RSA Public-Key Encryption

The learning objective of this lab is to see how the RSA algorithm is utilized in real life with the help of a CLI tool that will generate key pairs, encrypt, decrypt and sign a particular input file.

We'll be working with real life scenarios and some fabricated scenarios during the course of this lab. We'll also complete the function to verify the signatures by ourselves. For the sake of everyone's convenience, this tool is written in Javascript/Typescript, because there is a large support for Cryptography modules in this language

This lab covers the following topics:

- Public Key Cryptography
- RSA Algorithm, Key generation and PEM files
- Encryption and Decryption using RSA
- Digital Signatures

Prerequisites

- Have NodeJS and NPM installed on Seed Ubuntu
- Here are the commands to install the Latest Version of NodeJS on Ubuntu

```
sudo apt update
sudo apt upgrade
curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash -
sudo apt install -y nodejs
```

Task 1

In this task, we will be setting up the tool that we will be using. First clone the repository into your root directory by running

```
git clone https://github.com/AnirudhRevanur/rsa-tool ~/rsa-tool
```

After running this, we have the `rsa-tool` CLI interface in our root directory, but it cannot be run yet. Run the following commands so that the tool can be run globally from the terminal.

```
cd ~/rsa-tool
chmod +x install.sh
sudo ./install.sh
```

After the completion of this script, you should be seeing this in the terminal

RSA Tool has been set up successfully. You can now use 'rsa-tool' command.

When this shows up, it means that the tool has been setup. Confirm it by running the following in the terminal

```
rsa-tool
```

It should output saying that you are using the tool incorrectly and show you the ways you can use the tool

Important note

If any of the source code is changed, then in order to apply the new changes that you have made, you have to run the following commands

```
cd ~/rsa-tool
sudo npm unlink -g rsa-tool
npm run build
sudo npm link
```

Upload a screenshot of the output that you get from running `rsa-tool`

Task 2

In Google Classrooms, two files have been uploaded. One being `privateKey.pem` and the other being `secret.enc`. The goal of this task is to utilize the Private Key and decrypt the secret file. In order to do this, run the following commands

```
rsa-tool --decrypt secret.enc decrypted.txt privateKey.pem
```

Upload the screenshot of the output, and the text in `decrypted.txt`

Task 3

Now we shall first generate a Public-Private Key Pair so that we can perform encryption and decryption. From the previous task we would have seen that the way we generate a key pair is by running

```
rsa-tool --generate privateKey.pem publicKey.pem
```

What this does is that it generates two files in the current working directory, one called `publicKey.pem` and the other being `privateKey.pem`. These are your respective Public and Private Keys

Go through the source code and find out the number of bytes that is being used by default. Change this parameter, reload the tool by running the commands above in **Important Note** section, then generate a new key pair

Find the difference in the length of the private key that is generated, and upload a screenshot of the two different Private Key files showing the difference in the length.

Change your key size back to 2048 bits in the source code and run `npm unlink` `npm link`

Task 4

Now we'll perform RSA Encryption and send it over to a friend to decrypt it. You and your partner generate your own pair of keys. Share your public keys with each other, **but not the private Keys**. Generate a file and encrypt it **with your partner's Public Key**, then send them the file.

So you should have encrypted a file and sent it to your friend, and you should have received an encrypted file from your friend.

In order to encrypt the file, run the following command

```
rsa-tool --generate privateKey.pem publicKey.pem
rsa-tool --encrypt input.txt output.enc /path/to/publickey/of/friend
```

In order to decrypt the file, use **your private key** by running the following commands:

```
rsa-tool --decrypt input.enc output.txt /path/to/your/privatekey
```

Make sure that your key pair does not change during this exchange

Upload screenshots of you encrypting the file, your friend decrypting the file that you encrypted and you decrypting the file that you received. Include the Name and SRN of the person you exchanged keys with as well. For the sake of convenience, name your files as `YourSRN_YourName`

Task 5

So far in all these tasks, we have used RSA to encrypt and decrypt the files. But this is not how it works in the real world. In the real world, a file is encrypted with a symmetric key algorithm, like AES, then that shared key is then encrypted with RSA. This is because RSA cannot handle encrypting large data.

Your task for this one is to find out *how* RSA was able to encrypt large files. There is also a file called `realAlgos.ts` that was uploaded to Google Classroom. Go through the other source code files and fill in the commented lines and explain what the two functions do.

Upload the fixed code, and your explanation of the functions

Submission

Make sure that you upload a PDF of all your findings and screenshots wherever you find anything interesting. You can also add your own comments and changes to the task as you see fit. You can also add some more functionalities as you see fit to the source code, but make sure that you document every finding and everything interesting you see about this Lab.

Along with the PDF, also upload a zip file with this particular file structure and naming convention. If there is any file missing or if the names are off, then that will lead to marks being deducted as this lab is being corrected automatically and not manually. Follow the file structure as follows

```
SRN_Lab_5/
        |
        | - Task_4/
                | - input.txt (Plain text that you have encrypted)
                | - SRN_publicKey.pem (This is your friend's public Key, put
their SRN)
                | - SRN_privateKey.pem (This your private key, put your SRN)
                | - output.txt (Text that you decrypted)
        | - Task_5/
                | - realAlgo.ts
```