МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Протоколы открытого распределения ключей по алгоритму Хьюза

ОТЧЁТ
ПО ДИСЦИПЛИНЕ
«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»

студентки 5 курса 531 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Шуликиной Анастасии Александровны

Преподаватель Р. А. Фарахутдинов
_____
подпись, дата

Саратов 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данной лабораторной работе поставлена задача рассмотрения алгоритма Хьюза и протоколов открытого распределения ключей, на основе изученного материала программно реализовать протокол открытого распределения ключей, используя алгоритм Хьюза.

1 Цель работы и порядок её выполнения

Цель работы – изучение алгоритма Хьюза и его реализация в протоколе открытого распределения ключей.

Порядок выполнения работы:

1. Разобрать, что такое протоколы открытого распределения ключей.
2. Разобрать алгоритм Хьюза.
3. Произвести программную реализацию по созданию протокола открытого распределения ключей с помощью алгоритма Хьюза.

2 Теория

2.1 Протоколы открытого распределения ключей

Протокол (протокол) – описание последовательности алгоритмов, в процессе выполнения которой два или более участников последовательно исполняют эти алгоритмы и обмениваются сообщениями с целью решения некоторой поставленной перед ними задачи.

В качестве участников (субъектов, сторон) протокола могут выступать не только пользователи или абоненты, но и клиентские и серверные приложения.

Криптографическим протоколом (cryptographic protocol) называется любой протокол, в используются криптографические системы криптографические алгоритмы.

Основными характеристиками криптографического протокола являются:

— Прозрачность. Каждый участник протокола должен знать протокол и всю последовательность его действий.

— Однозначность. Действие каждого участника в протоколе должно быть однозначно определено.

— Полнота. Протокол должен быть полным — в нем должны быть указаны точные действия в любой возможной ситуации.

Основные задачи криптографических протоколов: криптографического протокола являются:

— Конфиденциальность (секретность какой-либо части информации);

— Аутентичность (подтверждение целостности или авторства);

— Неотслеживаемость предметов и субъектов протокола.

Основное требование к криптографическому протоколу гласит, чтобы невозможно было сделать или узнать больше, чем определено протоколом.

В криптографической практике при каждом сеансе связи принято пользоваться новым ключом шифрования. Этот ключ называют сеансовым. Такое ограничение на существование ключа является важным условием надёжности конфиденциальной связи, но в то же время создаёт дополнительную сложную задачу передачи сеансового ключа.

Если система имеет k пользователей, тогда для возможных секретных связей потребуется $C_k^2 = \frac{k(k-1)}{2}$ ключей. Но при этом мало вероятно, что все

эти ключи или большинство из них будут востребованы. Чтобы уменьшить утечку информации о ключах и облегчить процесс генерации ключей создают протоколы распределения ключей только по запросу на очередной сеанс связи.

Открытое распределение ключей (public key distribution) — протокол совместной выработки пользователями (общего) секретного ключа путем обмена сообщениями по открытому каналу связи. Протокол должен исключать возможность получения информации о ключе посторонними, а также любым участником до завершения им действий, предусмотренных протоколом.

2.2 Алгоритм Хьюза

В протоколе Диффи-Хеллмана по сути секретный ключ генерируется по ходу его проведения и никто не знает на сколько хорошими характеристиками он будет обладать. Что является одним из заметных недостатков этого протокола. Иной вариант алгоритма Диффи-Хеллмана, предложенный Хьюзом (Hughes), позволяет Алисе сначала генерировать ключ, проверить его надёжность, и уже потом послать его Бобу. Общие параметры те же. Он так же двухпроходный, графическая схема изображена на рис. 1

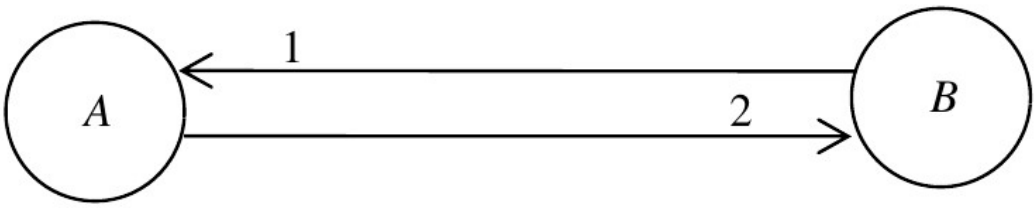

Рисунок 1 – Схема двухпроходного алгоритма Хьюза

1. Алиса выбирает случайное большое целое число x и генерирует $k = g^x mod n$;
2. Боб выбирает случайное большое целое число y и посылает Алисе $Y = g^y mod n$;
3. Алиса посылает Бобу $X = Y^x mod n$;
4. Боб вычисляет $z = y^{-1}$ $k' = X^z mod n$.
5. Проверяем k и k', если все выполнено правильно, то $k == k'$.

Преимуществом этого протокола над протоколом Диффи-Хелмана состоит в том, что k можно вычислить заранее, до взаимодействия, и Алиса может шифровать сообщения с помощью k задолго до установления соединения

с Бобом. Она может послать сообщение сразу множеству людей, а передать ключ позднее каждому по отдельности.

