Криптографија
- други део -

доц. др Драган Ђокић

Математички факултет, Универзитет у Београду

dragan.djokic@matf.bg.ac.rs

18. - 21. фебруар 2025.

Дифи-Хелманова размена (усаглашавање) кључа

- користи коначно поље $\mathbb{F}_q$ и један елемент $g \in \mathbb{F}_q$
  - Најбоље је да g буде генератор мултипликативне групе $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$, а прихватљиво је и да буде елемент великог реда

Дифи-Хелманова размена (усаглашавање) кључа

- користи коначно поље $\mathbb{F}_q$ и један елемент $g \in \mathbb{F}_q$
  - Најбоље је да g буде генератор мултипликативне групе $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$, а прихватљиво је и да буде елемент великог реда
- Дифи-Хелманова размена кључа се заснива на следећем:
  - ♣ ако знамо $g \in \mathbb{F}_q^*$ и $n \in \mathbb{N}$ лако је одредити g^n (степен дефинисан преко множења по модулу p и $f(x)$)
  - ♠ али ако знамо g и g^n тешко је одредити n

Алгоритам:

- Алиса и Бобан бирају степен простог броја $q = p^d$ (приближно 200-цифрен) и генератор $g \in \mathbb{F}_q^*$ и објављују q и g

Алгоритам:

- Алиса и Бобан бирају степен простог броја $q = p^d$ (приближно 200-цифрен) и генератор $g \in \mathbb{F}_q^*$ и објављују q и g
- Алиса бира свој тајни кључ $a_A \in \mathbb{N}$, рачуна и објављује само g^{a_A} (јавни кључ)

Алгоритам:

- Алиса и Бобан бирају степен простог броја $q = p^d$ (приближно 200-цифрен) и генератор $g \in \mathbb{F}_q^*$ и објављују q и g
- Алиса бира свој тајни кључ $a_A \in \mathbb{N}$, рачуна и објављује само g^{a_A} (јавни кључ)
- Слично, Бобан бира тајни кључ $a_B \in \mathbb{N}$, рачуна и објављује само g^{a_B} (јавни кључ)

Алгоритам:

- Алиса и Бобан бирају степен простог броја $q = p^d$ (приближно 200-цифрен) и генератор $g \in \mathbb{F}_q^*$ и објављују q и g
- Алиса бира свој тајни кључ $a_A \in \mathbb{N}$, рачуна и објављује само g^{a_A} (јавни кључ)
- Слично, Бобан бира тајни кључ $a_B \in \mathbb{N}$, рачуна и објављује само g^{a_B} (јавни кључ)
- И Алиса и Бобан могу израчунати $K = (g^{a_A})^{a_B} = (g^{a_B})^{a_A}$, и то ће бити њихов усаглашен кључ

Алгоритам:

- Алиса и Бобан бирају степен простог броја $q = p^d$ (приближно 200-цифрен) и генератор $g \in \mathbb{F}_q^*$ и објављују q и g
- Алиса бира свој тајни кључ $a_A \in \mathbb{N}$, рачуна и објављује само g^{a_A} (јавни кључ)
- Слично, Бобан бира тајни кључ $a_B \in \mathbb{N}$, рачуна и објављује само g^{a_B} (јавни кључ)
- И Алиса и Бобан могу израчунати $K = (g^{a_A})^{a_B} = (g^{a_B})^{a_A}$, и то ће бити њихов усаглашен кључ
- Цица зна само q, g g^{a_A} и g^{a_B}, и помоћу тога не може (довољно брзо) да одреди K

Степеновање поновљеним квадрирањем

Брзи алгоритам за ♣ тј. рачунање g^n у $\mathbb{F}_q^*$:

1. Редуковати степен на $n < q - 1$ због $g^{q-1} = 1$ (аналог Мале Фермаове теореме у $\mathbb{F}_q$)

Степеновање поновљеним квадрирањем

Брзи алгоритам за ♣ тј. рачунање g^n у $\mathbb{F}_q^*$:

1. Редуковати степен на $n < q-1$ због $g^{q-1} = 1$ (аналог Мале Фермаове теореме у $\mathbb{F}_q$)
2. Записати n бинарно $n = \overline{n_r n_{r-1} \dots n_1 n_0} = \sum_{i=0}^{r} n_i \cdot 2^i$, $n_i \in \{0, 1\}$

