# МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 510.52 DOI 10.17223/20710410/65/6

## О ГЕНЕРИЧЕСКОЙ СЛОЖНОСТИ ПРОБЛЕМЫ ВЫЧИСЛЕНИЯ ФУНКЦИИ ЭЙЛЕРА[1]

А. Н. Рыбалов

*Институт математики им. С. Л. Соболева СО РАН, г. Омск, Россия*

**E-mail:** alexander.rybalov@gmail.com

Изучается генерическая сложность проблемы вычисления функции Эйлера, имеющей важное значение для современной криптографии. Например, на предположении о её трудноразрешимости основывается криптостойкость знаменитой системы шифрования с открытым ключом RSA. Доказывается, что при условии трудноразрешимости этой проблемы в худшем случае и P = BPP для её решения не существует полиномиального сильно генерического алгоритма. Для сильно генерического полиномиального алгоритма нет эффективного метода случайной генерации входов, на которых этот алгоритм не может решить проблему. Таким образом, этот результат обосновывает применение проблемы вычисления функции Эйлера в криптографии с открытым ключом. Для доказательства теоремы используется метод генерической амплификации, который позволяет строить генерически трудные проблемы из проблем, трудных в худшем случае. Основной идеей этого метода является объединение эквивалентных входов в достаточно большие множества. Эквивалентность входов означает, что рассматриваемая проблема на них решается одинаково.

**Ключевые слова:** *генерическая сложность, функция Эйлера.*

## ON THE GENERIC COMPLEXITY OF THE PROBLEM OF COMPUTING THE EULER FUNCTION

A. N. Rybalov

*Sobolev Institute of Mathematics, Omsk, Russia*

We study the generic complexity of the problem of the Euler function computation. This problem has important applications in modern cryptography. For example, the cryptographic strength of the famous public key encryption system RSA is based on the assumption of its hardness. We prove that under the condition of worst-case hardness and P = BPP there is no polynomial strongly generic algorithm for this problem. For a strongly generic polynomial algorithm, there is no efficient method for random generation of inputs on which the algorithm cannot solve the problem. Thus, this result justifies the application of the problem of computing the Euler function in public key cryptography. To prove this theorem, we use the method of generic amplification,

[1]Работа поддержана грантом Российского научного фонда № 22-11-20019.

which allows us to construct generically hard problems from the problems that are hard in the classical sense. The main feature of this method is the cloning technique, which combines the input data of a problem into sufficiently large sets of equivalent input data. Equivalence is understood in the sense that the problem is solved in a similar way for them.

**Keywords:** *generic complexity, Euler function.*

## Введение

В современной криптографии интересны такие алгоритмические проблемы, которые, являясь (гипотетически) трудными в классическом смысле, остаются трудными и в генерическом смысле [1], т. е. для почти всех входов. Это объясняется тем, что при случайной генерации ключей в криптографическом алгоритме происходит генерация входа некоторой трудной алгоритмической проблемы, лежащей в основе криптостойкости алгоритма. Если проблема легкоразрешима почти всегда, то для почти всех таких входов её можно будет быстро решить и ключи почти всегда будут нестойкими. Поэтому проблема должна быть трудной для почти всех входов. Например, таким поведением обладают классические алгоритмические проблемы криптографии: распознавания квадратичных вычетов [2], дискретного логарифма [3], извлечения корня в группах вычетов [4].

Функция Эйлера $\varphi(x)$, равная количеству натуральных чисел, меньших $x$ и взаимно простых с $x$, играет важную роль в современной криптографии. Для её вычисления до сих пор неизвестно эффективных (полиномиальных) алгоритмов [5]. Этот факт, среди прочего, используется для обоснования стойкости знаменитого алгоритма шифрования с открытым ключом RSA [6]. Проблема вычисления функции Эйлера тесно связана с известной проблемой факторизации (разложения на множители) целых чисел: если бы существовал полиномиальный алгоритм для проблемы факторизации, то его можно было бы использовать для эффективного вычисления функции Эйлера. Однако для проблемы факторизации также неизвестно эффективных алгоритмов [5].

