МИНОБРНАУКИ РОССИИ
ФГБОУ ВО «СГУ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

СХЕМА ДИФФИ – ХЕЛЛМАНА

ЛАБОРАТОРНАЯ РАБОТА №6

студента 4 курса 431 группы
направления 10.05.01 — Компьютерная безопасность
факультета КНиИТ
Никитина Арсения Владимировича

Проверил
доцент _____ А. В. Жаркова

Саратов 2022

СОДЕРЖАНИЕ

1 Задание лабораторной работы

Придумать схему разделения ключа аналогичную схеме Диффи – Хелл-мана между n пользователями для $n \geqslant 3$.

2 Теоретическая часть

2.1 Описание стандартного протокола Диффи – Хеллмана

Пусть известно большое конечное поле F и порождающий элемент g мультипликативной группы F^*. Двум абонентам A и B требуется выбрать секретным образом какое-нибудь случайное большое число (секретный), используя указанные данные в качестве платформы.Сам выбор поля F и порождающего элемента g осуществляется по открытому каналу, и поэтому данные F, g являются открытыми.

Процесс разделения ключа между двумя пользователями можно разделить на следующие составляющие:

1. Установка;
2. Генерация случайного числа и вычисление открытого ключа;
3. Разделение ключа между пользователями.

Установка

Абоненты A и B договариваются о выборе поля F и порождающего элемента g. Для достижения криптостойкости наиболее приемлемым считается выбор такого поля F, чтобы длина его была как минимум 2048 бит, а также чтобы оно было простым, причем $(F-1)/2$ также должно быть простым числом.

Генерация случайного числа и вычисление открытого ключа

Абонент A выбирает случайным образом натуральное число x и вычисляет $y \equiv g^x \ (mod\ F)$ и передает y абоненту B.

Абонент B выбирает случайным образом натуральное число z и вычисляет $u \equiv g^z \ (mod\ F)$ и передает u абоненту A.

Разделение ключа между пользователями

Абонент A, зная x и u вычисляет элемент $u^x \equiv g^{zx} \ (mod\ F)$.

Абонент A, зная z и y вычисляет элемент $y^z \equiv g^{xz} \ (mod\ F)$.

Таким образом, абоненты получили одно и то же число, причем никто кроме их самих этим ключом не владеет.

2.2 Описание алгоритма для большего количества пользователей

Пусть ключ требуется разделить между n пользователями: $n \geqslant 3,\ A_1, A_2, ..., A_n$.

Выполним аналогичные операции для большего количества пользователей. Для этого заметим, что если требуется разделить ключ между одним пользователем и всеми остальными, то он должен быть последним пользователем, к которому попадет исходное число после всех возведений его в степень.

Установка

Абоненты $A_1, .., A_n$ договариваются о выборе поля F и порождающего элемента g.

Генерация случайного числа и вычисление открытого ключа

Пусть каждый абонент $A_1, ..., A_n$ выбрал свое случайное число s_{A_i} и сгенерировал свой открытый ключ $O_i \equiv g^{s_{A_i}}\ (mod\ F)$ $(i = \overline{1, n})$.

Заметим, что если абонент A_i передал свой открытый ключ другому абоненту, то на данном шаге получить свой разделенный закрытый ключ он никак не сможет, поэтому генерация открытого ключа абонентом и передача его по открытому каналу связи будет свидетельствовать о том, что началась новая цепочка передач ключа.

Разделение ключа между пользователями

Воспользуемся следующим свойством коммутативности показателей при последовательном возведении в степень:

$$(g^b\ mod\ p)^a\ mod\ p \equiv (g^a\ mod\ p)^b\ mod\ p \equiv g^{ab}\ mod\ p$$

Таким образом, можно рассмотреть последовательное возведение в степень не только для двух показателей, но и для большего их количества.

Имеем n пользователей: $A_1, A_2, ..., A_n$.