Степеновање поновљеним квадрирањем

Брзи алгоритам за ♣ тј. рачунање g^n у $\mathbb{F}_q^*$:

1. Редуковати степен на $n < q-1$ због $g^{q-1} = 1$ (аналог Мале Фермаове теореме у $\mathbb{F}_q$)
2. Записати n бинарно $n = \overline{n_r n_{r-1} \dots n_1 n_0} = \sum_{i=0}^{r} n_i \cdot 2^i$, $n_i \in \{0, 1\}$
3. Израчунати 1, g, g^2, $\left(g^2\right)^2 = g^{2^2}$, $\left(g^{2^2}\right)^2 = g^{2^3}$, $\left(g^{2^3}\right)^2 = g^{2^4}$, $\dots$, $\left(g^{2^{r-1}}\right)^2 = g^{2^r}$ (сваки је квадрат претходног)

Степеновање поновљеним квадрирањем

Брзи алгоритам за ♣ тј. рачунање g^n у $\mathbb{F}_q^*$:

1. Редуковати степен на $n < q-1$ због $g^{q-1} = 1$ (аналог Мале Фермаове теореме у $\mathbb{F}_q$)
2. Записати n бинарно $n = \overline{n_r n_{r-1} \dots n_1 n_0} = \sum_{i=0}^{r} n_i \cdot 2^i$, $n_i \in \{0, 1\}$
3. Израчунати 1, g, g^2, $\left(g^2\right)^2 = g^{2^2}$, $\left(g^{2^2}\right)^2 = g^{2^3}$, $\left(g^{2^3}\right)^2 = g^{2^4}$, $\dots$, $\left(g^{2^{r-1}}\right)^2 = g^{2^r}$ (сваки је квадрат претходног)
4. g^n је производ оних g^{2^i} за које је $n_i = 1$

Степеновање поновљеним квадрирањем

Брзи алгоритам за ♣ тј. рачунање g^n у $\mathbb{F}_q^*$:

1. Редуковати степен на $n < q-1$ због $g^{q-1} = 1$ (аналог Мале Фермаове теореме у $\mathbb{F}_q$)
2. Записати n бинарно $n = \overline{n_r n_{r-1} \ldots n_1 n_0} = \sum_{i=0}^{r} n_i \cdot 2^i$, $n_i \in \{0, 1\}$
3. Израчунати 1, g, g^2, $\left(g^2\right)^2 = g^{2^2}$, $\left(g^{2^2}\right)^2 = g^{2^3}$, $\left(g^{2^3}\right)^2 = g^{2^4}$, $\ldots$, $\left(g^{2^{r-1}}\right)^2 = g^{2^r}$ (сваки је квадрат претходног)
4. g^n је производ оних g^{2^i} за које је $n_i = 1$

Доказ:

$$g^n = g^{\sum_{i=0}^{r} n_i \cdot 2^i} = \prod_{i=0}^{r} g^{n_i \cdot 2^i} = \prod_{\substack{0 \leqslant i \leqslant r \\ n_i = 1}} g^{2^i}$$

Временска сложеност

у случају $q = p$ прост:

- за множење k-битног броја a и l-битног броја b треба kl операција, где је $k \sim \log a$ и $l \sim \log b$
- исто за целобројно дељење и узимање остатка по модулу

Временска сложеност

у случају $q = p$ прост:

- за множење k-битног броја a и l-битног броја b треба kl операција, где је $k \sim \log a$ и $l \sim \log b$
- исто за целобројно дељење и узимање остатка по модулу
- У нашем алгоритму:
  1. $O\left(\log^2 p\right)$ операција јер је $\log p \sim$ број битова броја p
  2. $O(r \log n) = O\left(\log^2 p\right)$ операција - јер r пута понављамо дељење са 2 и остатак по модулу 2 (а $r \sim \log n \leqslant \log p$)
  3. r квадрирања, а за свако квадрирање треба по $O\left(\log^2\left(g^{2^i}\right)\right) = O\left(\log^2 p\right)$ операција
  4. највише r множења која захтевају по највише $O\left(\log^2 p\right)$ операција
- укупно $O\left(r \log^2 p\right) = O\left(\log^3 p\right)$

Временска сложеност

у случају $q = p$ прост:

