实验四 Needham-Schroeder Protocol

课程名称：《信息安全》SOFT130018.01

任课老师：李景涛

助教：雷哲

实验目的

- Understanding Needham-Schroeder (Public Key) Protocol
- Understanding man-in-the-middle(MITM) attack against Needham- Schroeder (Public Key) Protocol

实验内容

Needham-Schroeder protocol allows to prove the identity of the end users communicating, and also prevents a middle man from eavesdropping.

Many existing protocols are derived from one proposed by Needham and Schroeder (1978), including the widely used Kerberos authentication protocol suite.

There are two types of Needham-Schroeder protocol.

- Needham-Schroeder protocol with *symmetric* key
- Needham-Schroeder protocol with *asymmetric* key

The symmetric one is used in kerberos infrastructure. But we will examine Needham-Schroeder protocol with *asymmetric* key encryption for educational purpose.

The public-key protocol

Alice (A) and Bob (B) use a trusted server (S) to distribute public keys on request. S holds Alice's public key K_{PA} and Bob's public key K_{PB}. $S's$ public key, K_{PS}, is well known.

- K_{PA} and K_{SA}, respectively public and private halves of an encryption key-pair belonging to A.
- K_{PB} and K_{SB}, similar belonging to B.
- K_{PS} and K_{SS} , similar belonging to S.

Now Alice (A) and Bob (B) wish to authenticate with each other and they propose to use the following protocol:

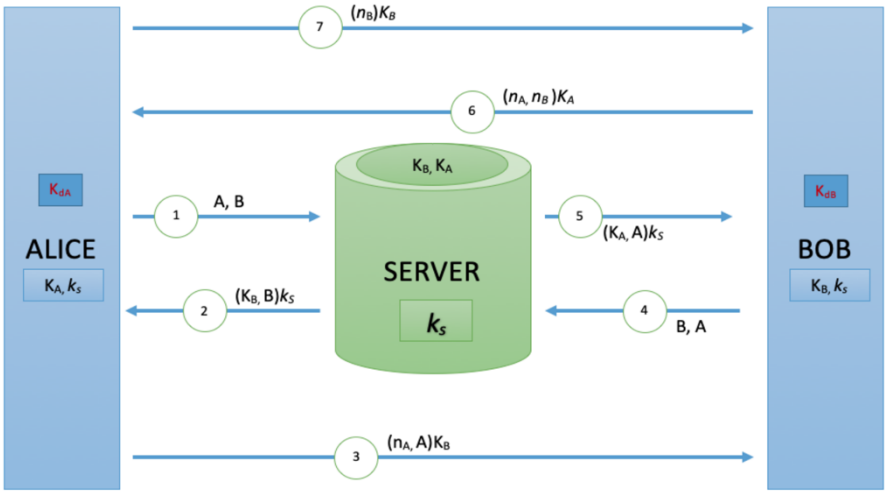
1) Dear S, This is A and I would like to get B's public key. Yours sincerely, A.

2) Dear A, Here is B's public key signed by me. Yours sincerely, S.

3) Dear B, This is A and I have sent you a nonce only you can read. Yours sincerely, A.
4) Dear S, This is B and I would like to get A's public key. Yours sincerely, B.

5) Dear B, Here is A's public key signed by me. Yours sincerely, S.
6) Dear A, Here is my nonce and yours, proving I decrypted it. Yours sincerely, B.
7) Dear B, Here is your nonce proving I decrypted it. Yours sincerely, A.

1. $A \rightarrow S: A, B$
2. $S \rightarrow A: \{K_{PB}, B\}_{K_{SS}}$
3. $A \rightarrow B: \{N_A, A\}_{K_{PB}}$
4. $B \rightarrow S: B, A$
5. $S \rightarrow B: \{K_{PA}, A\}_{K_{SS}}$
6. $B \rightarrow A: \{N_A, N_B\}_{K_{PA}}$
7. $A \rightarrow B: \{N_B\}_{K_{PB}}$



At the end of the protocol, A and B know each other's identities, and know both N_A and N_B. These nonces are not known to eavesdroppers.

Implement this protocol to demonstrate how it works.

Attacking the Needham-Scroeder (Public Key) Protocol

This protocol is vulnerable to a man-in-the-middle attack. If an impostor I can persuade A to initiate a session with them, they can relay the messages to B and convince B that he is communicating with A.

Ignoring the traffic to and from S, which is unchanged, the attack runs as follows:

$A \rightarrow I: \{N_A, A\}_{K_{PI}}$

$I \rightarrow B: \{N_A, A\}_{K_{PB}}$

$B \rightarrow I: \{N_A, N_B\}_{K_{PA}}$

$I \rightarrow A: \{N_A, N_B\}_{K_{PA}}$

$A \rightarrow I: \{N_B\}_{K_{PI}}$

$I \rightarrow B: \{N_B\}_{K_{PB}}$

At the end of the attack, B falsely believes that A is communicating with him, and that N_A and N_B are known only to A and B.

