Universidade de São Paulo
Instituto de Ciências Matemáticas e Computação de São Carlos
SSC747 - Engenharia de Segurança

Relatório de desenvolvimento do trabalho 1

Docente:
Dra. Kalinka Castelo Branco

Alunos:
Aulos Plautius Marino 7986409
Wesley Tiozzo 8077925

Maio de 2017

Sumário

1 ALGORITMO

O projeto se refere à implementação de um algoritmo de criptografia em linguagem python que permite criptografar e decriptografar uma mensagem de texto.
O algoritmo escolhido foi o RSA, o qual é um algoritmo de criptografia assimétrico. O RSA envolve um par de chaves: uma chave pública, a qual pode ser conhecida por todos e uma chave privada, que deve ser mantida em segredo. Toda mensagem cifrada utilizando uma chave pública só pode ser decriptografada utilizando a respectiva chave privada.
A ideia do RSA é baseada no fato de que é difícil de fatorar um valor inteiro grande. A chave pública consiste de dois números, onde um dos valores é a multiplicação de dois números primos. A chave privada também é obtida através desses dois números primos. A força da encriptação se baseia no valor da chave, então se esse valor é dobrado ou triplicado, a força da encriptação aumenta exponencialmente. Chaves RSA geralmente possuem tamanho de 1024 ou 2048 bits.

- Chave pública: n = P*Q, onde ‘P’ e ‘Q’ são números números primos distintos.

- Chave privada: d = ((k * phi(n) + 1) / e), onde ‘e’ é o expoente criptografador.

1.1 RESTRIÇÕES

Tempo e processamento, esta implementação aceita apenas arquivos em formato de texto como entrada. O arquivo de texto não pode apresentar caracteres de código ASCII 221 ou superior, devido a limitação da linguagem (ocorre overflow de variável quando são escolhidos números geradores grandes).

1.2 ESTRATÉGIA PARA TROCA DE CHAVE

No caso desta codificação do algoritmo, a chave está hardcoded no código, portanto qualquer pessoa com o programa pode encriptar ou decriptar a mensagem.

1.3 INSTRUÇÕES DE COMPILAÇÃO

O código foi implementado em Python, portanto sua compilação não é necessário. O código foi testado com sucesso em interpretadores python de versões 2.7.1 e 3.6.1. O código pode ser encontrado em: https://github.com/aulospl/EngSegCripto/tree/master/python

- Sintaxe de uso: `python main.py [e|d] [input_file]`
  - **e** para criptografar
  - **d** para decodificar

Exemplos de execução

- Execução do programa para encriptar uma mensagem:
  `python main.py e meu_texto.txt`

- Execução do programa para decriptografar uma mensagem:
  `python main.py d encrypted.txt`

2 SUGESTÕES DE MELHORIA

Como garantir confidencialidade, integridade, autenticidade e disponibilidade ?

- Confidencialidade: apenas o portador da chave privada pode decriptografar a mensagem, garantindo sigilo das informações e impedindo que pessoas não autorizadas tenham acesso ao conteúdo, ou seja, apenas o remetente e o destinatário possuem conhecimento a respeito da mensagem.
- Integridade: a mensagem não sofre perda durante o processo de criptografia e decodificação, portanto não há alteração do conteúdo das informações.
- Autenticidade: através do uso de assinatura digital, pois através dela é garantido não-repúdio a informação e garantia de identidade.
- Disponibilidade: garantindo que a informação estará disponível para acesso no momento desejado.

Qual seria a melhor abordagem para a disponibilização de acesso remoto à smart home, uso de serviços de nuvem ou a implementação de um servidor em casa?

- Toda mensagem entre o cliente e servidor deve ser encriptada com o algoritmo independente do serviço utilizado, porém através de um serviço de nuvem, se caso alguma falha ocorrer, a responsabilidade fica sobre a empresa que está provendo o serviço de nuvem.

3 REFERÊNCIAS

[1] “Python - reference,” https://docs.python.org/3/

[2] “Geeks for geeks - RSA”, www.geeksforgeeks.org/rsa-algorithm-cryptography/

[3] “Wikipedia - RSA”, https://en.wikipedia.org/wiki/RSA_(cryptosystem)