



УНИВЕРСИТЕТ ИТМО

ФГАОУ ВО «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

Лабораторная работа №6

Вариант 7

Лабушев Тимофей

Группа Р3402

Санкт-Петербург

2021

Лабораторная работа №6.

Расшифрование криптограммы на основе эллиптических кривых

Цель работы

Зная секретный ключ n_b и шифртекст, получить исходное сообщение, используя приведенный алфавит на основе кривой $E_{751}(-1, 1) : y^2 = x^3 - 1x + 1 \pmod{751}$ и генерирующей точки $G = (0, 1)$.

Задание

Вариант 7:

- Зашифрованный текст: $((16, 416), (128, 672)), ((56, 419), (59, 386)), ((425, 663), (106, 24)), ((568, 355), (145, 608)), ((188, 93), (279, 398)), ((425, 663), (99, 295)), ((179, 275), (269, 187)), ((188, 93), (395, 337)), ((188, 93), (311, 68)), ((135, 82), (556, 484)), ((56, 419), (106, 727)), ((16, 416), (307, 693))$
- Секретный ключ n_b : 12

Эллиптические криптосистемы

Криптостойкость алгоритма RSA, рассмотренного в предыдущих лабораторных работах, основывается на сложности факторизации произведения двух больших простых чисел ($n = pq$). В основе эллиптических криптосистем лежит сложность определения k при знании точки G , принадлежащей кривой $E_p(a, b)$, и произведения kG .

Секретный ключ n_b выбирается случайным образом. Публичный ключ P_b вычисляется как $P_b = n_b G$.

Для шифрования выбирается случайное число k и рассчитывается точка kG . К исходному сообщению P_m прибавляется произведение публичного ключа и выбранного числа, kP_b . Зашифрованное сообщение состоит из пары точек $(kG, P_m + kP_b)$.

Для дешифрования из второй точки необходимо вычесть kP_b , что при наличии секретного ключа n_b сводится к вычитанию произведения n_b и kG :

$$(P_m + kP_b) - n_b(kG) = (P_m + k(n_b G)) - n_b(kG) = P_m + kn_b G - kn_b G = P_m.$$

Чтобы получить исходное сообщения без знания секретного ключа, атакующему придется вычислить k , что считается вычислительно сложной задачей.

```
. begin
.   # Describes a point on a curve
.   const Point = Tuple{Int64, Int64}
.
.   struct Curve
.       # Describes an elliptic curve of form y^2 = x^3 + Ax + B (mod p)
.       a::Int64
.       b::Int64
.       p::Int64
.   end
.
.   task_G = (0, 1)
.   task_nb = 12
.   task_ciphertext = [
.       ((16,416),(128,672)), ((56,419),(59,386)), ((425,663),(106,24)),
.       ((568,355),(145,608)), ((188,93),(279,398)), ((425,663),(99,295)),
.       ((179,275),(269,187)), ((188,93),(395,337)), ((188,93),(311,68)),
.       ((135,82),(556,484)), ((56,419),(106,727)), ((16,416),(307,693))
.   ]
.
.   E = Curve(-1, 1, 751) # y^2 = x^3 - x + 1 \pmod{751}
. end
```

Given private key nb = 12, G = (0, 1)

```
Decrypted (kG, Pm_kPb) = ((16, 416), (128, 672))
-nb*kG = (519, 38), Pm = Pm_kPb + (-nb*kG) = (236, 39) = и
Decrypted (kG, Pm_kPb) = ((56, 419), (59, 386))
-nb*kG = (499, 595), Pm = Pm_kPb + (-nb*kG) = (235, 732) = з
Decrypted (kG, Pm_kPb) = ((425, 663), (106, 24))
-nb*kG = (750, 750), Pm = Pm_kPb + (-nb*kG) = (229, 600) = г
Decrypted (kG, Pm_kPb) = ((568, 355), (145, 608))
-nb*kG = (406, 397), Pm = Pm_kPb + (-nb*kG) = (240, 309) = о
Decrypted (kG, Pm_kPb) = ((188, 93), (279, 398))
-nb*kG = (327, 643), Pm = Pm_kPb + (-nb*kG) = (247, 266) = т
Decrypted (kG, Pm_kPb) = ((425, 663), (99, 295))
-nb*kG = (750, 750), Pm = Pm_kPb + (-nb*kG) = (240, 309) = о
Decrypted (kG, Pm_kPb) = ((179, 275), (269, 187))
-nb*kG = (116, 659), Pm = Pm_kPb + (-nb*kG) = (229, 151) = в
Decrypted (kG, Pm_kPb) = ((188, 93), (395, 337))
-nb*kG = (327, 643), Pm = Pm_kPb + (-nb*kG) = (237, 454) = л
Decrypted (kG, Pm_kPb) = ((188, 93), (311, 68))
-nb*kG = (327, 643), Pm = Pm_kPb + (-nb*kG) = (234, 587) = е
Decrypted (kG, Pm_kPb) = ((135, 82), (556, 484))
-nb*kG = (135, 82), Pm = Pm_kPb + (-nb*kG) = (238, 576) = н
Decrypted (kG, Pm_kPb) = ((56, 419), (106, 727))
-nb*kG = (499, 595), Pm = Pm_kPb + (-nb*kG) = (236, 39) = и
Decrypted (kG, Pm_kPb) = ((16, 416), (307, 693))
-nb*kG = (519, 38), Pm = Pm_kPb + (-nb*kG) = (234, 587) = е
```

