

PUBLIC KEY DISTRIBUTION AUTHORITY

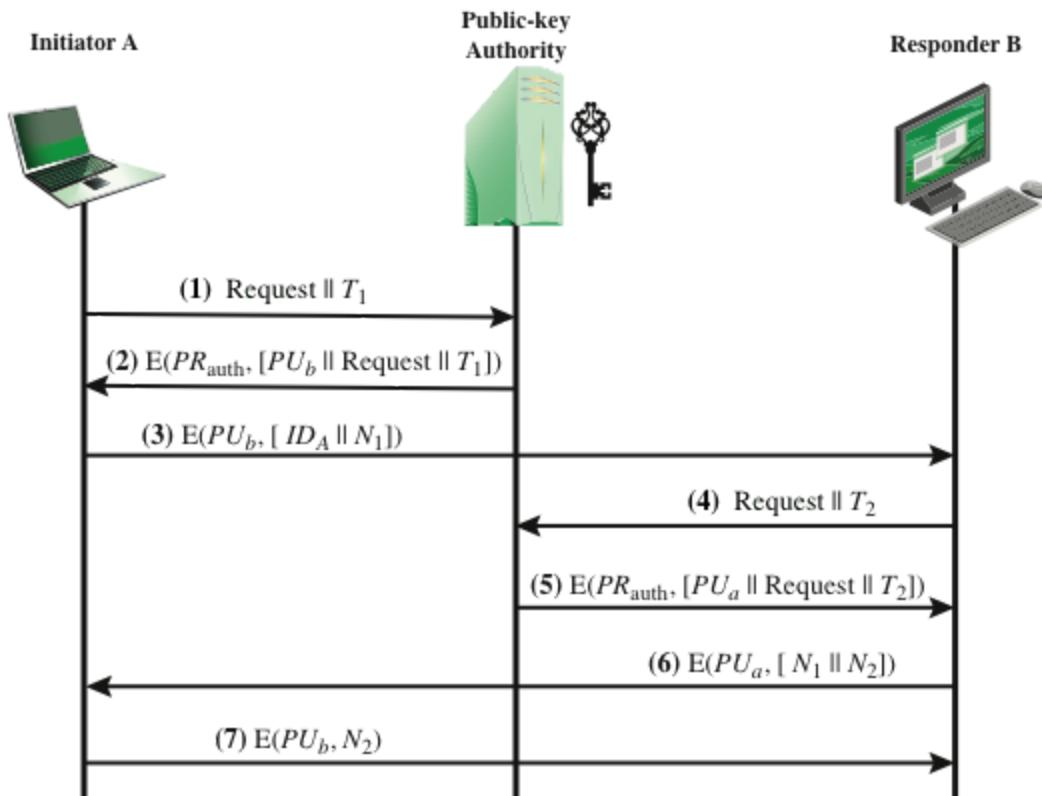
SIL765 - Assignment 2

Submitted By:
Mehak
2018MCS2143

PROBLEM STATEMENT

You are required to (a) build a PKDA, and (b) build clients that wish to confidentially send messages suitably encrypted with public key of receiver, but only after they know the other client's public key in a secure manner.

You are required to (a) build a, and (b) build clients that wish to send messages suitably encrypted with public key of receiver but of course only after they know each other's public key in a secure manner. Specifically use the scheme described below.



To do so, you will need to:

Assume:

4. that clients already (somehow) know the public key of the distribution authority, PKDA,
5. that the clients have their corresponding private keys with themselves, and
6. that PKDA has the public keys of all the clients,

Messages between PKDA and clients are encrypted using RSA algorithm and PKDA's private key,

Encrypted messages are sent/received between clients once they have each other client's public key, and finally

Find a way to generate and encode "current time" and "nonces".

Use the above to ensure client A can send 3 messages to B, viz Hi 1, Hi 2, and Hi 3. Client B in turn responds with Got-it 1, Got-it 2, etc. to messages received from A.

