УДК 512.772.7

Н. С. Колесников

ЭФФЕКТИВНАЯ РЕАЛИЗАЦИЯ ЭКСПОНЕНЦИАЛЬНОЙ ЧАСТИ АЛГОРИТМА ПОДСЧЕТА ТОЧЕК В ЯКОБИАНАХ ГИПЕРЭЛЛИПТИЧЕСКИХ КРИВЫХ РОДА 2

Задача вычисления порядка якобиана гиперэллиптической кривой является классической задачей теории чисел с приложениями в современной криптографии. Якобиан кривой используется для построения криптосистем, основанных на дискретном логарифме; как группа большого «неизвестного» порядка в конструкциях верифицируемых функций задержки (VDF) и других приложениях. В статье приводится обзор подходов к ускорению наиболее быстрого алгоритма подсчета точек в якобианах гиперэллиптических кривых – алгоритма Годри – Шоста. Данный алгоритм состоит из двух этапов: 1) нахождение числа точек (характеристического многочлена) по модулю малых простых чисел с объединением результата в один большой модуль по китайской теореме об остатках (полиномиальная часть алгоритма); 2) восстановление полного числа точек из информации по модулю с помощью алгоритмов, основанных на парадоксе дней рождения (экспоненциальная часть алгоритма). В теории алгоритм терминируется в первой части и имеет полиномиальную сложность $O(\log^8 q)$, где q – размер конечного поля. Однако на практике полиномиальная часть алгоритма останавливается на некотором простом числе ℓ из-за ограничений по использованию памяти, которая растет как $O(\ell^4)$, и полное число точек находят уже по экспоненциальному алгоритму. В данной работе выполнена многопоточная реализация экспоненциальной части алгоритма Годри – Шоста на языке программирования C++, дается оценка ее эффективности по затратам времени и памяти.

Computing the order of Jacobian of a hyperelliptic curve is a common number-theoretical problem that has lots of applications in modern cryptography. Namely, Jacobians are applicable to constructions of DLP-based cryptosystems, as well as constructions of verifiable delay functions (VDF's), since they can be viewed as large groups of unknown order. In this article, we present an overview of approaches to accelerate Gaudry-Schost point counting algorithm that is the fastest known algorithm for computing the order of Jacobians of hyperelliptic curves of genus 2. This algorithm consists of two stages: 1) computing the number of points (equivalently, the characteristic polynomial of the curve) modulo some small primes and combining the result into a large module using CRT (polynomial-time part); 2) restoring the number of points utilizing modular data using algorithms based on birthday paradox (exponential-time part). Theoretically, the algorithm terminates after the first stage with $O(\log^8 q)$ time-complexity, where q is a finite field modulus. However, in practice we terminate the polynomial-time part (due to high memory consumption), and we proceed to the second, memory-efficient, exponential-time part. This article presents a multithreaded C++ implementation of exponential part of Gaudry-Schost's point counting algorithm. We evaluate the efficiency of our multithreaded implementation.

Вестник Балтийского федерального университета им. И. Канта.
Сер.: Физико-математические и технические науки. 2020. № 4. С. 42 – 48.

Ключевые слова: гиперэллиптическая кривая, якобиан, подсчет точек, алгоритм Годри — Шоста, кривые рода 2.

Keywords: hyperelliptic curve, Jacobian, point counting, Gaudry — Schost algorithm, genus 2 curves.

Введение

Пусть $C: y^2 = f(x)$ — гиперэллиптическая кривая рода $g = 2$ (то есть $\deg(f) = 2g + 1$), определенная над конечным полем $\mathbb{F}_q$. Введем следующие обозначения:

