

# Математические основы защиты информации

## Лабораторная работа №5

### Факторизация целых чисел (экспоненциальные методы)

БГУ, ММФ, каф. ДУиСА,  
доцент Чергинец Д.Н.


































#### $p$ -метод Полларда

$p$ -метод Полларда основан на парадоксе дней рождения, который заключается в следующем. Чтобы вероятность того, что ваш день рождения совпадает с днем рождения кого-нибудь из группы людей, была больше  $1/2$  необходимо, чтобы группа состояла более чем из  $366/2 = 183$  человек. А на самом деле еще больше, так как среди 183 человек могут быть люди с одинаковыми днями рождения. В то же время вероятность того, что в группе из 23 человек найдутся два человека с одинаковым днем рождения, больше  $1/2$ . Парадокс заключается в том, что случайные события похожи, вероятности обоих больше  $1/2$ , однако для первого события необходима группа из 183 человек, а для второго достаточно лишь 23. Более того, в группе из 58 человек вероятность того, что в ней найдутся два человека с одинаковыми днями рождения превышает 99%. Хотя с логической точки зрения парадокса нет, это лишь теория вероятностей.

С точки зрения данного парадокса достаточно взять не так много элементов  $x_0, x_1, \dots, x_l \in \mathbb{Z}_n$ , чтобы среди них нашлись какие-нибудь два  $x_j, x_k$  такие, что  $x_j \equiv x_k \pmod{p}$ , где  $p$  — один из делителей числа  $n$ . Тогда  $p$  можно будет найти по формуле  $p = \text{GCD}[x_k - x_j, n]$ .

**Алгоритм. Черепаха с генератором случайных чисел.**

                                        . Составное число  $n \in \mathbb{N}$ .

                                                . Нетривиальный делитель числа  $n$ .

1. Задаем начальные данные:  $i := 0$ , выбираем случайное  $x_i \in \mathbb{N}$ ,  $x_i < n$ .
2. Присваиваем  $i := i + 1$  и выбираем случайное  $x_i \in \mathbb{N}$ ,  $x_i < n$ .
3. Для  $j = 0, 1, \dots, i - 1$  выполняем шаги 3.1, 3.2.
  - 3.1. Вычисляем  $d := \text{GCD}(x_i - x_j, n)$ .
  - 3.2. Если  $1 < d < n$ , то выдаем результат  $d$ , конец алгоритма.
4. Переходим к шагу 2.

$\rho$ -метод Полларда заключается в выборе случайной пары  $(f, x_0)$  (где  $f: \mathbb{Z}_n \rightarrow \mathbb{Z}_n$ ,  $x_0 \in \mathbb{Z}_n$ , на практике в качестве функции  $f(x)$  берут многочлен) и вычислении последовательности  $x_0, x_1, \dots, x_L \in \mathbb{Z}_n$ ,  $x_i := f(x_{i-1})$ .

Мы имеем рекуррентную последовательность элементов конечного множества

$$x_0, x_1, \dots, x_L \in \mathbb{Z}_n, x_i := f(x_{i-1}).$$

Найдутся такие  $j, k \in \mathbb{N}$ , что  $x_j \equiv x_k \pmod{p}$ , причем  $x_{j+m} \equiv x_{k+m} \pmod{p}$  для всех  $m \in \mathbb{N}$ .

Первый способ найти такие  $j, k$ , это метод полного перебора, он заключается в том, что новый вычисленный элемент  $x_i$  сравнивают с каждым вычисленным ранее  $x_0, x_1, \dots, x_{i-1}$ . В этом случае наибольший общий делитель  $\text{GCD}[x_i - x_j, n]$  необходимо вычислить  $1 + 2 + 3 + \dots + k = \frac{1+k}{2} k$  раз. Вычисления нужно проводить пока не окажется, что  $\text{GCD}[x_i - x_j, n] > 1$ , если при этом  $\text{GCD}[x_i - x_j, n] < n$ , то  $\text{GCD}[x_i - x_j, n]$  является делителем.