Все вычисления производятся в по модулю p:

1. Пользователь A_1 вычисляет $g^{s_{A_1}}$ и передет результат пользователю A_2
2. Пользователь A_2 вычисляет $g^{s_{A_1} \cdot s_{A_2}}$ и передает результат пользователю A_3
3. $\cdots$

4. Пользователь A_n вычисляет $g^{(s_{A_1} \cdot s_{A_2} \cdots \cdot s_{A_{n-1}}) \cdot s_{A_n}}$ и получает общий секретный ключ.
5. Затем пользователь A_2 вычисляет $g^{s_{A_2}}$ и передет результат пользователю A_3
6. Пользователь A_3 вычисляет $g^{s_{A_2} \cdot s_{A_3}}$ и передает результат пользователю A_4
7. $\cdots$
8. Пользователь A_n вычисляет $g^{s_{A_2} \cdot s_{A_3} \cdots \cdot s_{A_n}}$ и передет результат пользователю A_1.
9. Пользователь A_1 вычисляет $g^{(s_{A_2} \cdot s_{A_3} \cdots \cdot s_{A_n}) \cdot s_{A_n}}$ и получает общий секретный ключ.
10. Таким образом, общий секретный ключ будет получен для всех пользователей.

Для пользователя с номером $k : 2 \leq k < n$ секретный ключ будет вычисляться по формуле:

$$(((((g^{s_{A_{k+1}}} \ mod \ p)^{s_{A_{k+2}}} \ mod \ p...)^{s_{A_n}} \ mod \ p)^{s_{A_1}} \ mod \ p...)^{s_{A_{k-1}}} \ mod \ p)^{s_{A_k}} \ mod \ p \equiv$$
$$(g^{s_{A_1} \cdot \ldots \cdot s_{A_{k-1}} \cdot s_{A_{k+1}} \cdot \ldots \cdot s_{A_n}} \ mod \ p)^{s_{A_k}} \ mod \ p \equiv g^{s_{A_1} \cdot \ldots \cdot s_{A_{k-1}} \cdot s_{A_{k+1}} \cdot \ldots \cdot s_{A_n} \cdot s_{A_k}} \ mod \ p.$$

Для пользователя с номером $k = 1$ секретный ключ будет вычисляться по формуле:

$$(((g^{s_{A_2}} \ mod \ p)^{s_{A_3}} \ mod \ p...)^{s_{A_n}} \ mod \ p)^{s_{A_1}} \ mod \ p \equiv (g^{s_{A_2} \cdot \ldots \cdot s_{A_n}} \ mod \ p)^{s_{A_k}} \ mod \ p \equiv$$
$$g^{s_{A_2} \cdot \ldots \cdot s_{A_n} \cdot s_{A_k}} \ mod \ p.$$

Для пользователя с номером $k = n$ секретный ключ будет вычисляться по формуле:

$$(((g^{s_{A_1}} \ mod \ p)^{s_{A_2}} \ mod \ p...)^{s_{A_{n-1}}} \ mod \ p)^{s_{A_n}} \ mod \ p \equiv$$
$$(g^{s_{A_1} \cdot \ldots \cdot s_{A_{n-1}}} \ mod \ p)^{s_{A_k}} \ mod \ p \equiv g^{s_{A_1} \cdot \ldots \cdot s_{A_{n-1}} \cdot s_{A_k}} \ mod \ p.$$

3 Практическая часть

3.1 Пример работы алгоритма

```
Введите простое число p: 31
Введите любое число из множества порождающих элементов (3, 11, 12, 13, 17, 21, 22, 24): 3
Введите количество пользователей: 3
Введите секретное число для пользователя 1: 12
Введите секретное число для пользователя 2: 21
Введите секретное число для пользователя 3: 23
1: 2 -> 3 -> 1
Пользователь 2 вычислил 3 ^ 21 mod 31 и получил 15, затем передал это значение пользователю 3
Пользователь 3 вычислил 15 ^ 23 mod 31 и получил 27, затем предал это значение пользователю 1
Пользователь 1 вычислил 27 ^ 12 mod 31 и получил секретный ключ 16.

2: 3 -> 1 -> 2
Пользователь 3 вычислил 3 ^ 23 mod 31 и получил 11, затем передал это значение пользователю 1
Пользователь 1 вычислил 11 ^ 12 mod 31 и получил 16, затем предал это значение пользователю 2
Пользователь 2 вычислил 16 ^ 21 mod 31 и получил секретный ключ 16.

3: 1 -> 2 -> 3
Пользователь 1 вычислил 3 ^ 12 mod 31 и получил 8, затем передал это значение пользователю 2
Пользователь 2 вычислил 8 ^ 21 mod 31 и получил 8, затем предал это значение пользователю 3
Пользователь 3 вычислил 8 ^ 23 mod 31 и получил секретный ключ 16.
```