- за множење k-битног броја a и l-битног броја b треба kl операција, где је $k \sim \log a$ и $l \sim \log b$
- исто за целобројно дељење и узимање остатка по модулу
- У нашем алгоритму:
  1. $O\left(\log^2 p\right)$ операција јер је $\log p \sim$ број битова броја p
  2. $O(r \log n) = O\left(\log^2 p\right)$ операција - јер r пута понављамо дељење са 2 и остатак по модулу 2 (а $r \sim \log n \leqslant \log p$)
  3. r квадрирања, а за свако квадрирање треба по $O\left(\log^2\left(g^{2^i}\right)\right) = O\left(\log^2 p\right)$ операција
  4. највише r множења која захтевају по највише $O\left(\log^2 p\right)$ операција
- укупно $O\left(r \log^2 p\right) = O\left(\log^3 p\right)$
- за множење $g^n = gg \ldots g$ би требало $O\left(n \log^2 p\right)$ операција

Напомене:

- Заједно са МФТ имамо и алгоритам за рачунање $a^{-1} = a^{p-2}$ у $\mathbb{Z}_p$ временске сложености $O\left(\log^3 p\right)$

Напомене:

- Заједно са МФТ имамо и алгоритам за рачунање $a^{-1} = a^{p-2}$ у $\mathbb{Z}_p$ временске сложености $O\left(\log^3 p\right)$
- Алгоритам ради и за модул m који није прост, али може да ради спорије јер морамо прескочити 1. корак
  - уместо Мале Фермаове могли бисмо користи Ојлерову теорему, али видећемо да је израчунавање Ојлерове функције преспоро.

Напомене:

- Заједно са МФТ имамо и алгоритам за рачунање $a^{-1} = a^{p-2}$ у $\mathbb{Z}_p$ временске сложености $O\left(\log^3 p\right)$
- Алгоритам ради и за модул m који није прост, али може да ради спорије јер морамо прескочити 1. корак
  - уместо Мале Фермаове могли бисмо користи Ојлерову теорему, али видећемо да је израчунавање Ојлерове функције преспоро.
- Када је $q = p^d$:
  - множење у $\mathbb{F}_q = \mathbb{Z}_p[x] / (f\mathbb{Z}_p[x])$ је обухвата
    - множење полинома у $\mathbb{Z}_p[x]$, тј. рачунање $O(d) \cdot O(d)$ коефицијената преко множења у $\mathbb{Z}_p$ које захтева $O\left(\log^2 p\right)$, и потом сабирање. То је $O\left(d^2 \log^2 p + d^2 \log p\right) = O\left(\log^2 q\right)$ операција.
    - замена $x^{2d-2}, x^{2d-3}, \ldots, x^d$ полиномима мањег степена (тим редом), њих има $O(d)$ и ту ће „најскупља“ операција бити инверз по модулу p (јер f можда није моничан)
    - укупно $O\left(\log^2 q + d \log^3 p\right) = O\left(\log^3 q\right)$

Напомене:

- Заједно са МФТ имамо и алгоритам за рачунање $a^{-1} = a^{p-2}$ у $\mathbb{Z}_p$ временске сложености $O\left(\log^3 p\right)$
- Алгоритам ради и за модул m који није прост, али може да ради спорије јер морамо прескочити 1. корак
  - уместо Мале Фермаове могли бисмо користи Ојлерову теорему, али видећемо да је израчунавање Ојлерове функције преспоро.
- Када је $q = p^d$:
  - множење у $\mathbb{F}_q = \mathbb{Z}_p[x]/(f\mathbb{Z}_p[x])$ је обухвата
    - множење полинома у $\mathbb{Z}_p[x]$, тј. рачунање $O(d) \cdot O(d)$ коефицијената преко множења у $\mathbb{Z}_p$ које захтева $O\left(\log^2 p\right)$, и потом сабирање. То је $O\left(d^2 \log^2 p + d^2 \log p\right) = O\left(\log^2 q\right)$ операција.
    - замена $x^{2d-2}, x^{2d-3}, \ldots, x^d$ полиномима мањег степена (тим редом), њих има $O(d)$ и ту ће „најскупља“ операција бити инверз по модулу p (јер f можда није моничан)
    - укупно $O\left(\log^2 q + d\log^3 p\right) = O\left(\log^3 q\right)$
  - последица: степеновање поновљеним квадрирањем има сложеност $O\left(\log^4 q\right)$