**Задание 1.**

Реализовать алгоритм **Черепаха с генератором случайных чисел**. Является ли данный алгоритм:

- вероятностным?
- детерминированным?
- всегда ли выдает ответ?

Модифицируем данный алгоритм. Случайную последовательность будем строить рекуррентно при помощи функции

$$x_i := f(x_{i-1}),$$

где  $f: \mathbb{Z}_n \rightarrow \mathbb{Z}_n$ ,  $x_0 \in \mathbb{Z}_n$ .


Как правило используют функцию


$$f(x) := x^2 + a \pmod{n},$$

где  $a$  – входящий параметр алгоритма,  $-10 < a < 10$ , его берут небольшим, что уменьшить вычислительную сложность алгоритма.

Теперь алгоритм Черепаха выглядит следующим образом (измененные пункты выделены **шрифтом**):

Алгоритм. **Черепаха**.

 Составное число  $n \in \mathbb{N}$ ,  $a, x_0 \in \mathbb{Z}$ .

 Нетривиальный делитель числа  $n$ .

1. Задаем начальные данные:  $i := 0$ ,  $f(x) := x^2 + a \pmod{n}$ .
2. Присваиваем  $i := i + 1$ ,  $x_i := f(x_{i-1})$ .
3. Для  $j = 0, 1, \dots, i - 1$  выполняем шаги 3.1, 3.2, 3.3.
  - 3.1. Вычисляем  $d := \text{GCD}(x_i - x_j, n)$ .
  - 3.2. Если  $1 < d < n$ , то выдаем результат  $d$ , конец алгоритма.
  - 3.3. Если  $d = n$ , то выдаем результат “делитель не найден, возьмите другую последовательность”, конец алгоритма.
4. Переходим к шагу 2.



























## Задание 2.






















Реализовать алгоритм **Черепаха**. Сравнить скорость вычислений с алгоритмом **Черепаха с генератором случайных чисел**. Является ли данный алгоритм:

- вероятностным?
- детерминированным?

Следующий метод нахождения одинаковых элементов в последовательности, это метод Флойда или метод Черепахи и Зайца. Он заключается в том, что сравниваются лишь элементы  $x_i$  и  $x_{2i}$ , где  $x_i$  – Черепаха (Tortoise),  $x_{2i}$  – Заяц (Hare). Нечетный элемент  $x_{2i+1}$  вообще не сравнивается.

**Алгоритм. Черепаха и Заяц с хранением последовательности.**

                         . Составное число  $n \in \mathbb{N}$ ,  $a, x_0 \in \mathbb{Z}$ .

                         . Нетривиальный делитель числа  $n$ .

1. Задаем начальные данные:  $i := 0$ ,  $f(x) := x^2 + a \pmod{n}$ .
2. Присваиваем  $i := i + 1$ ,  $x_{2i-1} := f(x_{2i-2})$ ,  $x_{2i} := f(x_{2i-1})$ .
3. Вычисляем  $d := \text{GCD}(x_{2i} - x_i, n)$ .
4. Если  $1 < d < n$ , то выдаем результат  $d$ , конец алгоритма.
5. Если  $d == n$ , то выдаем результат “делитель не найден, возьмите другую последовательность”, конец алгоритма.
6. Переходим к шагу 2.

**Задание 3.**

Реализовать алгоритм **Черепаха и Заяц с хранением последовательности**. Сравнить его с алгоритмом Черепаха.

**Задание 4.**

При помощи одного из реализованных выше алгоритмов найти делители чисел  $n_1$  и  $n_2$  и сравнить их время вычисления, объяснить, от чего оно зависит.