В данной работе изучается генерическая сложность проблемы вычисления функции Эйлера. Доказывается, что при условии трудноразрешимости этой проблемы в худшем случае и $\mathrm{P} = \mathrm{BPP}$ для неё не существует полиномиального сильно генерического алгоритма. Для сильно генерического полиномиального алгоритма нет эффективного метода случайной генерации входов, на которых этот алгоритм не может решить проблему. Таким образом, этот результат обосновывает применение проблемы вычисления функции Эйлера в криптографии с открытым ключом. Здесь класс BPP состоит из проблем, разрешимых за полиномиальное время на вероятностных машинах Тьюринга. Считается, что класс BPP совпадает с классом P, то есть любой полиномиальный вероятностный алгоритм можно эффективно дерандомизировать, построив полиномиальный алгоритм, не использующий генератор случайных чисел и решающий ту же самую проблему. Хотя равенство $\mathrm{P} = \mathrm{BPP}$ до сих пор не доказано, имеются веские основания в его пользу [7].

## 1. Предварительные сведения

В данной работе множеством входов для алгоритмов является множество натуральных чисел $\mathbb{N}$, записанных в двоичной форме. Под размером $\mathrm{size}(x)$ натурального числа $x$ понимается длина его двоичной записи.

Для подмножества $S \subseteq \mathbb{N}$ определим *последовательность относительных плотностей*

$$\rho_n(S) = \frac{|S_n|}{|\mathbb{N}_n|},\ n = 1, 2, 3, \ldots,$$

где $\mathbb{N}_n$ — множество натуральных чисел размера $n$; $S_n = S \cap \mathbb{N}_n$. Здесь для любого конечного множества через $|A|$ обозначено число его элементов. Легко проверить, что $|\mathbb{N}_n| = 2^{n-1}$.

*Асимптотической плотностью* множества $S$ назовём верхний предел

$$\rho(S) = \overline{\lim_{n\to\infty}} \rho_n(S).$$

Множество $S$ называется *генерическим*, если $\rho(S) = 1$, и *пренебрежимым*, если $\rho(S) = 0$. Назовём множество $S$ *сильно пренебрежимым*, если последовательность $\rho_n(S)$ экспоненциально быстро сходится к нулю, т. е. существуют константы $\sigma$, $0 < \sigma < 1$, и $C > 0$, такие, что для любого $n$

$$\rho_n(S) < C\sigma^n.$$

Множество $S$ называется *сильно генерическим*, если его дополнение $\mathbb{N} \setminus S$ сильно пренебрежимо.

Алгоритм $\mathcal{A} : \mathbb{N} \to \mathbb{N} \cup \{?\}$ называется (*сильно*) *генерическим*, если:

1) $\mathcal{A}$ останавливается на всех входах из $\mathbb{N}$;
2) множество $\{x \in \mathbb{N} : \mathcal{A}(x) \neq ?\}$ является (сильно) генерическим.

Генерический алгоритм $\mathcal{A}$ вычисляет функцию $f : \mathbb{N} \to \mathbb{N}$, если для всех $x \in \mathbb{N}$

$$\big(\mathcal{A}(x) \neq ?\big) \ \Rightarrow \big(\mathcal{A}(x) = f(x)\big).$$

Имеется существенное различие между генерически разрешимыми проблемами и сильно генерически разрешимыми проблемами. Для сильно генерического полиномиального алгоритма нет эффективного метода случайной генерации входов, на которых этот алгоритм не может решить проблему. Допустим, имеется проблема $S$, разрешимая на некотором разрешимом за полиномиальное время генерическом множестве $G \subseteq \mathbb{N}$, для которого

$$\frac{|G \cap \mathbb{N}_n|}{|\mathbb{N}_n|} = \frac{n-1}{n}.$$

Таким образом, $G$ — генерическое, но не сильно генерическое множество. Теперь хоть и проблема $S$ разрешима для почти всех входов, тем не менее есть эффективный способ получить «плохой» вход, на котором генерический алгоритм не работает. Полиномиальный алгоритм для генерации плохих входов следующий:

1) сгенерировать равномерно случайный вход $x$ размера $n$;
2) если $x \in G$, повторить шаг 1, иначе закончить.

