UNIVERSIDADE FEDERAL FLUMINENSE

REDES II PARA CIÊNCIA DA COMPUTAÇÃO
2016.2

RELATÓRIO

DESCRIÇÃO DO PROTOCOLO PROJETADO E DA CODIFICAÇÃO REALIZADA

Por Guilherme Magalhães e Juliana Jalloule

Professor Diego Passos

DESCRIÇÃO DO PROTOCOLO PROJETADO

[explicações de quais algoritmos criptográficos foram usados e para qual fim]

No início da conexão o servidor e cliente trocam suas chaves públicas. Essa troca é assumida como não sendo interferida e o cliente confia 100% na chave pública do servidor, assim como o servidor confia 100% na chave pública do cliente, para fins de simplificação.

O algoritmo criptográfico usado é o RSA, para garantir os três requisitos de segurança.

GARANTIA DOS REQUISITOS DE SEGURANÇA

CONFIDENCIALIDADE

Cliente aplica sua chave privada ao arquivo e concatena o resultado ao arquivo não-criptografado (assina digitalmente). Esse resultado, por sua vez, é criptografado pela chave pública do servidor, garantindo que somente o servidor conseguirá ver o real conteúdo da mensagem, o que garante confidencialidade.

AUTENTICIDADE E INTEGRIDADE

O servidor recebe esta mensagem e descriptografa usando sua chave privada. O resultado será o arquivo plano e o arquivo criptografado pela chave privada do cliente. O servidor aplica a chave pública do cliente a esta parte da mensagem e chega em um outro arquivo plano. O servidor compara os dois arquivos que possui, o que garante integridade e autenticidade. Isto porque se houvesse alguma mudança no arquivo durante o percurso, esta comparação falharia pois a assinatura digital não bateria com o texto plano, garantindo integridade. Da mesma forma, por estar encriptografado com a chave pública do servidor, somente ele poderá ver o conteúdo da mensagem e, por estar assinado digitalmente pelo cliente, somente ele poderia ter assinado daquela forma. Ou seja, a comparação só terá um resultado positivo, se o cliente tiver assinado a mensagem, garantindo autenticidade.

FUNCIONAMENTO DA IMPLEMENTAÇÃO

MAIN.JAVA

Coordena a inicialização do programa em um dos 4 modos dados como opção ao usuário. Estes 4 modos são explicados brevemente a seguir:

1 – Inicia servidor com chaves pública e privada setadas previamente.

2 – Inicia cliente com chaves pública e privada setadas previamene.

3 – Inicia servidor sem chaves pública e privada setadas previamente. As chaves são requisitadas ao usuário no início da execução do programa.

4 - Inicia cliente sem chaves pública e privada setadas previamente. As chaves são requisitadas ao usuário no início da execução do programa.

RSA.JAVA

É inicializado com chaves pública, privada e N de quem inicializou o módulo. Pode também recever a chave pública e o N de uma outra parte.

Dado que o módulo possui estas informações, a parte que o inicializa pode usar suas diversas opções disponíveis, que estão listadas a seguir:

- Encriptar array de bytes com sua chave pública.

- Encriptar array de bytes com sua chave privada.

- Encriptar array de bytes com chave pública de outra parte.

- Decriptar array de bytes com sua chave privada.

- Decriptar array de bytes com sua chave pública.

- Decriptar array de bytes com chave pública de outra parte.

TCPCLIENTE.JAVA

Responsável por receber inputs do usuário, listados a seguir:

- Endereço ao qual o arquivo será enviado.

- Nome do arquivo a ser enviado.

Realiza também troca de chave pública com o servidor de destino, carrega o conteúdo do arquivo em array de bytes e encripta este array de bytes utilizando o módulo RSA.

TCPSERVER.JAVA

Responsável por receber inputs do usuário, listados a seguir:

- Nome com o qual o arquivo recebido deverá ser gravado.

Também decripta o array de bytes recebido utilizando o módulo RSA para chegar ao arquivo plano.

PROCESSO DE COMPILAÇÃO DA IMPLEMENTAÇÃO

- Linguagem utilizada: Java.

É necessário possuir Java Runtime Environment 8+ e rodar o arquivo RedesII.java que está localizado dentro da pasta dist. Para executar, bastar utilizar no terminal o comando "java –jar *nome_do_arquivo*".

MODO DE UTILIZAÇÃO DA IMPLEMENTAÇÃO

O arquivo a ser enviado deve estar na mesma pasta que o programa. O arquivo que vai ser gerado pelo servidor com o nome solicitado vai ser salvo na mesma pasta do programa.

É necessário garantir que o servidor está rodando antes de rodar o cliente. Além disso, basta seguir os passos de interação indicados pelo programa durante sua execução.