**Вариант 1.**

$n_1 = 15\,506\,156\,499\,551\,801\,071$ ;

$n_2 =$

62 868 210 912 435 676 236 063 110 171 611 552 388 850 726 581 321 443 446 871 402 813 418 011 834 \;  
255 546 376 426 442 433 367 780 232 341 829 909 876 174 267 356 648 704 288 645 335 775 262 924 \;  
915 608 120 133 502 414 785 983 748 439 328 046 795 993 277 337 869 292 939 901;

**Вариант 2.**

$n_1 = 59\,523\,843\,526\,082\,987\,497$ ;

$n_2 =$

79 923 374 586 285 156 147 820 038 497 546 494 825 176 412 377 564 495 897 351 070 054 076 985 013 \;  
313 637 663 550 307 709 700 492 789 018 226 106 959 662 590 786 932 816 156 783 375 519 786 976 \;  
435 854 115 343 950 135 169 694 546 580 722 585 598 589 389 999 500 342 042 863;

### Вариант 3.

$n_1 = 250\,555\,579\,698\,426\,283;$

$n_2 =$

41 220 545 481 281 889 966 969 492 498 051 061 589 102 446 563 174 388 598 626 134 746 729 206 699 \\  
882 766 512 826 923 833 210 650 429 764 673 812 076 262 586 428 127 165 108 100 437 049 590 467 \\  
499 913 625 296 023 687 050 676 867 783 808 603 120 181 062 808 207 599 561 741;

### Вариант 4.

$n_1 = 28\,583\,250\,660\,505\,205\,447;$

$n_2 =$

24 629 038 524 191 696 534 022 517 615 450 287 746 198 751 569 124 235 908 879 489 363 857 098 440 \\  
281 061 546 510 340 582 403 191 291 790 963 231 187 039 734 658 242 248 204 950 706 011 609 357 \\  
371 176 972 863 434 476 473 757 154 594 229 127 154 243 854 648 922 835 209 179;

### Вариант 5.

$n_1 = 447\,943\,286\,410\,807\,723;$

$n_2 =$

20 804 133 795 306 884 594 348 721 762 089 178 088 957 463 751 716 917 580 674 270 050 489 667 877 \\  
050 224 805 465 290 642 974 007 555 559 845 388 590 449 844 486 955 734 571 691 305 979 150 728 \\  
031 034 966 781 515 142 543 579 313 882 027 044 883 184 242 846 250 185 331 107;

### Вариант 6.

$n_1 = 78\,889\,106\,975\,637\,973\,169;$

$n_2 =$

1 705 859 568 610 261 576 367 982 241 748 580 347 227 807 548 677 472 990 284 706 511 240 429 295 \\  
335 914 824 937 508 446 730 059 206 398 152 204 298 240 889 682 924 092 931 009 056 651 548 822 \\  
384 502 519 671 850 289 426 312 956 051 591 458 992 245 847 469 118 868 192 647;

### Вариант 7.

$n_1 = 32\,928\,314\,705\,119\,407\,281;$

$n_2 =$

316 731 684 955 218 103 932 939 831 171 619 536 237 119 265 849 656 481 999 791 679 871 065 863 532 \\  
924 995 705 593 331 407 781 359 063 811 763 583 346 461 995 384 038 286 742 535 681 780 320 210 \\  
589 286 691 121 496 488 093 178 048 762 294 323 378 492 949 193 785 608 947;

### Вариант 8.

$n_1 = 3\,826\,639\,320\,665\,423\,579;$

$n_2 =$

24 539 302 806 595 788 422 136 609 186 903 152 826 590 905 191 986 868 326 530 214 429 016 351 797 \\  
257 238 459 035 855 534 435 095 857 861 062 792 595 222 455 975 261 425 995 428 951 937 482 306 \\  
286 362 422 990 970 185 973 074 729 975 880 220 075 558 841 006 906 042 777 677;

**Вариант 9.** $n_1 = 45\ 260\ 410\ 724\ 299\ 792\ 603;$  $n_2 =$ 

6 436 757 861 882 784 524 943 295 692 431 987 907 011 305 932 129 473 353 361 122 296 307 267 472 \

547 975 570 850 632 237 033 883 641 706 412 474 164 263 705 099 230 959 965 388 384 593 002 122 \

931 407 031 065 731 053 472 142 708 061 320 798 123 602 038 764 089 863 396 423;

