Міністерство освіти і науки України

Національний технічний університет України

"Київський політехнічний інститут імені Ігоря Сікорського"

Навчально-науковий Фізико-технічний інститут

Теоретико-числові алгоритми в криптології

Комп'ютерний практикум №3

Реалізація та застосування алгоритму дискретного логарифмування index-calculus

Виконали студенти:
група ФІ-12
Юрчук Олексій
група ФІ-14
Фурутіна Євгенія

Київ 2024

Тема: Реалізація та застосування алгоритму дискретного логарифмування index-calculus

Мета: Ознайомлення з алгоритмом дискретного логарифмування index-calculus. Програмна реалізація цього алгоритму та визначення його переваг, недоліків та особливостей застосування. Практична оцінка складності роботи та порівняння різних реалізацій цього алгоритму.

Завдання:
А) Написати програму, що реалізовує алгоритм index-calculus для груп типу Zp* без розпаралелювання та з розпаралелюванням етапу побудови системи лінійних рівнянь.
Б) Застосувати дві реалізації алгоритму index-calculus до самостійно сформованих задач дискретного логарифма, поступово збільшуючи порядок числа p. Рекомендується починати з порядку 3. Для кожної задачі заміряти час роботи обох реалізацій. Зупинити збільшення порядку числа p, коли час роботи хоча б однієї з реалізації перевищує певне значення, наприклад 5 хвилин. Мінімальне обмеження часу роботи – 3 хвилини. Для формування задач дискретного логарифма можна використовувати допоміжну програму з КП #2.
В) Порівняти результати часу роботи обох реалізацій алгоритму з пунктів 2 та 3. Створити візуалізації залежності часу роботи реалізації від порядку простого числа p. Пояснити результати.
Г) Сформувати результати дослідження, зокрема інформація про всі ітерації процесу збільшення порядку простого числа p — сформована задача і її розв’язок.
Теоретичні відомості:

- В загальному про алгоритм:

Алгоритм ***index-calculus***.

Вхід алгоритму:

- α – генератор групи G;
- $\beta \in G$;
- n – порядок групи G.

Вихід алгоритму: x таке, що $0 \le x \le n-1$, $\alpha^x = \beta$.

Кроки алгоритму:

1. Формуємо факторну базу з елементів групи $S = \{p_1, p_2, \dots, p_t\} \subseteq G$. Де $p_i, i = \overline{1,t}$ — прості числа, такі що $1 < p_i < B$, де $B = c \cdot \exp\left(\frac{1}{2} \cdot (\log n \log\log n)^{1/2}\right)$, c — деяка додатна константа. При виконанні цієї лабораторної роботи рекомендується використовувати константу $c = 3.38$.
2. Формуємо лінійні рівняння, невідомими в яких є дискретні логарифми за основою α кожного з елементів бази:
   a) Випадково обираємо значення k, $0 \le k \le n-1$ та обчислюємо α^k.

b) Перевіряємо значення α^k на гладкість за базою S:

$$\alpha^k = \prod_{i=1}^{t} p_i^{c_i},$$

де $c_i \geq 0$. Якщо α^k не є гладким, повертаємось на крок a) та обираємо нове значення k. Для цього кроку рекомендується використовувати реалізації алгоритмів факторизації з КП #1.

c) Логарифмуємо за основою α співвідношення, отримане в пункті b):

$$k = \sum_{i=1}^{t} c_i \log_\alpha p_i \bmod n,$$

де n – порядок групи.

d) Повторюємо кроки a)-c) поки не отримаємо систему рівнянь, що має єдиний розв'язок відносно значень $\log_\alpha p_i$, $i = \overline{1,t}$.

Зазвичай формують $t + c$ рівнянь, де t – розмір факторної бази, c – невелике додатне ціле число, наприклад, 10, 15, тощо. Цей крок легко розпаралелити, даючи різним процесорам шукати співвідношення незалежно, а вже отримані співвідношення збирає та обробляє центральний процесор.

3. Розв'язуємо сформовану в пункті 2 систему рівнянь відносно дискретних логарифмів за основою α кожного з елементів факторної бази: $\log_\alpha p_i$, $i = \overline{1,t}$.

4. Обчислюємо $\log_\alpha \beta$, використовуючи значення $\log_\alpha p_i$, $i = \overline{1,t}$:

a) Випадково обираємо значення l, $0 \leq l \leq n-1$ та обчислюємо $\beta \cdot \alpha^l$.

b) Перевіряємо, чи $\beta \cdot \alpha^l$ є гладким числом відносно факторної бази S, якщо ні – повертаємось на крок a) та обираємо нове значення l. Інакше маємо:

$$\beta \cdot \alpha^l = \prod_{i=1}^{t} p_i^{d_i},$$

де $d_i \geq 0$.

c) Логарифмуємо за основою α співвідношення з пункту b):

$$\log_\alpha \beta + l = \sum_{i=1}^{t} d_i \cdot \log_\alpha p_i \bmod n,$$

тобто шуканий дискретний логарифм обчислюється за співвідношенням:

$$\log_\alpha \beta = \sum_{i=1}^{t} d_i \cdot \log_\alpha p_i - l \bmod n,$$

де $\log_\alpha p_i$, $i = \overline{1,t}$ — відомі з попереднього кроку.

Хід роботи

Було виміряно час виконання алгоритму для кожної трійки значень. Результати були згруповані у таблицю:

Параметр p	Без розпаралелювання	Час роботи, seconds
p = 3	$\alpha = 3$ $\beta = 45$ $p = 149$ $x = 10$	0.0003396000247448683
p = 4	$\alpha = 2795$ $\beta = 473$	0.00038790004327893257

	$p = 3821$ $x = 581$	
p = 5	$\alpha = 21730$ $\beta = 7231$ $p = 80111$ $x = 62511$	0.005632999999339506
p = 6	$\alpha = 81130$ $\beta = 276305$ $p = 409163$ $x =$	Алгоритм підвис, на підборі значень розв'язку слр

Коментар:
Бачимо, що зі збільшенням порядку p, зростає час роботи алгоритму, але ця залежність не є експоненційною.

Висновки:
Найперше, що хочеться сказати – це про труднощі реалізації лабораторної:
- Найважчим, було реалізувати частину з правильним обчисленням розв'язків системи лінійних рівнянь. Доводилось підбирати числа, щоб обернена матриця існувала

В загальному, алгоритм index-calculus має свої переваги: він доволі швидкий(переконалися в цьому на практиці), його можна розпаралелити(на жаль не встигли, віримо на слово), та

Додатки:

Увесь код можна знайти за посиланням, на GitHub:
https://github.com/MansteinOrGuderian/NTA-3

Docker enjoyer instruction:

В термінал пишемо: docker run -i mansteinorguderian/ntathird:latest
(деталі запуску - https://github.com/MansteinOrGuderian/NTA-3/blob/main/Docker%20Instruction.md)