Пример: $57^{1616} \pmod{97}$

Broj 97 je prost i $NZD(57, 97) = 1$ pa možemo primeniti Malu Fermaovu teoremu. Pored toga, iskoristićemo činjenicu da je $80 = 64 + 16$.

$$57^{1616}(mod\ 97) = 57^{96 \cdot 16} \cdot 57^{80}(mod\ 97) = 57^{80}(mod\ 97) = 57^{64} \cdot 57^{16}(mod\ 97)$$

Odredimo vrednosti 57^{2^i}, za $i \in [1, 6]$ po modulu 97:

$$57^2 \equiv 48 \qquad 57^{16} \equiv 91^2 \equiv (-6)^2 \equiv 36$$
$$57^4 \equiv 48^2 \equiv 73 \qquad 57^{32} \equiv 36^2 \equiv 35$$
$$57^8 \equiv 73^2 \equiv 91 \qquad 57^{64} \equiv 35^2 \equiv 61$$

Sada lako računamo vrednost izraza:

$$57^{1616}(mod\ 97) = 57^{64} \cdot 57^{16}(mod\ 97) = 61 \cdot 36(mod\ 97) \equiv 62(mod\ 97)$$

Пример: $43^{257} \pmod{59}$

Broj 59 je prost i $NZD(59, 43) = 1$ pa možemo primeniti Malu Fermaovu teoremu:

$$43^{257}(mod\ 59) = 43^{25}(mod\ 59) = 43^{16} \cdot 43^{8} \cdot 43(mod\ 59)$$

Odredimo vrednosti 43^{2^i}, za $i \in [1, 4]$ po modulu 59:

$$43^2 \equiv 20 \qquad 43^8 \equiv 46^2 \equiv 51$$
$$43^4 \equiv 20^2 \equiv 46 \qquad 43^{16} \equiv 51^2 \equiv 5$$

Sada lako računamo vrednost početnog izraza:

$$43^{253}(mod\ 59) = 43^{16} \cdot 43^{8} \cdot 43(mod\ 59) = 5 \cdot 51 \cdot 43(mod\ 59) = 50$$

Дифи-Хелманово усаглашавање кључа у криптосистему са јавним кључем $p = 97,\ g = 5$

Алиса бира тајни кључ $a_A = 36$ и рачуна јавни кључ

$$
\begin{aligned}
g^{a_A} &= 5^{36}(mod\ 97) = 5^{32} \cdot 5^4 (mod\ 97) \\
& 5^4 \equiv 25^2 \equiv 43 \\
& 5^{32} \equiv (5^4)^8 \equiv 43^8 \equiv 6^4 \equiv 35 \\
g^{a_A} &= 43 \cdot 35 (mod\ 97) = 50 (mod\ 97)
\end{aligned}
$$

Дифи-Хелманово усаглашавање кључа у криптосистему са јавним кључем $p = 97,\ g = 5$

Алиса бира тајни кључ $a_A = 36$ и рачуна јавни кључ

$$\begin{aligned} g^{a_A} &= 5^{36}(mod\ 97) = 5^{32} \cdot 5^4 (mod\ 97) \\ & 5^4 \equiv 25^2 \equiv 43 \\ & 5^{32} \equiv (5^4)^8 \equiv 43^8 \equiv 6^4 \equiv 35 \\ g^{a_A} &= 43 \cdot 35 (mod\ 97) = 50 (mod\ 97) \end{aligned}$$

Бобан бира тајни кључ $a_B = 58$ и рачуна јавни кључ

$$\begin{aligned} g^{a_B} &= 5^{58}(mod\ 97) = 5^{32} \cdot 5^{16} \cdot 5^8 \cdot 5^2 (mod\ 97) \\ & 5^2 \equiv 25 \\ & 5^8 \equiv (5^2)^4 \equiv (25^2)^2 \equiv 43^2 = 6 \\ & 5^{16} \equiv (5^8)^2 \equiv 6^2 \equiv 36 \\ & 5^{32} \equiv 35 \\ g^{a_B} &= 25 \cdot 6 \cdot 36 \cdot 35 (mod\ 97) = 44 (mod\ 97) \end{aligned}$$