Действительно, вероятность получить только хорошие входы за $n^2$ раундов равна

$$\left(\frac{n-1}{n}\right)^{n^2} = \left(\left(1 - \frac{1}{n}\right)^n\right)^n \to e^{-n}.$$

Поэтому с вероятностью, очень близкой к 1, будет получен плохой вход. С другой стороны, легко видеть, что если проблема разрешима на сильно генерическом множестве, то такой алгоритм генерации потребует экспоненциального числа раундов и

будет неэффективным. Для приложений к криптографии это означает, что просто генерическая легкоразрешимость проблемы не делает эту проблему бесполезной для создания на ее основе криптосистемы, так как для неё существует эффективная процедура генерации трудных входов. В то же время сильно генерически легкоразрешимые проблемы в этом смысле бесполезны для криптографии.

Напомним некоторые понятия классической теории сложности вычислений [8]. *Время работы* $t_M(x)$ машины Тьюринга $M$ на входе $x \in \mathbb{N}$ — это число шагов машины от начала работы до остановки. Машина Тьюринга $M$ *полиномиальна*, если существует полином $p(n)$, такой, что для любого $x \in \mathbb{N}$ имеет место $t_M(x) < p(\mathrm{size}(x))$. Класс P состоит из подмножеств $\mathbb{N}$, распознаваемых полиномиальными машинами Тьюринга.

*Вероятностная машина Тьюринга* — это машина Тьюринга, в программе которой допускаются пары недетерминированных правил, которые одновременно применимы в данной ситуации. В процессе работы такой машины с вероятностью $1/2$ выбирается первое правило и с вероятностью $1/2$ — второе.

Время работы $t_M(x, \tau)$ вероятностной машины Тьюринга на входе $x$ зависит от вычислительного пути (последовательности выполненных команд) $\tau$. Вероятностная машина Тьюринга $M$ называется *полиномиальной*, если существует полином $p(n)$, такой, что для любого $x$ и для любого вычислительного пути $\tau$ машины $M$ на $x$ имеет место $t_M(x, \tau) < p(\mathrm{size}(x))$.

Обозначим через $\Pr[M(x) = y]$ вероятность того, что машина $M$ на входе $x$ выдаёт ответ $y$. Вероятностная машина $M$ *вычисляет* функцию $f : \mathbb{N} \to \mathbb{N}$, если для любого $x \in \mathbb{N}$ имеет место

$$\big(f(x) = y\big) \;\Rightarrow \Pr[M(x) = y] > 2/3.$$

Проблема распознавания множества $S \subseteq \mathbb{N}$ принадлежит классу BPP, если существует вероятностная полиномиальная машина Тьюринга $M$, вычисляющая характеристическую функцию множества $S$:

$$\chi_S(x) = \begin{cases} 1, & \text{если } x \in S, \\ 0, & \text{если } x \notin S. \end{cases}$$

Вероятностные машины Тьюринга формализуют понятие алгоритма, использующего генератор случайных чисел. Класс BPP — это класс проблем, эффективно решаемых такими вероятностными алгоритмами.

## 2. Функция Эйлера

Функция Эйлера $\varphi(x)$ для любого натурального числа $x$ возвращает количество натуральных чисел, меньших $x$ и взаимно простых с $x$. Важным свойством этой функции является её мультипликативность: для любых взаимно простых $x$ и $y$ имеет место $\varphi(xy) = \varphi(x)\varphi(y)$. Если $x = p_1^{k_1} \dots p_m^{k_m}$ — разложение числа $x$ по степеням простых, то

$$\varphi(x) = (p_1^{k_1} - p_1^{k_1-1}) \dots (p_m^{k_m} - p_m^{k_m-1}) = x\Big(1 - \frac{1}{p_1}\Big) \dots \Big(1 - \frac{1}{p_m}\Big). \tag{1}$$

В частности, $\varphi(p) = p - 1$ для любого простого $p$.

Формула (1) показывает связь проблемы вычисления функции Эйлера с известной проблемой факторизации (разложения на множители) целых чисел: если бы существовал полиномиальный алгоритм для проблемы факторизации, то его можно было бы использовать для эффективного вычисления функции Эйлера. Однако до сих пор для этих двух проблем неизвестно полиномиальных алгоритмов [5].