- Jac_C — якобиан кривой C. Элементами якобиана являются классы эквивалентности дивизоров кривой с представителями — редуцированными дивизорами;
- $D = \big(u(x), v(x)\big) \in Jac_C$ — представление редуцированного дивизора в координатах Мамфорда — Кантора, где $u(x)$ и $v(x)$ — многочлены с коэффициентами из поля $\mathbb{F}_q$. Для рассматриваемых в статье кривых рода 2 редуцированные дивизоры имеют вид $D = (x^2 + u_1 x + u_0,\ v_1 x + v_0)$. Таким образом, элемент якобиана может быть представлен четверкой $(u_0, u_1, v_0, v_1) \in \mathbb{F}_q^4$.
- $\chi(T) = T^4 - s_1 T^3 + s_2 T^2 - s_1 q T + q^2$ — характеристический многочлен эндоморфизма Фробениуса, индуцируемого в якобиане отображением $(x, y) \mapsto (x^q, y^q)$ на кривой. Имеем $\#Jac_C\big(\mathbb{F}_q\big) = \chi(1)$, поэтому вычисление числа точек эквивалентно нахождению $\chi(T)$. Кроме того, имеются границы (Хассе — Вейля и более точные для рода 2 [4, с. 3, (1)]) для коэффициентов $\chi(T)$ и числа точек:

$$|s_1| \le 4\sqrt{q},\ 2|s_1|\sqrt{q} - 2q \le s_2 \le \frac{s_1^2}{4} + 2q$$

$$\text{и } \big(\sqrt{q} - 1\big)^4 \le \#Jac_C\big(\mathbb{F}_q\big) \le \big(\sqrt{q} + 1\big)^4.$$

На множестве классов дивизоров Jac_C определена операция сложения дивизоров, что делает ее группой, однако порядок этой группы трудно вычислим — сложность нахождения порядка равна $O(log^8 q)$ при использовании алгоритма Годри — Шоста [2], что находится на грани практического применения. Такие группы традиционно используются в дизайне криптосистем с открытым ключом благодаря большой сложности дискретного логарифмирования, а с недавнего времени — также в конструкциях верифицируемых функций задержки (VDF) [3], криптографических аккумуляторов [6] и ненадежных CHAPKов (Trustless SNARKs) [7].

Подсчет числа точек в якобиане необходим для оценки безопасности построенных криптографических примитивов и генерации параметров для криптосистем на основе дискретного логарифма (как альтернатива генерации кривых методом комплексного умножения).

Подход, предложенный Годри и Шостом [2; 4], состоит из двух частей.

1. Полиномиальная часть: нахождение числа точек или характеристического многочлена по модулю как можно большего числа малых

простых ℓ и объединение данной информации в единый модуль m по китайской теореме об остатках. Соответствующий алгоритм представлен в [2].

2. Экспоненциальная часть: зная число точек $\#Jac_C(\mathbb{F}_q)$ по модулю m, восстановить полное число точек $\#Jac_C(\mathbb{F}_q)$, используя алгоритмы на основе парадокса дней рождения. Соответствующий алгоритм впервые был предложен в [5] и требовал большого (экспоненциального) количества памяти. Годри и Шост [4] улучшили его, снизив объем требуемой памяти до полиномиального.

Данная работа посвящена оптимизации и параллельной реализации экспоненциальной части алгоритма Годри — Шоста. Рассмотрим его более подробно.

Алгоритм поиска коллизий Годри — Шоста

Рассматриваемый алгоритм поиска коллизий является аналогом алгоритма Мацуо — Чао — Цудзии [5] с меньшими затратами памяти.

Вход: гиперэллиптическая кривая $C/\mathbb{F}_q$: $y^2 = f(x)$ рода 2; m (простое число) — модуль, для которого известны $\overline{s_1}, \overline{s_2}$ такие, что

$$\overline{s_1} \equiv s_1 \bmod m,$$
$$\overline{s_2} \equiv s_2 \bmod m$$

(элементы $\overline{s_1}, \overline{s_2} \in \mathbb{F}_m$ вычисляются в полиномиальной части алгоритма Годри — Шоста).

Выход: $N = \chi(1)$, где $\chi(T) = T^4 - s_1T^3 + s_2T^2 - s_1qT + q^2$,

$$s_1 = \overline{s_1} + m\widetilde{s_1},$$
$$s_2 = \overline{s_2} + m\widetilde{s_2}.$$

Алгоритм:

1. Выберем границы в соответствии с [4]:

$$B_{1,min} \le s_1 \le B_{1,max},$$
$$B_{2,min} \le s_2 \le B_{2,max}.$$

2. Введем обозначения:

$$R = \{(\sigma_1, \sigma_2), \sigma_1 \in [B_{1,min}, B_{1,max}], \sigma_2 \in [B_{2,min}, B_{2,max}]\},$$
$$K' \coloneqq K + m\left(-\frac{\lfloor B_{1,min} + B_{1,max} \rfloor}{2}(q+1) + \frac{\lfloor B_{2,min} + B_{2,max} \rfloor}{2}\right),$$
$$W \coloneqq \{K'D + (-\sigma_1(q+1) + \sigma_2)mD, (\sigma_1, \sigma_2) \in R\},$$
$$T \coloneqq \{(-\sigma_1(q+1) + \sigma_2)mD, (\sigma_1, \sigma_2) \in R\}.$$

При этом имеем $|W| = |T| = \#R$, $|W \cap T| \subseteq [0{,}25 \cdot \#R, \#R]$.