**Вариант 10.** $n_1 = 17\ 872\ 915\ 729\ 489\ 265\ 701;$  $n_2 =$ 

3 497 238 429 261 178 430 121 604 330 458 488 917 322 924 762 581 518 325 042 646 524 800 246 065 \

882 694 778 879 999 252 588 323 311 654 089 988 267 564 319 334 837 124 267 695 878 381 841 100 \

323 828 744 316 405 209 602 479 432 156 274 361 102 583 359 111 483 990 492 777;

**Вариант 11.** $n_1 = 51\ 360\ 046\ 065\ 605\ 614\ 993;$  $n_2 =$ 

4 289 596 073 614 462 528 939 400 260 807 756 152 293 690 302 174 749 665 942 927 347 373 812 156 \

376 366 101 586 070 157 095 788 124 889 034 654 062 700 148 126 761 447 663 606 728 794 348 123 \

599 327 848 015 076 198 785 983 094 296 676 882 078 834 302 448 228 312 296 547;

**Вариант 12.** $n_1 = 65\ 309\ 880\ 565\ 151\ 380\ 421;$  $n_2 =$ 

1 547 231 468 862 916 054 982 403 834 772 959 218 887 879 739 421 200 986 075 613 091 990 934 504 \

896 760 789 734 724 058 142 635 169 415 057 517 173 161 958 128 528 153 890 038 977 446 320 660 \

013 200 168 479 581 018 694 947 749 263 661 147 360 435 549 764 084 083 539 163;

**Вариант 13.** $n_1 = 70\ 406\ 137\ 085\ 501\ 736\ 787;$  $n_2 =$ 

148 589 439 566 935 036 902 277 716 633 273 286 191 280 191 661 036 281 157 897 010 356 149 210 393 \

167 704 097 004 228 701 743 268 942 309 963 380 969 983 789 872 618 911 816 251 734 948 907 962 \

274 966 766 056 834 451 584 040 352 281 471 955 280 774 724 455 724 489 723;

**Вариант 14.** $n_1 = 76\ 595\ 422\ 539\ 295\ 090\ 891;$  $n_2 =$ 

7 092 837 507 204 389 865 199 984 093 265 881 611 024 839 281 933 861 153 632 651 434 328 708 556 \

276 047 982 816 383 554 312 817 342 113 183 938 490 281 168 616 863 581 512 459 404 541 222 851 \

716 676 375 731 348 587 697 968 637 918 432 687 640 118 167 565 586 895 198 207;

## Вариант 15.

$n_1 = 9\,548\,476\,718\,187\,751\,919;$

$n_2 =$

37 408 094 210 202 514 872 371 878 294 189 431 387 135 568 483 291 410 541 934 857 482 715 470 008 \;  
908 520 156 237 598 797 424 108 316 306 810 309 412 813 622 590 662 174 943 500 638 192 353 764 \;  
001 843 771 025 800 231 040 956 247 654 876 045 199 975 832 171 811 153 214 253;

В Алгоритме **Черепаша и Заяц с хранением последовательности** строится достаточно длинная последовательность  $x_0, x_1, \dots, x_L$ , для ее хранения необходимо использовать большой объем памяти, этого можно избежать. Для этого мы будем хранить лишь текущее положение Черепаша и Заяца.

Алгоритм. **Черепаша и Заяц.**

Составное число  $n \in \mathbb{N}$ ,  $a, x_0 \in \mathbb{Z}$ .

Нетривиальный делитель числа  $n$ .

1. Задаем начальные данные:

Hare :=  $x_0$ , Tortoise :=  $x_0$ ,  $f(x) := x^2 + a \pmod{n}$ .

2. Двигаем фишки

Hare :=  $f(f(\text{Hare}))$ , Tortoise :=  $f(\text{Tortoise})$ .

3. Вычисляем  $d := \text{GCD}(\text{Hare} - \text{Tortoise}, n)$ .

4. Если  $1 < d < n$ , то выдаем результат  $d$ , конец алгоритма.

5. Если  $d = n$ , то выдаем результат “делитель не найден, возьмите другую последовательность”, конец алгоритма.