**Лемма 1.** Существует полиномиальный алгоритм, который для любого натурального числа $x$ и любого простого $p$, такого, что $p$ не делит $x$, по входу $(x, p, \varphi(xp))$ находит значение $\varphi(x)$.

***Доказательство.*** Из свойства мультипликативности функции Эйлера следует, что $\varphi(xp) = (p-1)\varphi(x)$. Таким образом, искомый полиномиальный алгоритм работает следующим образом: на вход ему подаются числа $x$, $p$ и $\varphi(xp)$; алгоритм делит $\varphi(xp)$ на $p-1$ и находит значение $\varphi(x)$. Это делается за полиномиальное от размеров чисел $x$, $p$ и $\varphi(xp)$ время. ■

Для произвольного натурального числа $a$ размера $n$ определим множество

$$S(a) = \{pa : p \ - \text{простое}, \ \text{size}(p) = n^2\}.$$

**Лемма 2.** Для любого достаточно большого $a$ имеет место

$$\frac{|S(a)|}{|\mathbb{N}_{n^2+n}|} > \frac{1}{n^2 2^{n+1}}.$$

***Доказательство.*** Оценим снизу число $|S(a)|$. Заметим, что

$$|S(a)| = \pi(2^{n^2}) - \pi(2^{n^2-1}), \tag{2}$$

где функция $\pi(x)$ определяет число простых чисел, не превосходящих $x$. Из асимптотического закона распределения простых чисел следует, что для достаточно больших $x$ имеют место оценки

$$0{,}9 \cdot \log e \cdot \frac{x}{\log x} < \pi(x) < 1{,}1 \cdot \log e \cdot \frac{x}{\log x},$$

где $\log x$ — двоичный логарифм $x$. Поэтому, учитывая (2), имеем

$$|S(a)| > 0{,}9 \cdot \log e \cdot \frac{2^{n^2}}{n^2} - 1{,}1 \cdot \log e \cdot \frac{2^{n^2-1}}{n^2-1} = \log e \cdot 2^{n^2-1}\left(\frac{1{,}8}{n^2} - \frac{1{,}1}{n^2-1}\right) >$$
$$> \log e \cdot 2^{n^2-1}\left(\frac{1{,}8}{n^2} - \frac{1{,}2}{n^2}\right) = \frac{0{,}6 \cdot \log e \cdot 2^{n^2-1}}{n^2} > \frac{2^{n^2-2}}{n^2}.$$

Так как $|\mathbb{N}_{n^2+n}| = 2^{n^2+n-1}$, получаем

$$\frac{|S(a)|}{|\mathbb{N}_{n^2+n}|} > \frac{2^{n^2-2}}{n^2 \cdot 2^{n^2+n-1}} = \frac{1}{n^2 2^{n+1}}.$$

Лемма 2 доказана. ■

**Лемма 3.** Полиномиальный алгоритм для вычисления функции Эйлера $\varphi(x)$ существует тогда и только тогда, когда существует полиномиальный алгоритм для распознавания множества $EF = \{(x, k) : \varphi(x) \leqslant k \leqslant x\}$.

***Доказательство.*** Пусть существует полиномиальный алгоритм для вычисления функции Эйлера. Тогда для того, чтобы определить, принадлежит ли пара $(x, k)$ множеству $EF$, надо просто вычислить значение $\varphi(x)$ и сравнить его с $k$.

Обратно, пусть существует полиномиальный алгоритм $\mathcal{A}$ для распознавания множества $EF$. Тогда для вычисления значения $\varphi(x)$ нужно действовать следующим образом:

1) $L := 1$, $R := x - 1$;
2) $C := [(L + R)/2]$;
3) если $(x, C) \in EF$, то $R := C$, иначе $L := C$;
4) если $L \neq R$, то вернуться на шаг 2, иначе перейти на шаг 5;
5) выдать $\varphi(x) = R$.

