Sistema Criptográfico RSA em Python 3

Carlos Henrique Torres de Andrade[1]

Resumo: Este artigo relata o processo de programação de um sistema criptográfico RSA em python 3. Detalha-se desde dos módulos que definem a criptografia como também os módulos de interação com o usuário e de utilização de arquivos como uma espécie de banco de dados simplificado.

I. Introdução

RSA é um algoritmo de criptografia de dados, o qual o seu nome se deve a três professores do Instituto de Tecnologia de Massachusetts (MIT), Ronald Rivest, Adi Shamir e Leonard Adleman, que inventaram este algoritmo que é considerado uma das mais bem sucedidas implementações de sistemas de chaves assimétricas, muito devido a sua segurança. Por fundamenta-se em teorias clássicas dos números, para um bom entendimento desse algoritmo, é necessário que o programador, ou qualquer pessoa que estude-o, tenha um conhecimento matemático prévio sobre números primos e aritmética modular.

II. Descrição do RSA do Sistema

Para representar a criptografia RSA foi criado uma Classe **Rsa** que possui como atributos as chaves necessária para o algoritmo e seus limties.

A. Geração das Chaves

Como descrito anteriormente, o RSA é um algoritmo de criptografia assimétrico, e por isso, ele deve ter um conjunto de chaves públicas e privadas. Para gerar essas chaves foi criado, em Python 3, um método **setup** para a Classe Rsa, que busca, primeiramente, gerar dois números primos aleatórios **p** e **q**, utilizou-se para isso o método **choice()** da biblioteca **secrets** para escolher aleatoriamente dois primos entre um intervalo especificado no código, no caso, entre 0 e 1000. Em seguida pode-se calcular **n** que é a multiplicação de **p** e **q**, e também, pode-se calcular a função totiente, utilizada para achar a última, que é dada por:

$$\phi(t) = (p-1) \times (q-1) \tag{1}$$

Além desses valores, também definiu-se o valor inteiro **e** como um valor fixo equivalente a 3. A escolha desse valor se deu por ser um valor considerado padrão nas implementações da comunidade. Com isso, calcula-se o valor da última chave **d**, para isso, utiliza-se uma implementação do Algoritmo de Euclides estendido. Ao fim, tem-se o conjunto de chaves definidos sendo **(e, n)** as chaves públicas e **(p, q, d)** as chaves privadas.

B. Encriptação

Com a chaves definidas, torna-se possível encriptar mensagens dentro da limitação recomendada (letra minúscula do alfabeto inglês e/ou números). Para isso, aplicou-se a seguinte equação para cada caractere da String a ser encriptada:

$$C = M^e \mod (n) \tag{2}$$

Utilizando essa equação para cada um dos caracteres tem-se a mensagem criptografada. É importante ressaltar a utilização do método **ord()** para que se possa realizar as operações da encriptação.

C. Decriptação

Para decifrar a mensagem codificada utiliza-se a seguinte equação:

$$M = C^d \mod (n) \tag{3}$$

Assim, consegue-se restaurar a mensagem que antes fora codificada.

III. Descrição dos Outros Módulos do Sistema

O sistema conta com outros importantes módulos que não o Rsa em si, mas que são importantes para a coesão do programa.

A. Usuário

Criou-se um classe **User** que envolve, junto com o **main**, a maior parte da interação do programa com o sistema, fazendo um controle de que está acessando os outros recursos do sistema. É a partir dele que o sistema apresenta as funcionalidades de cadastro, login, lista de usuários e deleção de contas.

B. Tratamento de Entrada de Dados

Criou-se duas classe: **InputTreatment** e **fileTreatment** que envolve todo tratamento de entrada de dados, seja por terminal ou arquivos. Graças a esses módulos, não só não há preocupação com usuário quebrando o sistema por um input errado, como também foi possível através da utilização de arquivo .txt criar um sistema de salvar os dados do sistema, tanto as chaves geradas como os usuários já cadastrados no sistema de forma codificada, sem a utilização de Banco de Dados. Assim, o sistema permite o usuário escolher no começo do programa se prefere usar as informações salvas ou usar o sistema do zero. Além disso, ao final do programa, ao encerrá-lo, é possível escolher gravar ou não a situação presente do sistema.

IV. Conclusão

O sistema apresentou um bom desempenho no geral com respostas instantâneas para a todos os comandos do sistema, com exceção das operações que envolvem a decriptação, o que já era esperado devido a ser a parte do programa mais pesada matematicamente falando. Além disso, ao fim desse projeto ficou bastante claro, mesmo que o sistema desenvolvido tenha tido um escopo muito menor do que os utilizados praticamente, o tamanho da robustez desses sistemas e o porque de ele ser amplamente utilizado em diversas áreas da sociedade.

References

[1] https://pt.wikipedia.org/wiki/RSA-(sistema-criptográfico)
[2] https://brilliant.org/wiki/extended-euclidean-algorithm/