29,±87),(33,±5),(35,±2),(36,±87),(38,±95),(40,±99),(45,±46),(47,±33),(48,±10),(51,±26),(52,±26),(54,±15),(55,±69),(57,±74),(59,±14),(64,±96),(65,±3),(68,±5),(69,±37),(71,±23),(73,0),(74,±84),(75,±32),(79,±19),(82,±81),(88,±13),(89,±28),(91,±55),(92,±22),(94,±7),(95,±66),(96,±26),(98,±5),(99,±9),(102,±97),(106,±39),(108,±37),(110,±25),(111,±65),(114,±90),(115,±79),(118,±98),(119,±81),(120,±75),(121,±6),(123,±14),(125,±70),(126,±4),(130,±55),(132,±23),(134,±87),(136,±78),(138,±36),(141,±90),(142,±50),(143,±90),(144,±35),(146,±53),(149,±32),(150,±47),(151,±48),(152,±11),(154,±41),(155,±62),(159,±31),(161,±44),(162,±63),(163,±3),(165,±2),(167,±24),(170,±3),(171,±85),(172,±12),(173,±57),(174,±32),(177,±55),(180,±16),(181,±93),(183,±21),(185,±40),(186,±89),(187,±45),(189,±42),(193,±42),(195,±23),(197,±81),(198,±2), 199

Получилось 199 решений и плюс бесконечно удаленная точка — всего 200 точек. ***Очень редкая кривая, у нее ровно q+1 точка***.

Задача №5. *Построить график эллиптической кривой*.

Используем полученные результаты. Обратим внимание, что она совсем не похожа на эллиптическую кривую над полем действительных чисел

```
    using Plots
x=[0,6,10,11,12,14,15,16,17,18,19,21,22,29,33,35,36,38,40,45,47,48,51,52,54,
55,57,59,64,65,68,69,71,73,74,75,79,82,88,89,91,92,94,95,96,98,99,102,106,
108,110,111,114,115,118,119,120,121,123,125,126,130,132,134,136,138,141,142,1
43,144,146,149,150,151,152,154,155,159,161,162,163,165,167,170,171,172,173,
174,177,180,181,183,185,186,187,189,193,195,197,198];
y=[40,21,21,24,58,40,29,42,14,83,77,24,37,87,5,2,87,95,99,46,33,10,26,26,15,6
9,74,14,96,3,5,37,23,0,84,32,19,81,13,28,55,22,7,66,26,5,9,97,39,37,25,65,90,
79,98,81,75,6,14,70,4,55,23,87,78,36,90,50,90,35,53,32,47,48,11,41,62,31,44,6
3,3,2,24,3,85,12,57,32,55,16,93,21,40,89,45,42,42,23,8,2];
 data=[y,-y]
    plot(x,data)
    savefig("ros.png")
```

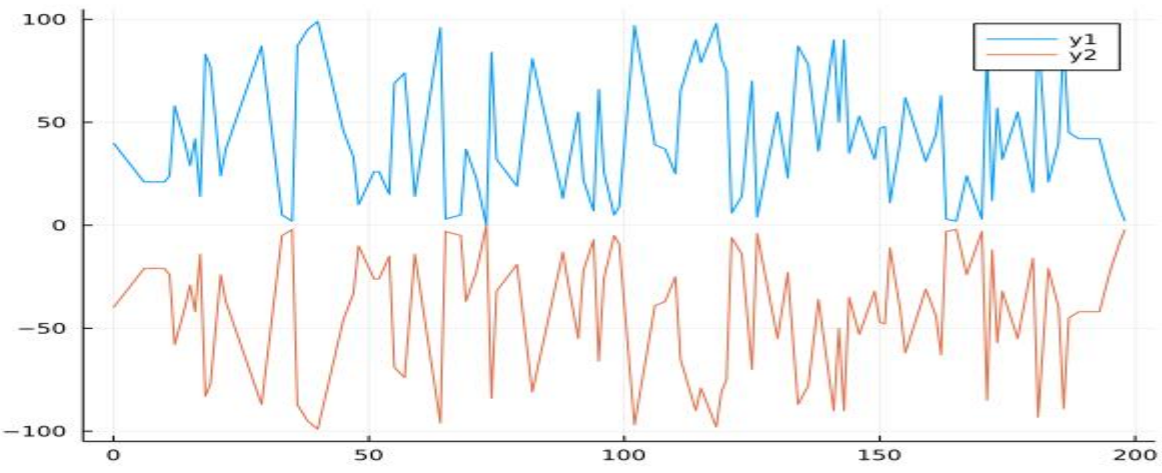

Рисунок 1 — График кривой L над полем GF(199)