Здесь через $[x]$ обозначена целая часть числа $x$. В этом алгоритме на каждой итерации уточняются границы $L \leqslant \varphi(x) \leqslant R$ до тех пор, пока отрезок $[L, R]$ не «схлопнется» в точку и мы не получим точное значение $\varphi(x)$. Так как длина отрезка на каждой итерации делится пополам, число итераций ограничено значением $[\log x]$ — размером числа $x$. Поэтому алгоритм полиномиален. ■

## 3. Основной результат

**Теорема 1.** Если существует сильно генерический полиномиальный алгоритм, вычисляющий функцию Эйлера, то существует вероятностный полиномиальный алгоритм, вычисляющий функцию Эйлера на всём множестве входов.

***Доказательство.*** Пусть существует сильно генерический полиномиальный алгоритм $\mathcal{A}$, вычисляющий функцию Эйлера. Построим по $\mathcal{A}$ вероятностный полиномиальный алгоритм $\mathcal{B}$, вычисляющий функцию Эйлера на всём множестве входов. На натуральном числе $a$ размера $n$ алгоритм $\mathcal{B}$ работает следующим образом:

1) Повторяет $4n^2$ раз шаги 2–9.
2) Генерирует случайно равновероятно натуральное нечётное число $p$ размера $n^2$.
3) С помощью полиномиального алгоритма Агравала — Каяла — Саксены [9] проверяет, является ли $p$ простым числом.
4) Если $p$ простое, переходит на шаг 7.
5) Возвращается на шаг 2.
6) Если за $4n^2$ шагов не получено простое число, то останавливается и выдаёт ответ 0.
7) Иначе запускает алгоритм $\mathcal{A}$ на числе $ap$.
8) Если $\mathcal{A}(ap) = \varphi(ap)$, то, по лемме 1, находит за полиномиальное время значение функции Эйлера для $a$.
9) Если $\mathcal{A}(ap) = ?$, то выдаёт ответ 0.

Заметим, что полиномиальный вероятностный алгоритм $\mathcal{B}$ выдаёт правильный ответ на шаге 8, а на шагах 6 и 9 — неправильный. Нужно доказать, что вероятность того, что ответ выдаётся на шаге 6 или шаге 9, меньше $1/3$.

Оценим вероятность выдачи ответа на шаге 6. Это бывает, только если за $n^2$ раундов генерации числа $p$ не было получено простое число. Вероятность того, что простое число не получается за один раунд, равна

$$1 - \frac{\pi(2^{n^2}) - \pi(2^{n^2-1})}{2^{n^2-1}} < 1 - \frac{2^{n^2-2}}{n^2 \cdot 2^{n^2-1}} = 1 - \frac{1}{2n^2}.$$

Здесь верхняя оценка получается аналогично доказательству леммы 2. Вероятность того, что простое число не получится за все $4n^2$ раундов, не превосходит

$$\left(1 - \frac{1}{2n^2}\right)^{4n^2} < e^{-2} < 0{,}15.$$

Оценим теперь вероятность выдачи ответа на шаге 9. Число $a$ имеет размер $n$, значит, размер числа $ap$ равен $n^2+n$. Вероятность того, что $\mathcal{A}(ap)=?$, не больше

$$\frac{|\{x\in\mathbb{N}:\mathcal{A}(x)=?\}_{n^2+n}|}{|S(a)|}=\frac{|\{x\in\mathbb{N}:\mathcal{A}(x)=?\}_{n^2+n}|}{|\mathbb{N}_{n^2+n}|}\,\frac{|\mathbb{N}_{n^2+n}|}{|S(a)|}.$$

По лемме 2

$$\frac{|\mathbb{N}_{n^2+n}|}{|S(a)|}<n^2 2^{n+1}.$$

Так как множество $\{x\in\mathbb{N}:\mathcal{A}(x)=?\}$ сильно пренебрежимое, то существует константа $\alpha>0$, такая, что

$$\frac{|\{x\in\mathbb{N}:\mathcal{A}(x)=?\}_{n^2+n}|}{|\mathbb{N}_{n^2+n}|}<\frac{1}{2^{\alpha(n^2+n)}}$$

для любого $n$. Поэтому искомая вероятность ответа на шаге 9 не больше

$$\frac{n^2 2^{n+1}}{2^{\alpha(n^2+n)}}=\frac{n^2}{2^{\alpha(n^2+n)-n-1}}<0{,}15$$

при больших $n$. Таким образом, вероятность ответа на шагах 6 и 9 не превосходит $0{,}15+0{,}15<1/3$. ■

**Теорема 2.** Если для вычисления функции Эйлера не существует полиномиального алгоритма и P = BPP, то для её вычисления не существует сильно генерического полиномиального алгоритма.

