MINI PROJECT FOR COMPUTE AND NETWORK SECURITY

Here is the name of your system

Submitted to

School of Information, Computer and Communication Technology
Sirindhorn International Institute of Technology
Thammasat University

May 2020

by

StudentName1 StuID1
StudentName2 StuID2

Abstract

A short paragraph summarizing the provided features of the implemented system.

Contents

List of Figures

Chapter 1

Project Concept

In this chapter, you should write a good and concise introduction to your project.

1.1 Summary

What is done in the project? For example:

- Implement RSA algorithms (key generation and enc/decr)
- Use the implemented RSA algorithms and 3DES from the library xxx to provide two security services for email:
  - authentication: verify that received email really came from the sender.
  - confidentiality: protection from disclosure (no one but the receiver can successfully decrypt the encrypted email)

1.2 Typical Usage

How will the system be used in a typical scenario? What are some of the key features?

1.3 Main Challenges

What is the hardest and/or most time consuming part of the project?

Chapter 2

Project requirements

2.1 System description

Authentication of an email as having come from a specific sender follows these steps:

1. The sender creates a clear-text message.
2. The sender creates a SHA-1 message digest of the clear-text message.
3. The sender encrypts the SHA-1 message digest using the RSA asymmetric encryption algorithm with the sender's private key, producing a digital signature that is attached to the clear-text message.
4. A receiver uses the RSA asymmetric encryption algorithm with the sender's public key to decrypt the digital signature and recover the SHA-1 message digest.
5. The receiver generates a SHA-1 message digest from the clear-tex tmessage and compares the generated SHA-1 message digest with the decrypted SHA-1 message digest; if they match, then the message is accepted as authentic.

Confidentiality protection of a message follows these steps:

1. The sender generates a random 128-bit number to be used as a session shared secret key (SSSK) for this message only.
2. The sender encrypts the clear-text message, and appended digital signature,using a symmetric encryption algorithm, such as CAST-128, IDEA or 3DES, with the SSSK.
3. The sender then encrypts the SSSK using RSA with each recipient's public key(s) and then appends each uniquely encrypted copy of the SSSK to the black-text message.
4. Each receiver uses RSA with its private key to decrypt and recover their copy of the SSSK.

5. The decrypted SSSK is used to decrypt the black-text message thereby recovering the clear-text message.

2.2 Computational tasks

1. RSA algorithms
    (a) Key generation
        i. Prime number generation
        ii. Computing ...
    (b) Encryption
        i. Modular exponentiation
        ii. Encrypting a text/string
2.

2.3 Use cases

Chapter 3

Algorithm design and Implementation

3.1 Algorithm design

For each computational task, describe your algorithm to solve the task.

3.2 Implementation

Describe how you implemented the presented algorithms. You can write pseudo code or some code snippets.

Appendix A

Computing modular exponentiation

```
def decimalToBinary(m):
    return "{0:b}".format(int(m))


# Purpose: to compute a**m mod n efficiently
# where: a, m, n are integers

# Example: effModuloExp(2, 10, 15) returns 2**10%15 = 4


def effModuloExp(a, m, n):
    # convert m into binary
    bits = decimalToBinary(m)

    d = 1
    for bi in bits:
        d = d*d % n
        if bi != "0":
            d = (d*a) % n
    return d

# tests:
#print("(1000000000000**1000000)%100 = ", effModuloExp(1000000000000,


print("Try␣this␣expression␣directly?", (1000000000000**1000000)%100)
```