Рисунок 1

3.2 Код программы, реализующей рассмотренный алгоритм

```
import math


def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a


def coprime(a, b):
    return gcd(a, b) == 1


def is_prime(number):
    d = math.ceil(math.sqrt(number))
    x = 2
    while x <= d:
        if number % x == 0:
            return False
        x += 1
    return True


```

```
def get_parent_elements(m):

    counter = 1
    res = []

    for i in [i for i in range(2, m) if coprime(i, m)]:

        current_counter = 1
        elem_save = elem = i

        while elem != 1:
            elem *= elem_save
            elem %= m
            current_counter += 1

        if current_counter == counter:
            res.append(i)
        elif current_counter > counter:
            res = [i]
            counter = current_counter

    return res, counter


def order(elems):
    sl = {}
    for i in range(len(elems)):
        sl[elems[i-1]] = elems[i:] + elems[:i]
    return sl


def encrypt_message(message, full_key):
    encrypted_message = ""

    for symbol in message:
        encrypted_message += chr(ord(symbol) + full_key)
    return encrypted_message


def get_secret_number(subscriber):
    while True:
```

```
65          try:
66              secret_number = int(input(f'Введите секретное число для ' \
67                  f'пользователя {subscriber}: '))
68              if secret_number > 0:
69                  return secret_number
70              else:
71                  print('Вы ввели не положительное целое число')
72          except ValueError:
73              print('Вы ввели не число')
74
75
76  def make_secret_value(sub_order, subscribers_secret_numbers, p, g):
77
78      current_value = (g ** subscribers_secret_numbers[sub_order[0]]) % p
79
80      print(f'Пользователь {sub_order[0]} вычислил {g} ^ ' \
81          f'{subscribers_secret_numbers[sub_order[0]]} mod {p} и получил ' \
82              f'{current_value}, затем передал это значение пользователю ' \
83              f'{sub_order[1]};')
84
85      for current_sub in sub_order[1:]:
86
87          print(f'Пользователь {current_sub} вычислил {current_value} ^ ' \
88              f'{subscribers_secret_numbers[current_sub]} mod {p} ', end ='')
89
90          current_value = (current_value ** \
91              subscribers_secret_numbers[current_sub]) % p
92
93          if current_sub != sub_order[-1]:
94              print(f'и получил {current_value}, затем предал это значение ' \
95                  f'пользователю {sub_order[sub_order.index(current_sub) +
                    ↪ 1]};')
96          else:
97              print(f'и получил секретный ключ {current_value}.\n')
98
99
100 def main():
101
102     while True:
103         try:
104             p = int(input('Введите простое число p: '))
```

```
105             if not is_prime(p):
106                 print('Вы ввели не простое число')
107             else:
108                 break
109         except ValueError:
110             print('Вы ввели не число')
111 
112     parent_elements, counter = get_parent_elements(p)
113 
114     while True:
115         try:
116             g = int(input(f'Введите любое число из множества порождающих ' \
117                 f'элементов {tuple(parent_elements)}: '))
118             if not g in parent_elements:
119                 print('Вы ввели не порождающий элемент')
120             else:
121                 print(f'Порядок: {counter}')
122                 break
123         except ValueError:
124             print('Вы ввели не число')
125 
126     while True:
127         try:
128             number_of_subscribers = int(input('Введите количество ' \
129                 'пользователей: '))
130             break
131 
132         except ValueError:
133             print('Вы ввели не число')
134 
135     subscribers = [*range(1, number_of_subscribers + 1)]
136 
137     subscribers_secret_numbers = {i: get_secret_number(i) for i in
    ↪  subscribers}
138 
139     subscribers_order = order(subscribers)
140 
141     for subscriber in subscribers:
142         print(f'{subscriber}: ' + ' -> '.join(str(elem) for elem in \
143             subscribers_order[subscriber]))
144         make_secret_value(subscribers_order[subscriber], \
```

```
145             subscribers_secret_numbers, p, g)
146
147
148 if __name__ == '__main__':
149     main()
```