***Доказательство.*** Пусть существует сильно генерический алгоритм, вычисляющий функцию Эйлера. Тогда, по теореме 1, существует вероятностный полиномиальный алгоритм, вычисляющий её на всём множестве входов. Этот же алгоритм решает проблему распознавания множества $EF$, которая, по лемме 3, полиномиально эквивалентна проблеме вычисления функции Эйлера. Таким образом, проблема $EF$ лежит в классе BPP. Так как P = BPP, то она лежит и в классе P, а значит, для проблемы вычисления функции Эйлера существует полиномиальный алгоритм. Противоречие. ■

## ЛИТЕРАТУРА


1. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks // J. Algebra. 2003. V. 264. No. 2. P. 665–694.
2. *Рыбалов А. Н.* О генерической сложности проблемы распознавания квадратичных вычетов // Прикладная дискретная математика. 2015. № 2 (28). С. 54–58.
3. *Рыбалов А. Н.* О генерической сложности проблемы дискретного логарифма // Прикладная дискретная математика. 2016. № 3 (33). С. 93–97.
4. *Рыбалов А. Н.* О генерической сложности проблемы извлечения корня в группах вычетов // Прикладная дискретная математика. 2017. № 38. С. 95–100.
5. *Adleman L. M. and McCurley K. S.* Open problems in number theoretic complexity, II // LNCS. 1994. V. 877. P. 291–322.
6. *Rivest R., Shamir A., and Adleman L.* A method for obtaining digital signatures and public-key cryptosystems // Commun. ACM. 1978. V. 21. Iss. 2. P. 120–126.
7. *Impagliazzo R. and Wigderson A.* P = BPP unless E has subexponential circuits: Derandomizing the XOR Lemma // Proc. 29th STOC. El Paso, ACM, 1997. P. 220–229.
8. *Вялый М., Китаев А., Шень А.* Классические и квантовые вычисления. М.: МЦНМО, ЧеРо. 1999. 192 с.
9. *Agrawal M., Kayal N., and Saxena N.* PRIMES is in P // Ann. Math. 2004. V. 160. No. 2. P. 781–793.

REFERENCES

1. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks. J. Algebra, 2003, vol. 264, no. 2, pp. 665–694.
2. *Rybalov A. N.* O genericheskoy slozhnosti problemy raspoznavaniya kvadratichnykh vychetov [On generic complexity of the quadratic residuosity problem]. Prikladnaya Diskretnaya Matematika, 2015, no. 2 (28), pp. 54–58. (in Russian)
3. *Rybalov A. N.* O genericheskoy slozhnosti problemy diskretnogo logarifma [On generic complexity of the discrete logarithm problem]. Prikladnaya Diskretnaya Matematika, 2016, no. 3 (33), pp. 93–97. (in Russian)
4. *Rybalov A. N.* O genericheskoy slozhnosti problemy izvlecheniya kornya v gruppakh vychetov [On generic complexity of the problem of finding roots in groups of residues]. Prikladnaya Diskretnaya Matematika, 2017, no. 38, pp. 95–100. (in Russian)
5. *Adleman L. M. and McCurley K. S.* Open problems in number theoretic complexity, II. LNCS, 1994, vol. 877, pp. 291–322.
6. *Rivest R., Shamir A., and Adleman L.* A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM, 1978, vol. 21, iss. 2, pp. 120–126.
7. *Impagliazzo R. and Wigderson A.* P = BPP unless E has subexponential circuits: Derandomizing the XOR Lemma. Proc. 29th STOC, El Paso, ACM, 1997, pp. 220–229.
8. *Vyalyy M., Kitaev A., and Shen' A.* Klassicheskie i kvantovye vychisleniya [Classical and Quantum Computations]. Moscow, MCCME, CheRo, 1999. 192 p. (in Russian)
9. *Agrawal M., Kayal N., and Saxena N.* PRIMES is in P. Ann. Math., 2004, vol. 160, no. 2, pp. 781–793.