ASSIGNMENT 3

Project 1: RSA-based Public Key Distribution Authority(PKDA)

Contributors:

- Aekansh Kathunia(2021127)
- Paras Dhiman(2021482)

Abstract

This report presents the design and implementation of RSA-based Public Key Distribution Authority (PKDA). PKDA is responsible for managing and distributing the public keys to the intended recipients securely using RSA((Rivest-Shamir-Adleman). The system is implemented in Python where we demonstrate its functionality by building 2 clients that send requests to the PKDA for public-keys of themselves or that of other clients, and exchange messages with each other in a confidential manner, viz. suitably encrypted with public key of receiver, but only after they know the other client's public key in a secure manner.
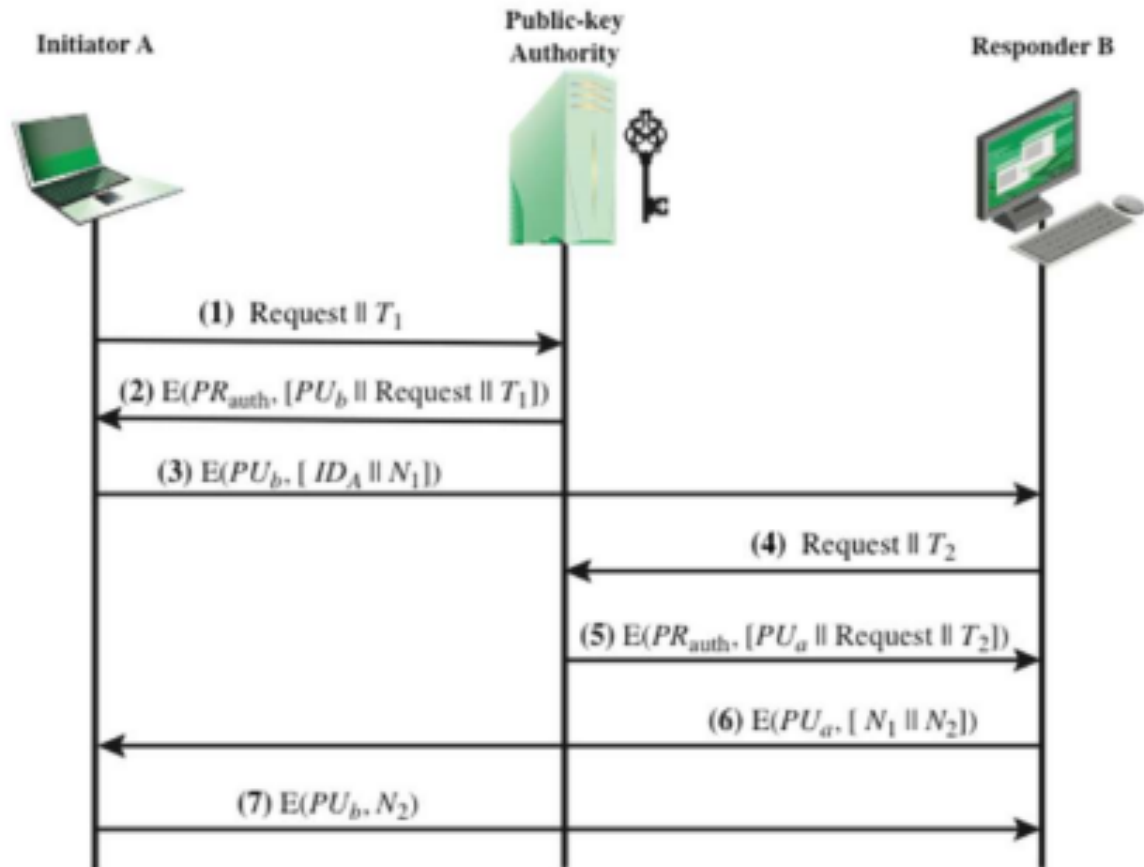
Introduction

RSA (Rivest-Shamir-Adleman) is a widely used asymmetric cryptographic algorithm that involves the use of public and private keys. In such a system:

- Each user or entity has a pair of cryptographic keys: a public key and a private key.
- The public key can be freely distributed and is used for encryption or verifying signatures.
- The private key is kept secret and is used for decryption or generating signatures.

In a scenario where RSA is employed for secure communication or data transfer, a Public Key Distribution Authority (PKDA) would be responsible for managing and distributing the public keys to the intended recipients securely. This ensures that each participant in the communication can obtain the necessary public keys to encrypt messages or verify signatures.

The PKDA would typically generate RSA key pairs, securely distribute the public keys, and maintain a trusted repository of public keys for participants to access when needed. It plays a crucial role in establishing the trustworthiness and integrity of the public key infrastructure (PKI) within a cryptographic system based on RSA encryption.

Implementation

Initiator A   Public-key Authority   Responder B

(1) Request $\| T_1$

(2) $E(PR_{auth}, [PU_b \| \text{Request} \| T_1])$

(3) $E(PU_b, [ID_A \| N_1])$

(4) Request $\| T_2$

(5) $E(PR_{auth}, [PU_a \| \text{Request} \| T_2])$

(6) $E(PU_a, [N_1 \| N_2])$

(7) $E(PU_b, N_2)$

Output

```
2024-03-31T17:43:12.241816
PKDA sends encrypted session key to Initiator A: [2253, 2483, 52, 3180, 1632, 2483]
2024-03-31T17:43:12.241816
PKDA sends encrypted session key to Responder B: [1598, 2812, 1430, 2173, 2323, 2812]
The nonce sent is:  d6af16ffd726e9dda3b3e220f7393384
Initiator A successfully verified Responder B's response (nonce2: d6af16ffd726e9dda3b3e220f7393384)
Responder B decrypts message from Initiator A: Hi1
Responder B sends response to Initiator A: [3129, 72, 3067, 707, 1165, 3067, 1632]
Initiator A decrypts response from Responder B: Got-it1
Responder B decrypts message from Initiator A: Hi2
Responder B sends response to Initiator A: [3129, 72, 3067, 707, 1165, 3067, 2146]
Initiator A decrypts response from Responder B: Got-it2
Responder B decrypts message from Initiator A: Hi3
Responder B sends response to Initiator A: [3129, 72, 3067, 707, 1165, 3067, 2483]
Initiator A decrypts response from Responder B: Got-it3
```