Message: изготовление

```
• with_terminal() do
•   println("Given private key nb = $(task_nb), G = $(task_G)\n")
•
•   msg = []
•   for (kG, Pm_kPb) in task_ciphertext
•       nbkG = elliptic_mul(E, kG, -task_nb)
•       Pm = elliptic_add(E, Pm_kPb, nbkG)
•       c = curve_lookup[Pm]
•
•       println("Decrypting (kG, Pm_kPb) = ($(kG), $(Pm_kPb))")
•       println("-nb*kG = $(nbkG), Pm = Pm_kPb + (-nb*kG) = $(Pm) = $(c)")
•
•       append!(msg, c)
•   end
•
•   println("\nMessage: $join(msg))")
• end
```

elliptic_add (generic function with 1 method)

```
• # Computes P1 + P2, defined as the point at the intersection
• # of the curve E and the straight line defined by P1 and P2
• function elliptic_add(E::Curve, P1::Point, P2::Point)::Point
•     x1, y1 = P1
•     x2, y2 = P2
•
•     # Find the slope of the line defined by P1 and P2
•     slope::Int64 = if P1 != P2
•         # If P1 is not equal to P2, the slope is (y2-y1)/(x2-x1).
•         # In modulo arithmetic, division is defined as multiplication
•         # by the multiplicative inverse - such b that a*b (mod p) = 1
•         (y2 - y1) * invmod(x2 - x1, E.p)
•     else
•         # If P1 == P2, we take the tangent line to E at point P1
•         # (3x1^2 + A)/(2y1), division is again replaced with multiplication
•         (3*x1^2 + E.a) * invmod(2y1, E.p)
•     end
•
•     # Find the point at the intersection of E and the line
•     x3 = mod(slope^2 - x1 - x2, E.p)
•     y3 = mod(slope*(x1 - x3) - y1, E.p)
•
•     (x3, y3)
• end
```