6. Переходим к шагу 2.

## Задание 5.

Реализовать Алгоритм **Черепаша и Заяц**. Сравнить его с Алгоритмом

**Черепаша и Заяц с хранением последовательности:**

Какой из алгоритмов работает быстрее?

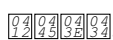
В каком из алгоритмов выполняется больше арифметических операций?

Итак, Черепаша передвигается по каждому элементу  $x_i$ , Заяц прыгает через элемент и передвигается лишь по четным элементам  $x_{2i}$ . Заменим Заяца на Ахиллеса (Achilles), который будет прыгать по элементам

$x_0, x_1, x_2, x_4, x_8, \dots, x_{2^h}$ , то есть длина каждого следующего прыжка равна всему пройденному расстоянию.

Согласно Зенону Элейскому Ахиллес никогда не догонит черепаху.

### Алгоритм. Черепаха и Ахиллес.

 Составное число  $n \in \mathbb{N}$ ,  $a, x_0 \in \mathbb{Z}$ .

 Нетривиальный делитель числа  $n$ .

1. Задаем начальные данные:

$$k := 0, \text{Hare} := x_0, \text{Achilles} := x_0, f(x) := x^2 + a \pmod{n}.$$

2. Для  $j = 1, 2, 3, \dots, 2^k$  выполняем операции 2.1-2.4.

2.1. Двигаем Черепаху

$$\text{Tortoise} := f(\text{Tortoise}).$$

2.2. Вычисляем  $d := \text{GCD}(\text{Achilles} - \text{Tortoise}, n)$ .

2.3. Если  $1 < d < n$ , то выдаем результат  $d$ , конец алгоритма.

2.4. Если  $d = n$ , то выдаем результат “делитель не найден, возьмите другую последовательность”, конец алгоритма.

3. Двигаем Ахиллеса

$$\text{Achilles} := \text{Tortoise}, k := k + 1.$$

4. Переходим к шагу 2.

### Задание 6.

Реализовать алгоритм **Черепаха и Ахиллес**. Сравнить его с алгоритмом Черепаха и Заяц.

### Задание 7.

Пусть  $f(x) = x^2 + 1, x_0 = 1$ . Для чисел  $n = n_1$  и  $n = n_2$  выяснить, существуют ли такие  $j$  и  $k$ , что  $1 < \text{gcd}(x_j - x_k, n) < n$ .

#### Вариант 1.

$$n_1 = 41\,053;$$

$$n_2 = 52\,357;$$

#### Вариант 2.

$$n_1 = 45\,511;$$

$$n_2 = 68\,143;$$



### Вариант 3.

$n_1 = 51\,353;$   
 $n_2 = 94\,657;$

### Вариант 4.

$n_1 = 57\,401;$   
 $n_2 = 98\,149;$

### Вариант 5.

$n_1 = 59\,431;$   
 $n_2 = 105\,641;$

### Вариант 6.

$n_1 = 63\,101;$   
 $n_2 = 171\,109;$

### Вариант 7.

$n_1 = 65\,771;$   
 $n_2 = 295\,913;$

### Вариант 8.

$n_1 = 65\,953;$   
 $n_2 = 298\,139;$

### Вариант 9.

$n_1 = 73\,027;$   
 $n_2 = 324\,721;$

### Вариант 10.

$n_1 = 74\,621;$   
 $n_2 = 436\,921;$

### Вариант 11.

$n_1 = 75\,917;$   
 $n_2 = 458\,329;$

### Вариант 12.

$n_1 = 76\,117;$   
 $n_2 = 492\,181;$

### Вариант 13.

$n_1 = 81\,791$ ;  
 $n_2 = 502\,681$ ;

### Вариант 14.

$n_1 = 84\,739$ ;  
 $n_2 = 579\,121$ ;

### Вариант 15.

$n_1 = 85\,973$ ;  
 $n_2 = 644\,659$ ;

### Задание 8.

Выбрать из реализованных выше алгоритмов наиболее эффективный.