3 Программная реализация

3.1 Результаты тестирования программы

На рисунках 1-5 можно увидеть работу, реализуемой программы, по рассмотренным алгоритмам.

```
"D:\SSU\Слеповичев\on python\Scripts\python.exe" "D:\SSU\Протоколы\hughes\on python\main.py"
Введите количество бит для генерации ключей (не более 1024 бит): 32
Публичный ключ Алисы (n, e): (19628119199073924447, 65537)
Приватный ключ Алисы (n, d): (19628119199073924447, 19434344761894109993)
Публичный ключ Боба (n, e): (30021801917110066997, 65537)
Приватный ключ Боба (n, d): (30021801917110066997, 19586984524558870997)
Введите сообщение, которое Алиса хочет отправить: 555557895
Зашифрованное сообщение, отправленное от Алисы к Бобу: 2902352616843716141
Расшифрованное сообщение Боба: 555557895
Сообщение было успешно зашифровано и расшифровано.
```

Рисунок 2 – Тест программы для отношения эквивалентности

```
Введите количество бит для генерации ключей (не более 1024 бит): 1024
Публичный ключ Алисы (n, e): (152915659378579901698563164502695589199846929358849406548136440285031832280392971917330368358328378764439815641773215971
Приватный ключ Алисы (n, d): (152915659378579901698563164502695589199846929358849406548136440285031832280392971917330368358328378764439815641773215971
Публичный ключ Боба (n, e): (98892660605392742551328415266707319552500537792335366454990210914291886985550641921284273790997844517758302855845111637
Приватный ключ Боба (n, d): (98892660605392742551328415266707319552500537792335366454990210914291886985550641921284273790997844517758302855845111637
Введите сообщение, которое Алиса хочет отправить: 555555555555555555555555588888888888888845
Зашифрованное сообщение, отправленное от Алисы к Бобу: 153692705878662052859681131908617514677785144877843790832892171973137651698718381530919998963
Расшифрованное сообщение Боба: 555555555555555555555555588888888888888845
Сообщение было успешно зашифровано и расшифровано.
```

Рисунок 3 – Результат программы проверки протокола открытого распределения ключей с помощью алгоритма Хьюза

```
Введите количество бит для генерации ключей (не более 1024 бит): 128
Публичный ключ Алисы (n, e): (41255473418320784977833921515205919495112281239227554280698218415031592830233, 65537)
Приватный ключ Алисы (n, d): (41255473418320784977833921515205919495112281239227554280698218415031592830233, 20673375383845352998413778727754972032965018511
Публичный ключ Боба (n, e): (22734303491241978407575906818305459040955382435430347079135253016874855309949, 65537)
Приватный ключ Боба (n, d): (22734303491241978407575906818305459040955382435430347079135253016874855309949, 87070054721786724755505326935847997607521091297
Введите сообщение, которое Алиса хочет отправить: 41255473418320784977833921515205959
Зашифрованное сообщение, отправленное от Алисы к Бобу: 19985654558490320064734461398868144860834872990112432386795564539742513271068
Расшифрованное сообщение Боба: 41255473418320784977833921515205959
Сообщение было успешно зашифровано и расшифровано.
```

Рисунок 4 – Результат программы проверки протокола открытого распределения ключей с помощью алгоритма Хьюза

3.2 Код программы, на основе рассмотренных алгоритмов, на языке Python

```
import random

def generate_prime(bits):
while True:
num = random.getrandbits(bits)
num |= 1  # Ensure it's odd
if is_prime(num):
return num

def is_prime(n, k=5):
if n <= 1:
return False
if n <= 3:
return True

for _ in range(k):
a = random.randint(2, n - 2)
if pow(a, n - 1, n) != 1:
return False

return True

def generate_keys(bits):
p = generate_prime(bits)
q = generate_prime(bits)

n = p * q
phi_n = (p - 1) * (q - 1)

e = 65537

d = mod_inverse(e, phi_n)
```

```
public_key = (n, e)
private_key = (n, d)

return public_key, private_key

def mod_inverse(a, m):
g, x, y = extended_gcd(a, m)
if g != 1:
raise Exception('Modular inverse does not exist')
return x % m

def extended_gcd(a, b):
if a == 0:
return (b, 0, 1)
else:
g, x, y = extended_gcd(b % a, a)
return (g, y - (b // a) * x, x)

def encrypt(message, public_key):
n, e = public_key
return pow(message, e, n)

def decrypt(ciphertext, private_key):
n, d = private_key
return pow(ciphertext, d, n)

def main():
bits = int(input("Enter the number of bits to generate keys
(max. 1024 bits): "))
if bits <= 0 or bits > 1024:
print("Incorrect number of bits.")
return
```

```
alice_public_key, alice_private_key = generate_keys(bits)
bob_public_key, bob_private_key = generate_keys(bits)

print("Alice's public key (n, e):", alice_public_key)
print("Alice's private key (n, d):", alice_private_key)
print("Bob's public key (n, e):", bob_public_key)
print("Bob's private key (n, d):", bob_private_key)

message = int(input("Enter the message Alice wants to send: "))
encrypted_message = encrypt(message, bob_public_key)

decrypted_message = decrypt(encrypted_message,
bob_private_key)

print("Encrypted message sent from Alice to Bob:",
encrypted_message)
print("Bob's decrypted message:", decrypted_message)

if message == decrypted_message:
print("The message was successfully encrypted and decrypted..")
else:
print("Error while encrypting and/or decrypting a message.")

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

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были рассмотрены теоретические сведения о протоколах открытого распределения ключей и подробно рассмотрен алгоритм Хьюза. Также, на основе рассмотренных материалов, была реализована программа.