Criptografia RSA

Bruno Felipe Firmino de Souza (brunofelip940@gmail.com)

Graduando em Engenharia Civil

Descrição e discursão do código em python elaborado:

A implementação do código de criptografia RSA se deu nos seguintes passos:
01) Foi definido a função Setup(), essa função implementada é capaz de analisar se as chaves públicas e privadas são válidas, caso os argumentos (n,e,d) sejam digitados da seguinte forma setup(n,e,d), onde 'n' representa o produto de dois números primos p e q (distintos entre si), 'e' representa o expoente de codificação e 'd' o expoente de decodificação. As equações: (i) $a^e = bmod(n)$, (ii) $b^d = amod(n)$ e (iii)$ed = 1mod((p-1)(q-1))$ formam a base do algoritmo da função setup(), onde a equação (i) permite criptografar um dado 'a' resultando em 'b', a equação (ii) permite descriptografar um dado 'b' retornando um dado 'a' e a equação (iii) permite obter o expoente de decodificação. Na teoria de criptografia RSA, os parâmetros (n,e) formam a chave pública e o parâmetro (d) forma a chave privada. No começo foi implementado uma pequena função que permite retornar um conjunto, dentro de um dado intervalo inferior e superior, dos números primos. A seguir foi importando o random para que a escolha de p e q dentro do conjunto dos números primos fosse aleatória. A função setup() foi dividida em três escolhas:
(i) A inserção dos três argumentos do setup(), onde essa parte pode verificar se os parâmetros n,e,d representam de fato uma chave pública e privada válidas ou não. Para uso dessa parte, basta digitar no console do IDE 'Cliente.setup(n,e,d)', se digitarmos Cliente.setup(611,7,79), será retornado ao usuário a seguinte informação ((611,7),(611,79)), caso contrário, Cliente.setup(610,7,79), será informado que a chave digitada não é válida. A base desse algoritmo é tomar os argumentos digitados, após isso, internamente foi colocado uma string a ser criptografada e descriptografada, cada letra dessa string é colocado em uma lista e após isso aplica-se a função ord(), onde convertemos o caractere em um número representativo 'a', assim aplicamos cada elemento dessa lista na equação (i), em que em linguagem python podemos traduzir ela como $b = (a^e)\%n$, onde encontramos o resto entre esses dois números, esse resto representa um caractere criptografado 'b'. Tomamos a lista dos caracteres criptografados e aplicamos cada elemento dessa lista na equação (ii), onde em python será $a = (b^d)\%n$, após isso, aplicamos cada elemento dessa lista na função chr(), em que permite transformar a (número representativo do caractere descriptografado) em caractere, concatenamos cada elemento da lista de caracteres (descriptografados) e obtemos uma string. Se a string inicial for igual a essa string, será impresso no console a informação ((n,e),(n,d)), do contrário, informará que a chave inserida não é válida.
(ii) Já para função setup(), basta digitar no console do IDE (Cliente.setup()), quando executada sem argumento, ela é capaz de retornar a chave aleatória ao usuário e é essa função que atua dentro da Classe para gerar uma chave que permite criptografar toda informação inserida no módulo cliente. O algoritmo nessa condição inicialmente gera um expoente aleatório 'e' (primo) e maior que 2, essa condição é necessária para dificutar que 'e' seja divisor comum de n, para garantir que 'e' e n não sejam múltiplos, foi condicionado ao código que '(p-1)' e '(q-1)' não serão múltiplos de 'e', por fim, 'p' e 'q' devem ser distintos, assim satisfazendo a equação (iii) e sendo possível encontrar um valor de 'd' adequado. Após o programa selecionar os valores de 'p' e 'q' que satisfaçam as condições necessárias, em python, a equação mencionada pode ser implementada nos seguintes passos: obtemos o valor de $(p-1)(q-1)$, como d>2, fazemos d=2, e para cada incremento unitário em 'd', será calculado o resto $(e*d)\%mod((p-1)(q-1)) == 1$, se o resto entre os valores da esquerda da igualdade for 1, o valor de d será estabelecido. Com isso determinamos (n,e,d), como é necessário retornar as chaves públicas e privadas dessa função, já que 'e' é aleatório, a saída se dá em uma tupla, em que o primeiro ele-

