

Министерство науки и высшего образования Российской Федерации  
Санкт-Петербургский Политехнический Университет Петра Великого

---

Институт компьютерных наук и кибербезопасности  
Высшая школа кибербезопасности

## **ЛАБОРАТОРНАЯ РАБОТА № 4**

### **«Криптосистемы на основе задачи дискретного логарифмирования»**

по дисциплине «Криптографические методы защиты информации»

Выполнил

студент гр. 45151004/90101

Кондачков Е.Д.

Преподаватель

Ассистент

Зубков Е.А.

Санкт-Петербург

2023

## Содержание

**Элементы оглавления не найдены.**

## **1. Цель работы**

Изучение протоколов шифрования с открытым ключом и электронной подписи, безопасность которых основана на задаче дискретного логарифмирования в конечном поле.

## **2. Задачи работы**

1. Согласно полученному варианту задания разработать криптосистему на основе задачи дискретного логарифмирования;
2. Сравнить разработанную систему с RSA, выявить преимущества и недостатки.

### 3. Ход работы

В ходе работы была реализована криптосистема по протоколу Диффи-Хеллмана, которая основывается на передаче пользователей друг другу частей ключа.

```
AttributeError: 'NoneType' object has no attribute 'typ'
PS T:\repos\Python\Cryptography\discrete_logarithm> python .\client.py
Generated parameters:
a = 3
x = 356757766703403976653295013783456848325076799385977812939236269862010539
9549453115298259976522273115560174729748876802777977497480440244033731148530
0672279617555865427862782585136388327198956017117244063339057359920227769260
3116564361437072621578104254251117523574640972877973464118746234559827048055
13658947
Rec'd a*y = 39050306784525501417159350169209389111767523724232565181967866
089980555994857
Ka = 73012832805447588242717959265755321964188346278489888001026378476689424
942969
AES key = 730128328054475882427179592657553219641883462784898880010263784766
89424942969
Enter the message:

PS T:\repos\Python\Cryptography\discrete_logarithm> python .\server.py
Connect client ('127.0.0.1', 6994)
Rec'd parameters:
a = 3
a*x = 405333625715726790802448327425997985603831211608714471999417273814849
68437531
Generated y = 3742711502444573621605234961732871053079193676072531876462961
44212515387532431542644783621486830337298374078241028219352432795768219519
529025986665938810137086762342703074940393205355913109715585021269214924229
966851947666630298791064147456107180589420890990214601119162862477306803489
2140222422338482720474
Kb = 7301283280544758824271795926575532196418834627848988800102637847668942
4942969
AES key = 730128328054475882427179592657553219641883462784898880010263784766
89424942969
```

Рисунок 1 – Работа алгоритма

На рисунке выше показано, как алгоритм производит обмен ключами:

1. Клиент генерирует случайный показатель  $x$  и считывает показатель  $a$ ;
2. Клиент отправляет серверу ASN1 файл, содержащий  $a$  и  $a^x$ ;
3. Сервер генерирует свой закрытый показатель  $y$ ;
4. Сервер расшифровывает ASN1 файл, берет оттуда  $a$  и отправляет клиенту  $a^y$ ;
5. Клиент (сервер), возводит  $a^x(a^y)$  в степень  $y(x)$ , получая таким образом ключ.

Контейнер с сообщением в зашифрованном виде выглядит следующим образом.

```
cryptocontainer
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Декодированный текст
00000000 30 13 31 0F 30 0D 04 02 10 82 02 01 20 30 00 30 0.1.0.... 0.0
00000010 00 30 00 30 00 04 20 48 03 71 C7 FB 2E 8F FB 5C .0.0.. Н.СЫ.Цы\
00000020 0D 8B 46 E3 33 7E 00 7D 7B EA 05 B7 35 CF 52 38 .<Fr3~.}к.·5PR8
00000030 E4 22 C4 F5 06 5C 6F д"Дх.\o
```

Рисунок 2 – Контейнер с зашифрованным сообщением

```
Certificate SEQUENCE (2 elem)
  tbsCertificate TBSCertificate [?] SET (1 elem)
    serialNumber CertificateSerialNumber [?] SEQUENCE (5 elem)
      OCTET STRING (2 byte) 1082
      INTEGER 32
      SEQUENCE (0 elem)
      SEQUENCE (0 elem)
      SEQUENCE (0 elem)
    signatureAlgorithm AlgorithmIdentifier SEQUENCE (0 elem)
```

Рисунок 3 – Расшифрованный контейнер

#### 4. Ответы на контрольные вопросы

1. Пусть  $G$  – конечная циклическая группа с заданной образующей. Порядок группы  $G$  известен и делится на простое число  $r$ . Как найти образующую циклической подгруппы порядка  $r$ ?

$$r * \frac{\text{ord}(G)}{r}$$

2. Покажите, что задача Мессии–Омуры сводится к задаче дискретного логарифмирования.