3. Зададим псевдослучайное (детерминированное) блуждание на множествах W, T:

3.1. Зафиксируем параметры $r \ge 0$, e_1, e_2 (определяются на практике), где e_1, e_2 — средняя (ожидаемая) длина одной цепочки случайного блуждания.

3.2. Вычисляем сдвиги:

$$O_{k,k',b} = (-1)^b \cdot \alpha_{k,k'}(q+1) \cdot m \cdot D + \beta_{k,k'} \cdot m \cdot D \in Jac_C,$$

$\forall k, k' \in [1, r]$, $\forall b \in [0,1]$. Множители $\alpha_{k,k'}$ и $\beta_{k,k'}$ здесь выбраны из равномерного распределения на отрезках $[0, 2l_1]$ и $[0, 2l_2]$ соответственно.

3.3. Задаем хеш-функцию $H: Jac_C \to [1, r]$.

3.4. Задаем «особые» точки множеств W и T, выбрав для них критерий, соответствующий некоторому значению хеш-функций. Например, будем считать «особыми» точки $P_W \in W$ и $P_T \in T$ такие, что $H_1(P_W) = 0$ и, соответственно, $H_2(P_T) = 0$.

3.5. Создаем два пустых списка — *listWild* и *listTame*.

4. Запускаем случайное блуждание по множеству W:

4.1. Выбираем случайным образом пару $(\sigma_1, \sigma_2) \in R$ и соответствующую ей точку $P_W \in W$. Полагаем $\alpha_1 \coloneqq \sigma_1$, $\beta_1 \coloneqq \sigma_2$.

4.2. Если $H_1(P_W) \neq 0$, то есть точка не является «особой»:

4.2.1. Вычисляем $\alpha_1 \coloneqq \alpha_{H_1(P_W),H_2(P_W)}$, $\beta_1 \coloneqq \beta_{H_1(P_W),H_2(P_W)}$.

4.2.2. $\sigma_1 \coloneqq \sigma_1 + \alpha_1$, $\sigma_2 \coloneqq \sigma_2 + \beta_1$, $P_W \coloneqq P_W + O_{H_1(P_W),H_2(P_W),1}$.

4.2.3. Возвращаемся к шагу 4.2.

4.3. Сохраняем найденную «особую» точку $(P_W, \alpha_1, \beta_1) \to listWild$.

5. Запускаем случайное блуждание по множеству T:

5.1. Выбираем случайным образом пару $(\sigma_1, \sigma_2) \in R$ и соответствующую ей точку $P_T \in T$. Полагаем $\alpha_2 \coloneqq \sigma_1$, $\beta_2 \coloneqq \sigma_2$.

5.2. Если $H_2(P_T) \neq 0$, то есть точка не является «особой»:

5.2.1. Вычисляем $\alpha_2 \coloneqq \alpha_{H_1(P_T),H_2(P_T)}$, $\beta_2 \coloneqq \beta_{H_1(P_T),H_2(P_T)}$.

5.2.2. $\sigma_1 \coloneqq \sigma_1 + \alpha_2$, $\sigma_2 \coloneqq \sigma_2 + \beta_2$, $P_T \coloneqq P_T + O_{H_1(P_T),H_2(P_T),0}$.

5.2.3. Возвращаемся к шагу 5.2.

5.3. Сохраняем найденную особую точку $P_T \to listTame$.

Замечание. Случайные блуждания по множествам W и T, то есть шаги алгоритма 4.1—4.2 и 5.1—5.2, выполняются параллельно и независимо. Алгоритм завершается после того, как будет найдена точка $P = P_T = P_W$ (коллизия между списками $listWild$ и $listTame$).