## ( $p - 1$ ) алгоритм Полларда

### $(p - 1)$ -алгоритм Полларда

- Пусть  $p$  — нетрив. простой делитель  $n$ .
- Известно разложение

$$p - 1 = q_1^{\alpha_1} q_2^{\alpha_2} \dots q_s^{\alpha_s}.$$

- Малая теорема Ферма.  $a \in \mathbb{N}$ ,  $\gcd(a, p) = 1$ ,

$$a^{p-1} \equiv 1 \pmod{p}.$$

- Тогда

$$d := \gcd(a^{p-1} - 1, n) \geq p.$$

Если  $d < n$ , то мы имеем нетривиальный делитель  $d$  числа  $n$ .

- Но нам неизвестны ни число  $p$ , ни его делители  $q_i$ .
- Предположим, что нам известны такие числа  $M, K \in \mathbb{N}$ , что

$$p < M, \quad q_i < K$$

для всех  $i = 1, 2, \dots, s$ .

$\ln[\cdot] :=$

- Понятно, что, например, при  $M = K = n$  неравенства удовлетворяются, но с вычислительной точки зрения нам такие большие числа  $M$  и  $K$  не подойдут.
- Через  $B$  обозначим все простые числа, меньшие  $K$ ,

$$B := \{2, 3, 5, 7, \dots, p_i, \dots, p_m\},$$

$p_i$  — простые,  $p_i < K$ .

- Так как  $q_i < K$ , то число  $p - 1$  имеет разложение

$$p - 1 = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_m^{\alpha_m},$$

где  $\alpha_i \geq 0$ , но мы не знаем  $\alpha_i$ . Так как  $p < M$ , то

$$\alpha_i < \frac{\ln M}{\ln p_i}.$$

- Обозначим  $\beta_i := \left\lfloor \frac{\ln M}{\ln p_i} \right\rfloor$ .

- Получили число

$$P := p_1^{\beta_1} p_2^{\beta_2} \dots p_m^{\beta_m}.$$

- Так как  $\beta_i \geq \alpha_i$ , то  $P$  делится на  $p - 1$ , поэтому

$$a^P \equiv 1 \pmod{p}.$$

Следовательно,  $\gcd(a^P - 1, n) \geq p$ .

## $(p - 1)$ -алгоритм Полларда

- **Вход.** Составное число  $n$ .  $M, K \in \mathbb{N}$ .
  - **Выход.** Нетривиальный делитель  $p$ .
1. Находим все простые делители, меньшие  $K$ ,  
 $B := \{2, 3, 5, 7, \dots, p_i, \dots, p_m\}$ .
  2. Выбираем случайное  $a \in \mathbb{N}$ ,  $1 < a < n$ , если  
 $d := \gcd(a, n) > 1$ , то  $d$  — делитель, конец алгоритма.
  3. Для  $i = 1, 2, \dots, m$  выполняем операции 3.1, 3.2.
    - 3.1. Вычисляем  $\beta_i := \left\lfloor \frac{\ln M}{\ln p_i} \right\rfloor$ .
    - 3.2. Вычисляем  $a := a^{p_i^{\beta_i}} \pmod{n}$ .
  4. Находим  $d := \gcd(a - 1, n)$ .
  5. Если  $d = 1$ , то выдаем результат «делитель не найден, возьмите  $M, K$  большими», конец алгоритма.
  6. Если  $1 < d < n$ , то выдаем рез.  $d$ , конец алг.
  7. Если  $d = n$ , то выдаем рез. «делитель не найден, возьмите  $M, K$  меньшими или другое  $a$ », конец алг.

### Задание 9.

Реализовать  $(p-1)$  алгоритм Полларда.

### Задание 10.

Разработать и реализовать алгоритм, который генерирует случайное простое число  $p$  длины  $L$  бит. При этом разложение на простые множители  $p - 1 = q_1^{\alpha_1} q_2^{\alpha_2} \dots q_s^{\alpha_s}$  содержит простые числа  $q_i$ , длина которых не превосходит  $N$  бит.