实验步骤

```
$ pip install
pycryptodome
```

- Client/Server/PKI/Adversary 分别对应于上述协议和攻击描述中的 A/B/S/I.
- `pki.py` : This re represents the trusted server who returns the RSA public key requested encrypted by the requester's public key.
- `client.py` : This le represents a client who wants to transfer a le to a storage server.
- `server.py` : This le represents a simple le storage server.
- `adversary.py` : This le represents Adversary, a malicious le storage server.
- `.asc` 文件为相应的 RSA 公钥或私钥。（也可以自己写函数新生成）

1. 实现 PKI

```
def extract():
    """() -> NoneType
    Opens the public key infrastructure server to extract RSA public
keys.
```

```
    The public keys must have already been in the server's folder.
    """
# A, B --->
# <--- {K_PB, B}(K_PA)
```

2. 实现 NS 公钥协议

Implementation `ns_authentication` in `client.py` and `server.py`

```
def ns_authentication(sock, server_name):
    """(socket, str) -> bytes or NoneType
    Performs authentication via Needham-Schroeder public-key
protocol.
    Returns a symmetric session key if authentication is successful,
    a None otherwise.

    :sock: connection to storage server
    :server_name: name of storage server
    """
    # WRITE YOUR CODE HERE!

    # get RSA key of Client
    # get public key of file transfer server
    # A -- {N_A, A}(K_PB) --> B
    # A <-- {N_A, N_B}(K_PA) -- B
    # check if Server actually did recieve Client's nonce
    # A -- {K, N_B}(K_PB) --> B
    # get confirmation

    print("Client: connection verified!")
    return ssn_key

def ns_authentication(conn):
    """(socket, str) -> bytes or NoneType
    Performs authentication via Needham-Schroeder public-key
protocol.
    Returns a symmetric session key and client's name if
authentication
    is successful, a None otherwise.

    :sock: connection to storage server
    :server_name: name of storage server
    """
    # WRITE YOUR CODE HERE!
```

```
print("Server: connection verified!")
return ssn_key, client_name
```

To run the Needham-Schroeder protocol between Client and Server, first open a command-line shell and run server.py with the following command:

```
$ python server.py
```

To execute the protocol, go to the `client` folder and open another shell window to execute `client.py` :

```
$ python client.py -s server -u my_file.txt
```

After running, a folder named `client` with the inputted le should appear under the server folder.

3. 实现对 NS 公钥协议的中间人攻击

Implementing `attack` :

```
def attack(conn):
    """(socket) -> (bytes, str) or NoneType
    Performs a man-in-the-middle attack between the client and Bob's
storage server.
    Returns the session key and clients name if attack was
successful, otherwise
    returns None.

    :conn: connection to the client (victim)
    """
    # get RSA key of Adversary for decrypting
    # A -- {N_A, A}(KP_M) --> M
    # get public key of Server for encrypting
    # reencrypt request for Server
    # open connection with Server
    # M -- {N_A, A}(KP_B) --> B
    # M <-- {N_A, N_B}(KP_A) -- B
    # A <-- {N_A, N_B}(KP_A) -- M
    # A -- {K, N_B}(KP_M) --> M
    # M -- {K, N_B}(KP_B) --> B
    # check if MITM attack was successful
    # WRITE YOUR CODE HERE!
```

```
    if int(sock.recv(1024)) == RESP_VERIFIED:
        print("Adversary: I got in!")
        upload_bad_file(sock, ssn_key)
        return ssn_key, client_name
    else:
        print("Adversary: wtf...")
    print("Adversary: attack completed")
```

To run the attack, adversary.py and server.py must be first running in separate shells:

```
$ python server.py
$ python adversary.py
```

To execute the attack, go to the client folder and open another shell to execute client.py :

```
$ python client.py -s adversary -u my_file.txt
```

After running, a le named bad_file.txt would be appear in the client folder and the server folder.

实验要求和评分

- 编程语言、编译运行时、所用工具等实验环境原则上不限，**建议**在给出的 Python 程序框架基础上补完。（如使用其他实验环境，需要完成同等任务，并在实验报告中写明。）
- 评分内容如下：

内容	总分 100
PKI	20
NS	30
NS Attack	40
Document(实验报告)	10

实验提交

- 不分组，独立完成 project
- 提交内容清单：

1. 实验报告（按**实验报告模板**书写，提交 pdf 格式文件，命名格式：学号+姓名+实验四.pdf）
2. 项目源代码（命名格式：学号_ns 压缩文件夹）
3. 鼓励录制短视频，介绍代码结构、演示运行结果、分析计算量等

- 提交方式：所有提交内容以压缩文件形式上传 elearning 提交（命名格式：学号+姓名+实验四）
- 提交截止时间：参照 eLearning 作业提交截止时间

参考资料

- https://en.wikipedia.org/wiki/Needham%E2%80%93Schroeder_proto col
- Socket Programming in Python (Guide)