Алиса рачуна размењени кључ K као

$$
\begin{aligned}
K = \; & (g^{a_B})^{a_A} = 44^{36}(mod\ 97) = 44^{32} \cdot 44^{4}(mod\ 97) \\
& 44^4 \equiv (44^2)^2 \equiv 93^2 \equiv (-4)^2 \equiv 16 \\
& 44^{32} \equiv (44^4)^8 \equiv 16^8 \equiv 62^4 \equiv 61^2 \equiv 35 \\
K = \; & 35 \cdot 16(mod\ 97) = 75(mod\ 97)
\end{aligned}
$$

Алиса рачуна размењени кључ K као

$$\begin{aligned} K = & \ (g^{a_B})^{a_A} = 44^{36}(mod\ 97) = 44^{32} \cdot 44^4(mod\ 97) \\ & 44^4 \equiv (44^2)^2 \equiv 93^2 \equiv (-4)^2 \equiv 16 \\ & 44^{32} \equiv (44^4)^8 \equiv 16^8 \equiv 62^4 \equiv 61^2 \equiv 35 \\ K = & \ 35 \cdot 16(mod\ 97) = 75(mod\ 97) \end{aligned}$$

Бобан рачуна размењени кључ K као

$$\begin{aligned} K = & \ (g^{a_A})^{a_B} = 50^{58}(mod\ 97) = 50^{32} \cdot 50^{16} \cdot 50^8 \cdot 50^2(mod\ 97) \\ & 50^2 \equiv 75 \\ & 50^8 \equiv (75)^4 \equiv (96)^2 \equiv 1 \\ & 50^{16} \equiv (50^8)^2 \equiv 1 \\ & 50^{32} \equiv (50^{16})^2 \equiv 1 \\ K = & \ 75 \cdot 1 \cdot 1 \cdot 1(mod\ 97) = 75(mod\ 97) \end{aligned}$$

- Кад кажемо бира број мислимо користи генератор случајних бројева

- Кад кажемо бира број мислимо користи генератор случајних бројева
- Али: Како се генерише случајан прост број?

- Кад кажемо бира број мислимо користи генератор случајних бројева
- Али: Како се генерише случајан прост број?
- Слабост Дифи-Хелмана:
  - Алиса и Бобан немају други канал комуникације и биће проблем ако се укључи између њих.
  - Ако Цица превари Алису, представи се као Боб, и размене кључ, онда Цица може (а Бобан не може) да чита поруке које Алиса шаље Бобану.
  - Ако се додатно Цица Бобану представи као Алиса, она може да мења Алисине поруке и прослеђује их Бобану.

Функција у phython-у која рачуна $a^n \pmod m$

```
def mod_pow(a, n, m):

    res = 1
    while n > 0:
        # Za svaku jedinicu u binarnom zapisu n
        if n % 2 == 1:
            # Rezultat je stara vrednost rezultata * a^2{i} za i-tu poziciju
            res = (res * a) % m
        a = (a * a) % m
        # n gubi poslednju cifru u binarnom zapisu
        n = n // 2
    return res
```

Напомена: подразумевамо да за улазне податке важи $0 \leqslant a < m$ и $0 \leqslant n < \varphi(m)$.

Дифи-Хелманово усаглашавање кључа

```
class Diffie_Helman:

    def __init__(self, q, g):
        self.q = q
        self.g = g
        self.pr = random.randrange(2, q)
        self.pub = mod_pow(self.g, self.pr, self.q)


    def get_key(self, pub_key):
        return mod_pow(pub_key, self.pr, self.q)
```

```
def main():
    q = get_prime(2**256)
    g = random.randrange(2, q)

    A = Diffie_Helman(q, g)
    B = Diffie_Helman(q, g)

    print(A.pr, A.pub)
    print(B.pr, B.pub)

    print(A.get_key(B.pub))
    print(B.get_key(A.pub))
```