Дано:  $t^a, t^b, t^{ab}$ . Найти:  $t$ . Введем обозначения:  $r = t^b, s = t^{ab}$ .

$$s = r^a$$

Задача определения  $a$  по известным  $s$  и  $r$  как раз является задачей дискретного логарифмирования.

3. Подвержен ли протокол шифрования Эль-Гамала к атаке на основе подобранных шифртекстов?

Такая атака заключается в сборе криптоаналитиком информации о шифре путем подбора шифртекстов и получения некоторых их расшифровок без известной информации о ключе. При сопоставлении ставших известными блоков исходного текста и его шифртекстов криптоаналитик может восстановить секретный ключ. При этом данной атаке могут быть подвержены шифр по схеме Эль-Гамала, RSA.

4. Перечислите задачи, положенные в основу схемы подписи Эль-Гамала. Почему важно, чтобы период генератора случайных чисел, используемого в схеме подписи, был достаточно большим?

- Увеличение криптостойкости при подписании;
- Криптостойкость основана на вычислительной сложности проблемы дискретного логарифмирования;
- Возможность многократно подписывать различные сообщения одной электронной подписью (одним ключом);
- Схема Эль-Гамала явилась прототипом множеству современных стандартов электронных подписей;

– Замена длинных чисел в операциях на остатки от деления от основы криптосистемы.

Период генератора случайных чисел, используемого в схеме подписи, должен быть достаточно большим, чтобы итоговое значение подписи не стало повторяться на разных сообщениях в период использования одного секретного ключа.

## **5. Выводы по проделанной работе**

В ходе данной лабораторной работы была разработана криптосистема, использующая протокол Диффи-Хеллмана. Данная криптосистема выгодно отличается от RSA своей простотой в установке ключа, так как для этого ей не требуется производить операцию логарифмирования, как, например, в первой лабораторной работе, где ключ шифровался посредством RSA. В то же время, данный алгоритм требует весьма трудоемкой операции возведения в степень, которая выполняется значительное время.



## Приложение А

### Листинг client.py

```
#this is a client
from config import address, port, a
import socket
from asn import ASN
import random as rnd
from Crypto.Util.number import getPrime
from Crypto.Random import get_random_bytes

def connect_server() -> socket.socket:
    connection = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    connection.connect((address, port))
    return connection

def get_parameter(p) -> int:
    return rnd.randint(1, p)

def exchange_keys(connection: socket.socket) -> int:
    p = getPrime(1024, randfunc=get_random_bytes)
    x = get_parameter(p)
    asn = ASN.encrypt_diffie_hellman(a**x, "client", p, a)
    connection.send(asn)
    asn = connection.recv(4096)
    c = int(ASN.decrypt_diffie_hellman(asn))
    print(f'Generated parameters:\na = {a}\nx = {x}')
    print(f'Recived a^y = {c}')
    key = c**x
    print(f'Ka = {key}')
    return key

def main():
    connection = connect_server()
    key = exchange_keys(connection)
    AESkey = key % (2**256)
    print(f'AES key = {AESkey}')
```

```
while(True):
    message = input("Enter the message: ")
    message_bytes = bytes(message, 'utf-8')
    message_len = len(message_bytes)
    extension_length = 16 - message_len
    for i in range (extension_length):
        message_bytes += b'\x03'
    asn = ASN.encrypt_aes_diffie_hellman(message_len, message_bytes)
    connection.send(asn)
    with open("cryptocontainer", "wb") as enc:
        enc.write(asn)
    if message == 'exit':
        break
    connection.close()
```

```
if __name__ == '__main__':
    main()
```

## Приложение Б

### Листинг server.py

```
#this is a server
import socket
from asn import ASN
import random as rnd

def start_server() -> socket.socket:
    listener = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    listener.bind(('127.0.0.1', 8000))
    listener.listen(0)
    connection, address = listener.accept()
    print(f'Connect client {address}\n')
    return connection

def get_parameter(p) -> int:
    return rnd.randint(1, p)

def exchange_keys(connection:socket.socket) -> int:
    asn = connection.recv(4096)
    p, a, c = ASN.decrypt_diffie_hellman(asn, "client")
    print(f'Recived parameters:\na = {a}\na^x = {c}')
    y = get_parameter(p)
    print(f'Generated y = {y}')
    asn = ASN.encrypt_diffie_hellman(a**y)
    connection.send(asn)
    key = c**y
    print(f'Kb = {key}')
    return key

def main():
    connection = start_server()
    key = exchange_keys(connection)
    AESkey = key % (2**256)
    print(f'AES key = {AESkey}')
    while(True):
        asn = connection.recv(4096)
```

```
        message_bytes, len = ASN.decrypt_aes_diffie_hellman(asn)
        message_bytes = message_bytes[:len]
        message = bytes.decode(message_bytes)
        print(message)
        if message == 'exit':
            break
    connection.close()

if __name__ == '__main__':
    main()
```