mento (tupla) é uma chave pública (n,e) e que o segundo elemento (tupla) é uma chave privada (n,d). Portanto, basta a classe cliente tomar uma variável recebendo o primeiro elemento e outra variável recebendo o segundo elemento.
(iii) Caso algum argumento não seja válido dentro da função setup(), será retornado que o argumento digitado é inválido ou algum outro aviso.
02) A função encrypt(S,chpul) recebe como argumentos uma string 'S' e uma tupla que representa a chave pública 'chpul'. A string S é transformada em uma lista, em que cada caractere de S é um elemento da lista 'vetS', a seguir, todos esses elementos são convertidos em números e inteiros representativos pelas funções int() e ord(), criptografamos essas informações pela equação (ii), por fim, a lista criptografada é transformada em uma string com espaçamentos.
IMPORTANTE: tentou-se converter os números criptografados em símbolos pela função chr, que representassem aqueles números, foi um sucesso, o que facilitaria bastante na questão da leitura e conversão desses elementos ao longo do algoritmo, entretanto, ocorreu um erro no python quando foi tentar escrever determinados caracteres no arquivo usuarios.txt, isto é, sempre criava um arquivo em branco e o algoritmo era interrompido, como não deu para analisar qual outro formato de arquivo ou configuração que impedisse esse tipo de problema, portanto, optou-se em obter strings onde números criptografados de cada caracter fossem separados por espaços.
03) A função decrypt(s,chpriv) recebe como argumentos uma string s criptografada e a chave privada chpriv, a necessidade da chave privada se faz em virtude dos parâmetros variáveis (n,d), o que facilita na primeira execução, quando se cria uma chave antes de existir um arquivo .txt com os dados da chave. A função pegará uma string s (as strings que retornam da função encrypt()), aplicando sobre a variável s o comando s.split(), com mesmo recebendo o resultado final, ocorre a exclusão dos espaços em branco e cria-se uma lista com os números criptografados, com a equação (ii), ocorre a descriptografia desses números, e com a função chr(), ocorre a conversão, transformando esses dados da lista com elementos de caracteres na string 'S' inicial, por fim ocorre a concatenação de toda a lista, resultando em 'S'.
04) O próximo passo é verificar se existe um arquivo usuarios.txt no diretório especificado no algoritmo, e caso esse arquivo exista, verificar se ele está vazio ou não. Essa verificação evita certos erros problemáticos (exemplo: o programa falhou por algum motivo, cria um arquivo em branco, o algoritmo valida a existência do arquivo, tenta ler as informações, como não tem nada, retorna um erro, o algoritmo só irá rodar nessa condição, se o arquivo em branco fosse excluído). Continuando, se o arquivo existir, os dados do usuario.txt serão lidos e jogados em uma lista, junto com as chaves públicas e privadas salvas, usadas para criptografar e descriptografar as informações de nome de usuários e senhas, no entanto se faz necessário tratar os dados extraídos do arquivo texto, já que retorna uma lista e cada elemento dela é reconhecida como string, nessa leitura, os elementos dessa lista vem com \n, onde se faz necessário a sua eliminação das strings da lista, realizado pelo comando 'string'.rstrip(\n). Em relação as chaves públicas e privadas, estes caracteres são removidos da string '(', ')', ',', após isso, existe espaçamento entre os números (n,e) e (n,d), é aplicado a função .split(), para transformar esses elementos em uma lista, depois disso, converte-se a lista em uma tupla e há substituição dessas informações nos índices 0 e 1 da lista de dados inicial gerada devido a leitura de usuarios.txt, e, por fim, pega essa lista de dados, pega-se cada elemento após o índice 1 dessa lista, e descriptografa todas as informações e substitui na lista nas posições correspondentes aos seus índices. Caso não exista o arquivo usuarios.txt ou se o arquivo estiver em branco, ele criará um novo ou substituirá o arquivo vazio, e gerará chaves de criptografia públicas e privadas novas, essas informações serão salvas na lista dados, onde o índice 0 corresponde a chave pública e o índice 1, a chave privada.
05) Após todo esse tratamento de dados, temos um módulo condicional com o while loop, onde ele só encerra caso o usuário condicione o programa a essa condição. Foram divididos no total de 4 possibilidades de entrada:
(i) Opção 0: essa opção faz com que o algoritmo seja encerrado. Além disso, ocorre toda a escrita de informações da lista dados para o arquivo usuarios.txt, onde as informações são criptografadas (usuários e senhas), pela função encrypt(), em função da chave pública criada ou já guardada no arquivo texto, é também salvo antes disso tudo, a chave pública, na primeira linha do arquivo texto e a chave privada na segunda linha do arquivo texto.

(ii) Opção 1: essa opção permite adicionar os usuários e suas respectivas senhas. É possível fornecer a quantidade de usuários a ser inseridos, caso entre acidentalmente nessa opção, foi colocado a opção 0 para não adicionar usuários e retornar ao menu de opções.
(iii) Opção 2: É o módulo onde permite que algum usuário existente na lista seja excluído. Caso algum nome exista, será retornado que o usuário foi apagado (junto com a senha). Caso não, será retornado que o usuário não existe.
(iv) Opção 3: É o módulo onde ocorre a procura de algum usuário dentro da lista dados e retorna se o usuário existe ou não naquele banco de dados.
Qualquer opção inválida será retornado no console do IDE. Foi configurado para o arquivo ser salvo em: C:\Users\Public.