INTRODUCTION

A public key infrastructure (PKI) is a set of roles, policies, and procedures needed to create, manage, distribute, use, store & revoke digital certificates and manage public-key encryption. The purpose of a PKI is to facilitate the secure electronic transfer of information for a range of network activities such as e-commerce, internet banking and confidential email. It is required for activities where simple passwords are an inadequate authentication method and more rigorous proof is required to confirm the identity of the parties involved in the communication and to validate the information being transferred. PKI provides assurance of public key. It provides the identification of public keys and their distribution. PKDA provides the facility to share the public keys between clients securely.

SYSTEM SPECIFICATION

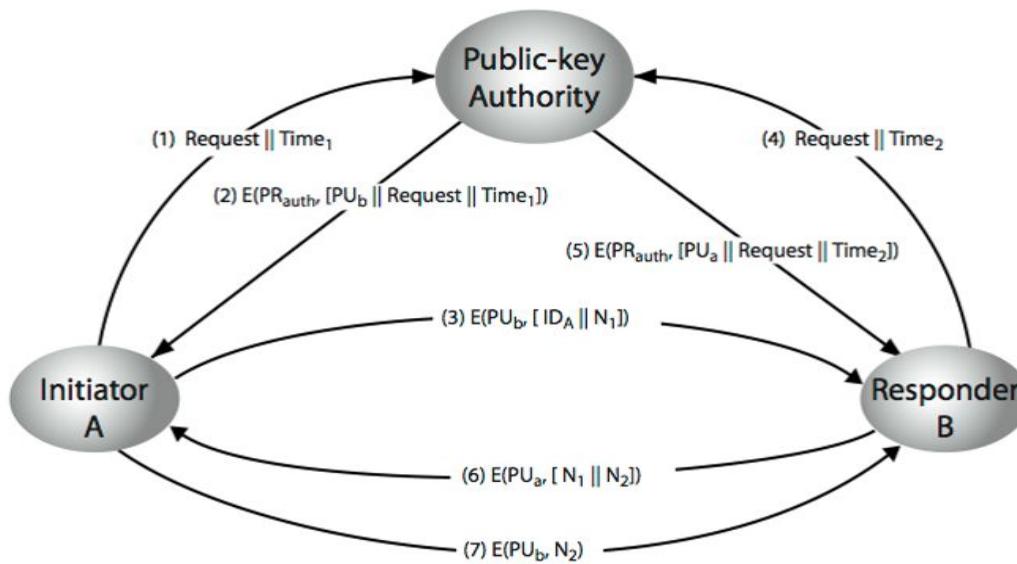
Language Used : Python

Assumptions :

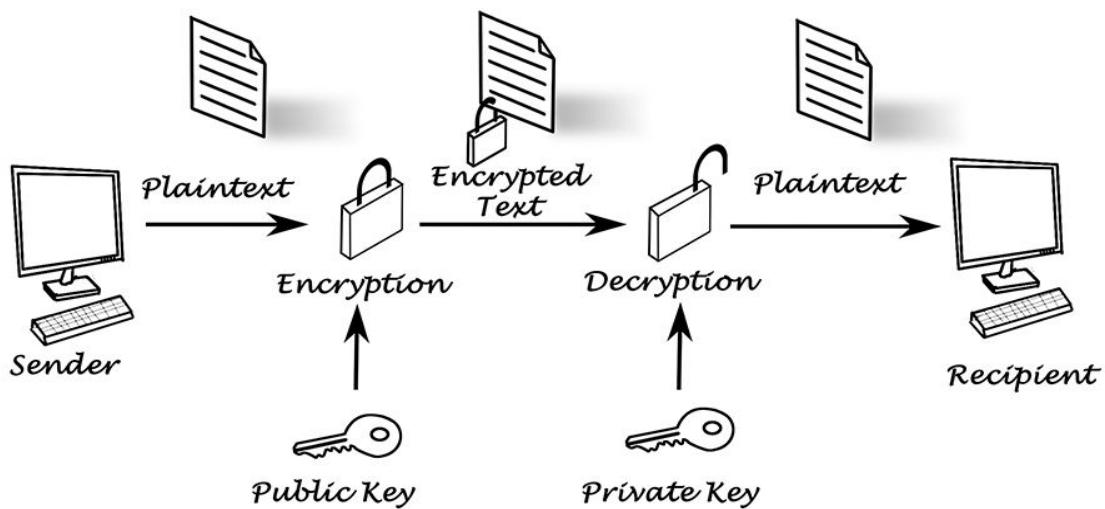
1. PKDA has public keys of Client A and Client B.
2. Clients A and B have their private keys and public key of PKDA.