Проблем дискретног логаритма ♠

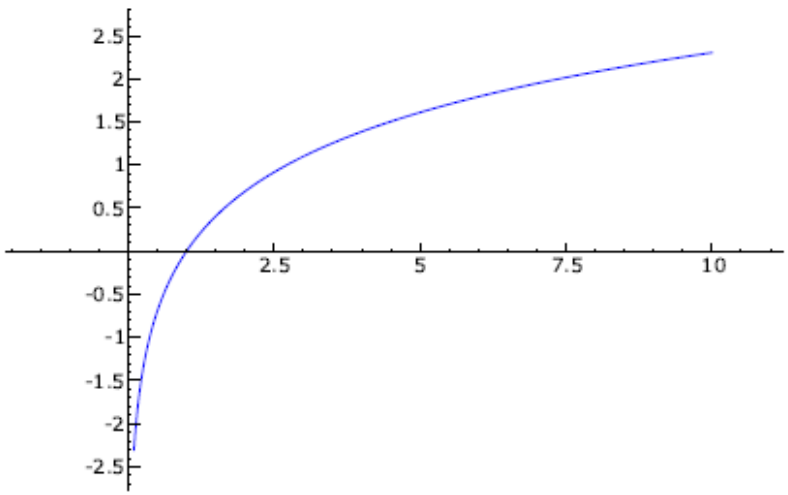
Дефиниција

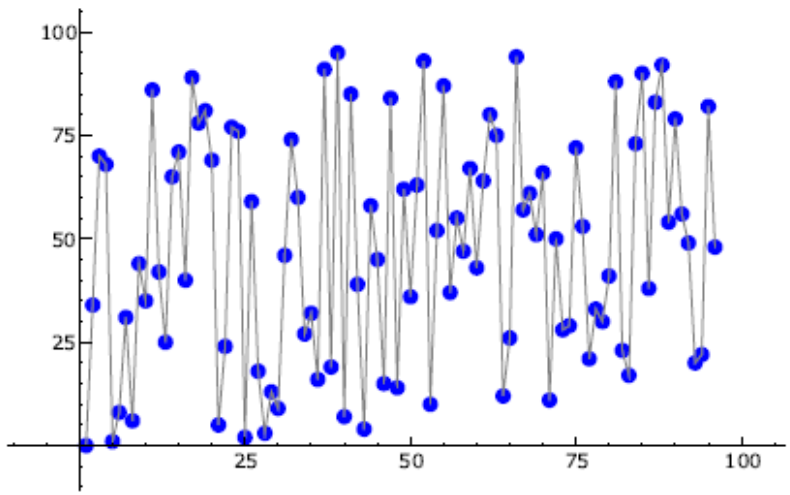
Нека је G група нпр. $\mathbb{F}_q^*$, и нека су $a, g \in G$. Најмањи природан број n (ако постоји) такав да је $a = g^n$ зовемо дискретни логаритам од a у основи g и означавамо са $\log_g a$.

Проблем дискретног логаритма ♠

Дефиниција

Нека је G група нпр. $\mathbb{F}_q^*$, и нека су $a, g \in G$. Најмањи природан број n (ако постоји) такав да је $a = g^n$ зовемо дискретни логаритам од a у основи g и означавамо са $\log_g a$.





Класичан логаритам и дискретни log_2 у групи $\mathbb{Z}_{53}^*$ (други делује непредвидиво)

График нацртан помоћу програма $SAGE$ доступан на `https://www.sagemath.org/`

```
sage: p = 53
sage: R = Integers(p)
sage: a = R.multiplicative_generator()
sage: v = sorted([(a^n, n) for n in range(p-1)])
sage: G = plot(point(v,pointsize=50,rgbcolor=(0,0,1)))
sage: H = plot(line(v,rgbcolor=(0.5,0.5,0.5)))
sage: G + H
```

График нацртан помоћу програма $SAGE$ доступан на `https://www.sagemath.org/`

```
sage: p = 53
sage: R = Integers(p)
sage: a = R.multiplicative_generator()
sage: v = sorted([(a^n, n) for n in range(p-1)])
sage: G = plot(point(v,pointsize=50,rgbcolor=(0,0,1)))
sage: H = plot(line(v,rgbcolor=(0.5,0.5,0.5)))
sage: G + H
```

- R.multiplicative_generator() враћа најмањи генератор циклične групе R (или избацује грешку ако R није циклична)

График нацртан помоћу програма $SAGE$ доступан на `https://www.sagemath.org/`

```
sage: p = 53
sage: R = Integers(p)
sage: a = R.multiplicative_generator()
sage: v = sorted([(a^n, n) for n in range(p-1)])
sage: G = plot(point(v,pointsize=50,rgbcolor=(0,0,1)))
sage: H = plot(line(v,rgbcolor=(0.5,0.5,0.5)))
sage: G + H
```