elliptic_mul (generic function with 1 method)

```
• # Computes kP via repeated addition of P + P. If k is negative, P is negated.
• function elliptic_mul(E::Curve, P::Point, k::Int)::Point
•     if k < 0
•         k = -k
•         # To negate a point, we keep the x coordinate and negate the y coordinate.
•         # In modulo arithmetic, -a (mod p) is equivalent to -a + p.
•         P = (P[1], -P[2] + E.p)
•     end
•
•     R = P
•
•     for i in 1:k-1
•         R = elliptic_add(E, R, P)
•     end
•
•     R
• end
```

```

• begin
•   # A mapping of points on the curve to plaintext characters
•   curve_syntable = ""
• 35 B (67, 84) 70 e (99, 456) 105 й (198, 527)
• 1 (33, 355) 36 C (67, 667) 71 f (100, 364) 106 K (200, 30)
• 2 ! (33, 396) 37 D (69, 241) 72 g (100, 387) 107 Л (200, 721)
• 3 " (34, 74) 38 E (69, 510) 73 h (102, 267) 108 M (203, 324)
• 4 # (34, 677) 39 F (70, 195) 74 i (102, 484) 109 H (203, 427)
• 5 \$ (36, 87) 40 G (70, 556) 75 j (105, 369) 110 O (205, 372)
• 6 % (36, 664) 41 H (72, 254) 76 k (105, 382) 111 П (205, 379)
• 7 & (39, 171) 42 I (72, 497) 77 l (106, 24) 112 P (206, 106)
• 8 ' (39, 580) 43 J (73, 72) 78 m (106, 727) 113 C (206, 645)
• 9 ( (43, 224) 44 K (73, 679) 79 n (108, 247) 114 T (209, 82)
• 10 ) (43, 527) 45 L (74, 170) 80 o (108, 504) 115 Y (209, 669)
• 11 * (44, 366) 46 M (74, 581) 81 p (109, 200) 116 Ф (210, 31)
• 12 + (44, 385) 47 N (75, 318) 82 q (109, 551) 117 X (210, 720)
• 13 , (45, 31) 48 O (75, 433) 83 r (110, 129) 118 Ц (215, 247)
• 14 - (45, 720) 49 P (78, 271) 84 s (110, 622) 119 Ч (215, 504)
• 15 . (47, 349) 50 Q (78, 480) 85 t (114, 144) 120 Ш (218, 150)
• 16 / (47, 402) 51 R (79, 111) 86 u (114, 607) 121 Щ (218, 601)
• 17 0 (48, 49) 52 S (79, 640) 87 v (115, 242) 122 Ъ (221, 138)
• 18 1 (48, 702) 53 T (80, 318) 88 w (115, 509) 123 Ы (221, 613)
• 19 2 (49, 183) 54 U (80, 433) 89 x (116, 92) 124 Ь (226, 9)
• 20 3 (49, 568) 55 V (82, 270) 90 y (116, 659) 125 Э (226, 742)
• 21 4 (53, 277) 56 W (82, 481) 91 z (120, 147) 126 Ю (227, 299)
• 22 5 (53, 474) 57 X (83, 373) 92 { (120, 604) 127 Я (227, 452)
• 23 6 (56, 332) 58 Y (83, 378) 93 | (125, 292) 128 а (228, 271)
• 24 7 (56, 419) 59 Z (85, 35) 94 } (125, 459) 129 б (228, 480)
• 25 8 (58, 139) 60 [ (85, 716) 95 ~ (126, 33) 130 в (229, 151)
• 26 9 (58, 612) 61 \ (86, 25) 96 A (189, 297) 131 г (229, 600)
• 27 : (59, 365) 62 ] (86, 726) 97 Б (189, 454) 132 д (234, 164)
• 28 ; (59, 386) 63 ^ (90, 21) 98 B (192, 32) 133 е (234, 587)
• 29 < (61, 129) 64 _ (90, 730) 99 Г (192, 719) 134 ж (235, 19)
• 30 = (61, 622) 65 ` (93, 267) 100 Д (194, 205) 135 з (235, 732)
• 31 > (62, 372) 66 a (93, 484) 101 E (194, 546) 136 и (236, 39)
• 32 ? (62, 379) 67 b (98, 338) 102 Ж (197, 145) 137 й (236, 712)
• 33 @ (66, 199) 68 c (98, 413) 103 З (197, 606) 138 к (237, 297)
• 34 A (66, 552) 69 d (99, 295) 104 И (198, 224) 139 л (237, 454)
• 140 м (238, 175) 145 c (243, 664) 150 ц (250, 14) 155 ы (253, 540)
• 141 н (238, 576) 146 т (247, 266) 151 ч (250, 737) 156 ь (256, 121)
• 142 o (240, 309) 147 y (247, 485) 152 ш (251, 245) 157 э (256, 630)
• 143 п (240, 442) 148 ф (249, 183) 153 щ (251, 506) 158 ю (257, 293)
• 144 p (243, 87) 149 x (249, 568) 154 ъ (253, 211) 159 я (257, 458) ""
•   curve_syntable = split(curve_syntable, c -> c in "\n\t")
•   curve_syntable = reshape(curve_syntable, 3, :)
•   curve_lookup = Dict{Point, Char}()
•   (map(s -> parse{Int, s}, split(r[3][2:end-1], ", "))...,) =>
•     r[2][1]
•     for r in eachcol(curve_syntable)
•   )
• end

```

Вывод

В ходе выполнения лабораторной работы был изучен общий принцип работы эллиптических криптосистем и реализован алгоритм расшифрования криптограмм.