PROCEDURE

1. Key Exchange (Communication between clients and PKDA)



2. Communication between clients using public key encryption



IMPLEMENTATION

1. Communication between PKDA and Client A

2. Communication between Server and Client B

```
New client connected: ('127.0.0.1', 38558)
Request received by client (encoded): Q2xpZW50IEIgfHwgQ2xpZW50IEEgfHwgMTU1MDg2MzM4NQ==

Request received by client (decoded): Client B || Client A || 1550863385
Response sent by PKDA (original): -----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDBq6fjm/L0mXD0plSw2qjYXsgb
nIyTNMw4XmuG9Q1xU/6faJrEA09ZVXUZtzLHoKbcON0PtNEptrNLVMJJtt+sfMz
vtDPhY8QcA0TFYgTIvohFLt0SoLSDuuWDA/d5egUQhKP6bPsuJUDJxyP0/8efYn
nYRacY00BK2/kdzmqQIDAQAB
-----END PUBLIC KEY-----Client B || Client A || 1550863385
Response sent by PKDA (encrypted): [REDACTED]
```

3. Communication between Client A and Client B

```
Enter q to quit
A: Hi 1
Message sent to B (original): Hi 1
Message sent to B (encrypted): Wv00I070080gD<00汎000000h00N0)0(0/010C0%0L0&w0m00:0Z000)C0I0.0000H00s0m00I0!~0I0
00>0"00J00Z00U0
W0000g080qIT0P0
Message received from B (encrypted): E00010!000R00A}0+10?0AL0~C|-000r0
000 00c.000$000su0W000x00d090I0
0~000H(k0000A)0
000T
,I0P:`th0r0pz0100nq
^[[?62;cMessage received from B (decrypted): 7df006318f704092ae45d9af5707c36c|e2bbb88df8b649219e1f6ff2647fabc7
A: Hi 2
Message sent to B (original): Hi 2
Message sent to B (encrypted): 00000000^0}Ac='%0lj0~00I0B;00V0
C0C000Z00@ P0E0_0000010x-#0{#60-W0000z000d\02R00?K8
Message received from B (encrypted): E00010!000R00A}0+10?0AL0~C|-000r0
000 00c.000$000su0W000x00d090I0
0~000H(k0000A)0
000T
,I0P:`th0r0pz0100nq
Message received from B (decrypted): 7df006318f704092ae45d9af5707c36c|e2bbb88df8b649219e1f6ff2647fabc7
A: q
Message sent to B (original): q
0q000%A^Y0010L0=0]0000m80D00I0;yP)0g000&r0'002ba00<000j00%0H(a8k0% s000b0u0D0d0s000amb00;0R#000
```

```
Message received from A (encrypted): Wv00I070080gD<00汎000000h00N0)0(0/010C0%0L0&w0m00:0Z000)C0I0.0000H00s0m00I0!~0I0
00>0"00J00Z00U0
W0000g080qIT0P0
Message received from A (decrypted): Hi 1
B: Got-it 1
Message sent to A (original): Got-it 1
Message sent to A (encrypted): *B1-H/t000E[(0[#-00M3T0Z0
0a0x# 000000}g0<xM-00c30090000R00]00K0{F0E0tk0*000}0 0HQ0000j0K~0H0y0~00:0}0
Message received from A (encrypted): 00000000^0}Ac='%0lj0~00I0B;00V0
C0C000Z00@ P0E0_0000010x-#0{#60-W0000z000d\02R00?K8
Message received from A (decrypted): Hi 2
B: Got-it 2
Message sent to A (original): Got-it 2
Message sent to A (encrypted): A&S00~,00( 000E00
R-000_00+`s000H100%004[0
000
U-00Y00y0A`00>0)0A#00000000
0q000%A^Y0010L0=0]0000m80D00I0;yP)0g000&r0'002ba00<000j00%0H(a8k0% s000b0u0D0d0s000amb00;0R#000
Message received from A (decrypted): q
```