- R.multiplicative_generator() враћа најмањи генератор циклične групе R (или избацује грешку ако R није циклична)
- Тачке нису добијене као $(n, \log_a n)$, већ (a^n, n) (и потом су сортиране растуће по 1. координати)

- Немамо формулу за израчунавање $n = \log_g a$ у $\mathbb{F}_q^*$

- Немамо формулу за израчунавање $n = \log_g a$ у $\mathbb{F}_q^*$
- Пример: $\log_2 6 = ?$ у групи $\mathbb{Z}_{23}^*$
  (рачунамо 2, 2^2, 2^3, ... и чекамо да се појави 6)

n	0	1	2	3	4	5	6	7	8	9	10	11
$2^n (mod\ 23)$	1	2	4	8	16	9	18	13	3	6	12	1
n	12	13	14	15	16	17	18	19	20	21	22	
$2^n (mod\ 23)$	2	4	8	16	9	18	13	3	6	12	1	

Добићемо да је $\log_2 6 = 20$ али траје предуго

- Немамо формулу за израчунавање $n = \log_g a$ у $\mathbb{F}_q^*$
- Пример: $\log_2 6 = ?$ у групи $\mathbb{Z}_{23}^*$
  (рачунамо 2, 2^2, 2^3, ... и чекамо да се појави 6)

n	0	1	2	3	4	5	6	7	8	9	10	11
$2^n (mod\ 23)$	1	2	4	8	16	9	18	13	3	6	12	1
n	12	13	14	15	16	17	18	19	20	21	22	
$2^n (mod\ 23)$	2	4	8	16	9	18	13	3	6	12	1	

Добићемо да је $\log_2 6 = 20$ али траје предуго

- Најгори случај за $n = \log_g a$ у $\mathbb{F}_q^*$: када је g генератор n велико - практично прођемо целу горњу табелу

- Немамо формулу за израчунавање $n = \log_g a$ у $\mathbb{F}_q^*$
- Пример: $\log_2 6 = ?$ у групи $\mathbb{Z}_{23}^*$
  (рачунамо 2, 2^2, 2^3, ... и чекамо да се појави 6)

n	0	1	2	3	4	5	6	7	8	9	10	11
$2^n (mod\ 23)$	1	2	4	8	16	9	18	13	3	6	12	1
n	12	13	14	15	16	17	18	19	20	21	22	
$2^n (mod\ 23)$	2	4	8	16	9	18	13	3	6	12	1	

Добићемо да је $\log_2 6 = 20$ али траје предуго

- Најгори случај за $n = \log_g a$ у $\mathbb{F}_q^*$: када је g генератор n велико - практично прођемо целу горњу табелу
  - То је $O(q)$ тестирања! (А временска сложеност степеновања је $O\left(\log^4 q\right)$)

Време за рачунање степена и дискретног логаритма у $\mathbb{F}_q$ помоћу Sage-a (који има на располагању све познате алгоритме)

Save Copy Run SageMath 10.3

```
K = GF(39916801^4,'x')
g = K.multiplicative_generator()
%time a=pow(g, 12345678910111213141516171819 20, K.order())
%time b=discrete_log(a, g, K.order() - 1)
is_prime(39916801), g, a, b
```

```
CPU times: user 55 µs, sys: 0 ns, total: 55 µs
Wall time: 58.9 µs
CPU times: user 364 ms, sys: 3.76 ms, total: 367 ms
Wall time: 393 ms

(True,
 x + 3,
 12508294*x^3 + 18207805*x^2 + 38793833*x + 2187988,
 12345678910111213141516171819 20)
```

$GF(q,'x')$ је коначно поље са $q = p^d$ елемененената, други аргумент је симбол којим желимо да Sage означи променљиву x у $\mathbb{Z}_p[x]$

- Не постоји довољно брз алгоритам који решава проблем дискретног логаритма у $\mathbb{F}_q^*$, тј. алгоритам чија је брзина упоредива са степеновањем

- Не постоји довољно брз алгоритам који решава проблем дискретног логаритма у $\mathbb{F}_q^*$, тј. алгоритам чија је брзина упоредива са степеновањем
- Постоје алгоритми који могу да смање време претраживања $O(q)$ из претх. примера
  - Такав је алгоритам Гељфонд-Шенкса временске сложености $O\left(\sqrt{q}\log^2 q\right)$

- Не постоји довољно брз алгоритам који решава проблем дискретног логаритма у $\mathbb{F}_q^*$, тј. алгоритам чија је брзина упоредива са степеновањем
- Постоје алгоритми који могу да смање време претраживања $O(q)$ из претх. примера
  - Такав је алгоритам Гељфонд-Шенкса временске сложености $O\left(\sqrt{q}\log^2 q\right)$
- Постоје и алгоритми који раде довољно ефикасно за неке специфичне q-ове
  - Такав је Полиг-Хелманов алгоритам који предпоставља да су сви прости чиниоци броја $q-1$ „мали“
  - Да ли Дифи-Хелман може да се имплементира тако да не бира такве q-ове?