Вернуть $N = K - (\alpha_1 - \alpha_2)(q+1)m + (\beta_1 - \beta_2)m$.

С учетом того, что $\widetilde{s_1}$ и $\widetilde{s_2}$ равномерно распределены на отрезках $[B_{1,min}, B_{1,max}]$ и $[B_{2,min}, B_{2,max}]$ соответственно, ожидаемое время работы алгоритма составляет $T \approx 2{,}43\sqrt{\#R} = 19{,}5\frac{q^{3/4}}{m}$.

Оптимизации алгоритма поиска коллизий

Для дальнейших оптимизаций алгоритма могут использоваться следующие методы, которые предлагается добавить в программную реализацию и протестировать на практике.

1. Метод Гэлбрейта — Рупрая [8] предполагает разбиение интервалов случайных блужданий $[B_{1,min}, B_{1,max}]$ и $[B_{2,min}, B_{2,max}]$ на два класса эквивалентности. При этом на шаге 4.2.1 вычисляются сразу две точки — P_W и $P_W{}'$, соответствующие эквивалентным значениям $\alpha_1 \sim \alpha_1{}'$, $\beta_1 \sim \beta_1{}'$. Предложенный метод теоретически позволяет уменьшить ожидаемое время работы алгоритма до $1{,}478\sqrt{\#R} = 11{,}9\frac{q^{3/4}}{m}$.

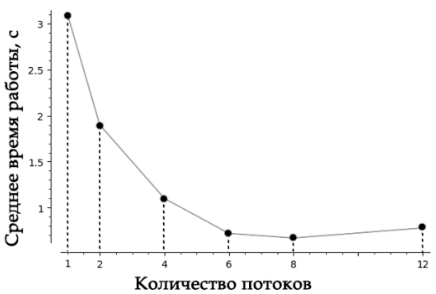
2. В анализе алгоритма Годри — Шоста предполагается, что распределение $\widetilde{s_1}$ и $\widetilde{s_2}$ равномерное. Этого достаточно для грубой оценки времени «в среднем». Но на самом деле распределение не является равномерным, и этот факт можно использовать для подсчета точек. Соответствующая оптимизация алгоритма была предложена в [10]. Распределение задается группой Сато — Тейта кривой, все возможные такие группы для рода 2 и соответствующие им распределения описаны в [11]. Заметим, что для кривой общего вида группа Сато — Тейта в большинстве случаев равна USp_4, и в алгоритме можно ограничиться данным случаем с соответствующим распределением ($\widetilde{s_1}$, $\widetilde{s_2}$). Другие группы Сато — Тейта соответствуют кривым с дополнительной структурой, например кривым с комплексным умножением, автоморфизмами, разложением якобиана на эллиптические кривые.

3. Метод Смита [9] для построения изогений между гиперэллиптическими и негиперэллиптическими кривыми рода 3. Данный метод использовался для вычисления дискретных логарифмов. Предполагается снижение сложности задачи с $O(q^{3/2})/m$ до $O(q/m)$.

Особенности реализации и результаты

За основу реализации взята экспериментальная программа Годри из пакета NTLJac2 [1], в которую было добавлено распараллеливание случайных блужданий на n потоков с помощью $std :: thread$. Реализация на C++ доступна под лицензией GNU GPL v2 и размещена на сервере GitHub. Представленные ниже результаты тестирования (рис. 1—3) были получены на машине с процессором XEON E-2146G (6 ядер/12 потоков) под управлением ОС Ubuntu 20.04.1 LTS, компилятор G++ из пакета GCC9.3.0.

В каждой из трех серий тестов была выбрана случайная кривая $y^2 = f(x)$ над конечными полями $\mathbb{F}_q$, где q — простое число, близкое к 2^{24}, 2^{30}, 2^{36} соответственно. Программа запускалась $N = 100$ раз с одинаковыми входными параметрами в каждом из режимов с числом потоков $\{1, 2, 4, 6, 8, 12\}$. Наибольшая эффективность реализации наблюдается при запуске в $n = 6$ потоков.