Задача №6. *Найти элемент максимального порядка на эллиптической кривой*. Наша кривая $y^2 = x^3 + ax + b$, $P(\mathrm{x}_1, \mathrm{y}_1) + \mathrm{Q}(\mathrm{x}_2, \mathrm{y}_2) = \mathrm{R}(\mathrm{x}_3, \mathrm{y}_3)$ — ее точки. Тогда при P = Q $\begin{cases} x_3 = k^2 - 2x_1 \\ \mathrm{y}_3 = k(\mathrm{x}_1 - \mathrm{x}_3) - y_1 \end{cases}, k = \dfrac{3x_1^2 + a}{2y_1}$,

при $P \neq Q$ $\begin{cases} x_3 = k^2 - (x_1 + x_2) \\ \mathrm{y}_3 = k(\mathrm{x}_1 - \mathrm{x}_3) - y_1 \end{cases}, k = \dfrac{y_2 - y_1}{x_2 - x_1}$.

Поскольку кривая имеет 200 элементов, то порядок максимального элемента может быть 10, 20, 25, 40, 50, 100, 200.

Первая программа — удвоение точки s

```
using Nemo
function rosa(p::Int,s::Vector{gfp_elem})
 F=GF(p)
 a= F(s[1]); b= F(s[2])
 k= (F(3)*a^2+F(3))*(F(2)*b)^(-1)
 a1 = k^2-F(2)*a; b1 = k*(a-a1)-b
 return [a1,b1]
end
```

Вторая — сложение точек s +S

```
function rosA(p::Int,s::Vector{gfp_elem},S::Vector{gfp_elem})
 F=GF(p)
 a= F(s[1]); b= F(s[2]); A= F(S[1]); B= F(S[2])
 if s[1] == S[1]
 return F(0)
 else
 k= (B-b)*(A-a)^(-1)
 A = k^2-(a+A)
 B = k*(a-A)-b;
 return [A,B]
end
end
```

Программа нахождения порядка точки s

```
function rosN(p::Int,s::Vector{gfp_elem})
 S= rosa(p,s)
 for i in 1:p
  S= rosA(p,s,S)
  if S[1] == s[1]
  println("s=",s, "->","N=",i+3)
  break
  end
 end
 end
julia> rosN(199,[GF(199)(102),GF(199)(97)])
s=gfp_elem[102, 97]->N=100
```

Значит точка [102,97] имеет порядок 100.

Пусть M — это множество точек кривой, без нулевой — они перечислены выше.

```
julia> for m in M
 rosN(199,[GF(199)(m[1]),GF(199)(m[2])])
 end
s=gfp_elem[0, 40]->N=100
s=gfp_elem[6, 21]->N=20
s=gfp_elem[10, 21]->N=200
```

Кривая очень хороша, поскольку как группа является циклической. Точка [10,21] имеет порядок 200. Так как функция Эйлера от 200 равна 80, то точек порядка 200 ровно 80 штук.

Обратим внимание на тип переменных в наших программах rosN(p::Int,s::Vector{gfp_elem}). Здесь p::Int — это целые числа, по умолчанию 64 битные, а s::Vector{gfp_elem} — это вектор с координатами из поля Галуа.

То, что координаты из поля Галуа очень важно, потому, что элемент 3 по модулю 7 и элемент 3 поля Галуа GF(7) для языка Julia — это совершенно разные объекты.

Выводы

Никакие из выше проведенных вычислений не могут быть проведены вручную за разумное время. Язык Julia лаконичен и ориентирован на математические вычисления. И может быть применен в любой области математики, как хорошее вспомогательное средство.

* Проект реализуется победителем Конкурса на предоставление грантов преподавателям магистратуры благотворительной программы «Стипендиальная программа Владимира Потанина» Благотворительного фонда Владимира Потанина

Список литературы

1. *Рожков, А. В.* Экспериментальная математика в КубГУ – первые результаты / А. В. Рожков. Текст: непосредственный // Новые информационные технологии в образовании и науке: материалы XIV международной научно-практической конференции конференции, Екатеринбург, 1–5 марта 2021 г. Екатеринбург: Рос. гос. проф.-пед. ун-т, 2021, С. 163–172.

2. *Введение* в теоретико-числовые методы криптографии / М. М. Глухов, И. А. Круглов, А. В. Пичкур, А. В. Черемушкин. Санкт-Петербург: Лань, 2021. URL: https://e.lanbook.com/reader/book/153680. Текст: электронный.