Алгоритам Гељфонд-Шенкса (baby-step-giant-step алгоритам)

- Циљ: израчунати $n = \log_g a$ у $\mathbb{F}_q^*$, где су a и g познати

Алгоритам Гељфонд-Шенкса (baby-step-giant-step алгоритам)

- Циљ: израчунати $n = \log_g a$ у $\mathbb{F}_q^*$, где су a и g познати
- Запишемо $n = mi + j$, где је $m = [\sqrt{q}]$ (цео део), $0 \leqslant i \leqslant m$, $0 \leqslant j \leqslant m - 1$

Алгоритам Гељфонд-Шенкса (baby-step-giant-step алгоритам)

- Циљ: израчунати $n = \log_g a$ у $\mathbb{F}_q^*$, где су a и g познати
- Запишемо $n = mi + j$, где је $m = [\sqrt{q}]$ (цео део), $0 \leqslant i \leqslant m$, $0 \leqslant j \leqslant m - 1$
- Тада је $g^j = a\,(g^{-m})^i$

Алгоритам Гељфонд-Шенкса (baby-step-giant-step алгоритам)

- Циљ: израчунати $n = \log_g a$ у $\mathbb{F}_q^*$, где су a и g познати
- Запишемо $n = mi + j$, где је $m = [\sqrt{q}]$ (цео део), $0 \leqslant i \leqslant m$, $0 \leqslant j \leqslant m - 1$
- Тада је $g^j = a\,(g^{-m})^i$
- Рачунамо парове $\left(j, g^j\right)$ и $\left(i, a\,(g^{-m})^i\right)$, за све i, j и чувамо их сортиране по другој координати

Алгоритам Гељфонд-Шенкса (baby-step-giant-step алгоритам)

- Циљ: израчунати $n = \log_g a$ у $\mathbb{F}_q^*$, где су a и g познати
- Запишемо $n = mi + j$, где је $m = [\sqrt{q}]$ (цео део), $0 \leqslant i \leqslant m$, $0 \leqslant j \leqslant m - 1$
- Тада је $g^j = a\,(g^{-m})^i$
- Рачунамо парове $\left(j, g^j\right)$ и $\left(i, a\,(g^{-m})^i\right)$, за све i, j и чувамо их сортиране по другој координати
- Када нађемо i и j за које се поклапа друга координата лако долазимо до n

Алгоритам Гељфонд-Шенкса (baby-step-giant-step алгоритам)

- Циљ: израчунати $n = \log_g a$ у $\mathbb{F}_q^*$, где су a и g познати
- Запишемо $n = mi + j$, где је $m = [\sqrt{q}]$ (цео део), $0 \leqslant i \leqslant m$, $0 \leqslant j \leqslant m - 1$
- Тада је $g^j = a\,(g^{-m})^i$
- Рачунамо парове $\left(j, g^j\right)$ и $\left(i, a\,(g^{-m})^i\right)$, за све i, j и чувамо их сортиране по другој координати
- Када нађемо i и j за које се поклапа друга координата лако долазимо до n
- Временска (али и просторна) сложеност $O\left(\sqrt{q}\log^2 q\right)$

Пример: $\log_3 37$ по модулу 101

- тада је $\left[\sqrt{101}\right] = 10$ и

giant step i	0	1	2	3	4	5	6	7	8	9
$3^{-10i} \pmod{101}$	1	14	95	17	36	100	87	6	84	65
$37 \cdot 3^{-10i} \pmod{101}$	37	13	81	23	19	64	88	20	78	82

- 3, 3^2 и 3^3 се не појављују у табели, али $3^4 = 81 \equiv 37 \cdot 3^{-20} \pmod{101}$
- следи $\log_3 37 = 20 + 4$