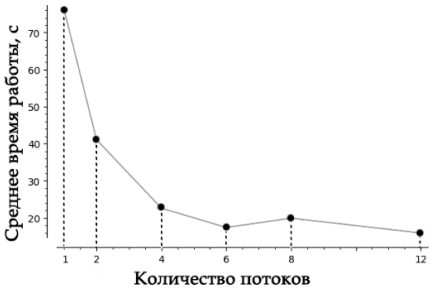


$$q = 16784711$$

$$f(x) = x^5 + 66737x^4 + 1073741x^3 + \\ +3827x^2 + 11537x + 374189$$

$$N = 100$$

Рис. 1. Время работы для $q \sim 24$ бит

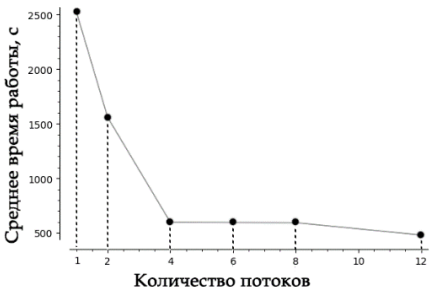


$$q = 107374127$$

$$f(x) = x^5 + 6673747x^4 + 10737411x^3 + \\ +73783827x^2 + 11537427x + 3741899$$

$$N = 100$$

Рис. 2. Время работы для $q \sim 30$ бит



$$q = 100000000003$$

$$f(x) = x^5 + 44951725753x^4 + \\ +35549991545x^3 + 14731848372x^2 + \\ +72263994727x + 67183483819$$

$$N = 100$$

Рис. 3. Время работы для $q \sim 36$ бит

Список литературы

1. *Gaudry P.* NTLJac2, Tools for genus 2 Jacobians in NTL. URL: http://www.lix.polytechnique.fr/Labo/Pierrick.Gaudry/NTLJac2/ (дата обращения: 13.10.2020).

2. *Gaudry P., Schost É.* Genus 2 point counting over prime fields // J. of Symbolic Computation. 2012. Vol. 47, № 4. P. 368—400.

3. *Dobson S., Galbraith S.D., Smith B.* Trustless Groups of Unknown Order with Hyperelliptic Curves // Cryptology ePrint Archive, Report 2020/196. URL: https://eprint.iacr.org/2020/196 (дата обращения: 10.09.2020).

4. *Gaudry P., Schost É.*A low-memory parallel version of Matsuo, Chao, and Tsujii's algorithm // International Algorithmic Number Theory Symposium. Springer, 2004. P. 208—222.

5. *Matsuo K., Chao J., Tsujii S.* An improved baby step giant step algorithm for point counting of hyperelliptic curves over finite fields // International Algorithmic Number Theory Symposium. Springer, 2002. P. 461—474.

6. *Boneh D., Bunz B., Fisch B.* Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains // Cryptology ePrint Archive, Report 2018/1188. URL: https://eprint.iacr.org/2018/1188.pdf (дата обращения: 10.09.2020).

7. *Bünz B., Fisch B., Szepieniec A.* Transparent SNARKs from DARK Compilers // Cryptology ePrint Archive, Report 2019/1229. URL: https://eprint.iacr.org/2019/1229.pdf (дата обращения: 10.09.2020).

8. *Galbraith S., Ruprai R.S.* Using Equivalence Classes to Accelerate Solving the Discrete Logarithm Problem in a Short Interval // International Workshop on Public Key Cryptography. Springer, 2010. P. 368—383.

9. *Smith B.* Isogenies and the discrete logarithm problem in Jacobians of genus 3 hyperelliptic curves // Advances in Cryptology — Eurocrypt. Springer, 2008. P. 163—180.

10. *Kedlaya K.S., Sutherland A.V.* Computing L-series of hyperelliptic curves // International Algorithmic Number Theory Symposium. Springer, 2008. P. 312—326.

11. *Fité F., Kedlaya K.S., Rotger, V., Sutherland, A.V.* Sato-Tate distributions and Galois endomorphism modules in genus 2 // Compositio Mathematica. 2012. T. 148, №5. P. 1390—1442.

Об авторе

Никита Сергеевич Колесников — мл. науч. сотр., Балтийский федеральный университет им. И. Канта, Россия.

E-mail: NiKolesnikov1@kantiana.ru

The author

Nikita S. Kolesnikov, Research Fellow, Immanuel Kant Baltic Federal University, Russia.

E-mail: NiKolesnikov1@kantiana.ru