DOCUMENTAÇÃO PROTOCOLO SSMS

Integrantes

Igor Inácio de Carvalho Silva - 725804
Vitoria Rodrigues Silva - 726598

Pré-requisitos

Para que a execução do programa ocorra com sucesso, os seguintes elementos são necessários:

- Java deve estar instalado e configurado de modo a permitir que o comando *java* seja reconhecido pelo Prompt de comando do Sistema Operacional Windows.
- Portanto, a variável *JAVA_HOME* deve estar configurada corretamente.

Execução do programa

O protocolo foi implementado na IDE Intellij, por isso os arquivos enviados estão em um padrão de projeto dessa IDE. Caso o acesso necessário seja apenas dos arquivos *.java*, eles estão no diretório "*SSMSProtocol-Igor-Vitoria/codigo-fonte/src"*.

Abaixo foram colocadas as instruções para executar os arquivos ***.jar***.
Após realizar o download do arquivo, faça a descompactação do arquivo no diretório desejado e em seguida:

- Abra duas janelas do *Prompt de Comando*. Para facilitar, chamaremos a primeira janela de **cmd-server** e a segunda janela de **cmd-client** (Figura 1).
- No **cmd-server**, digite o comando abaixo para iniciar a execução da classe Server (onde *<caminho do arquivo>* é o caminho absoluto do arquivo como, por exemplo, *"C:\Users\Maria\Dowloads\ssms\server.jar"*) e aperte *Enter*:

```
java -jar <caminho do arquivo>
```

- Neste momento, é esperado que a classe Server seja executada e gere a mensagem “Aguardando conexao”.
- Da mesma forma, execute o mesmo comando no **cmd-client** para executar a classe *Client*, mas agora com o caminho do arquivo ***Client.jar***.
- Neste momento, é esperado que o **cmd-client** gere um menu de opções para que os algoritmos de criptografia e seus parâmetros sejam selecionados.

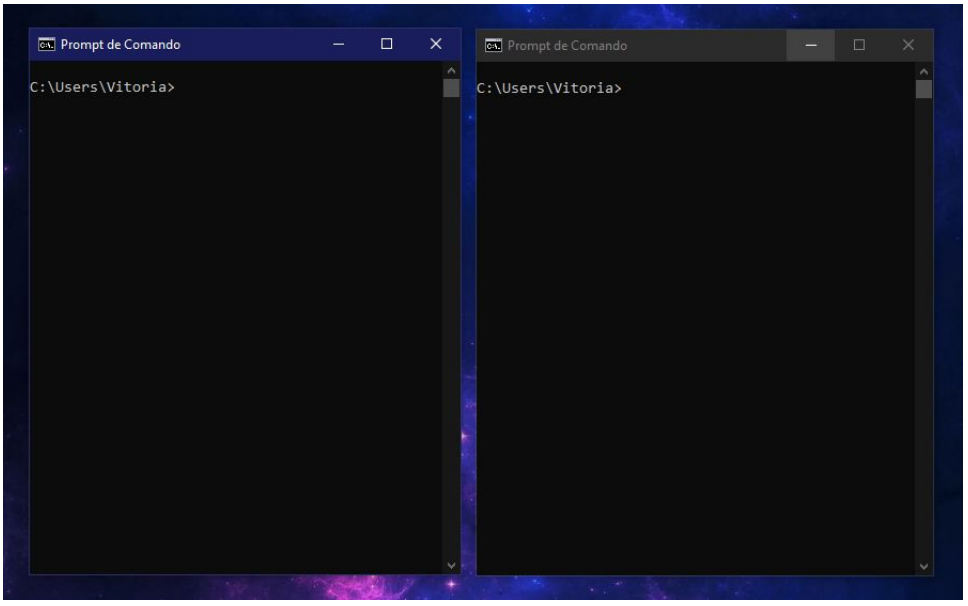

Figura 1 - cmd-server e cmd-client

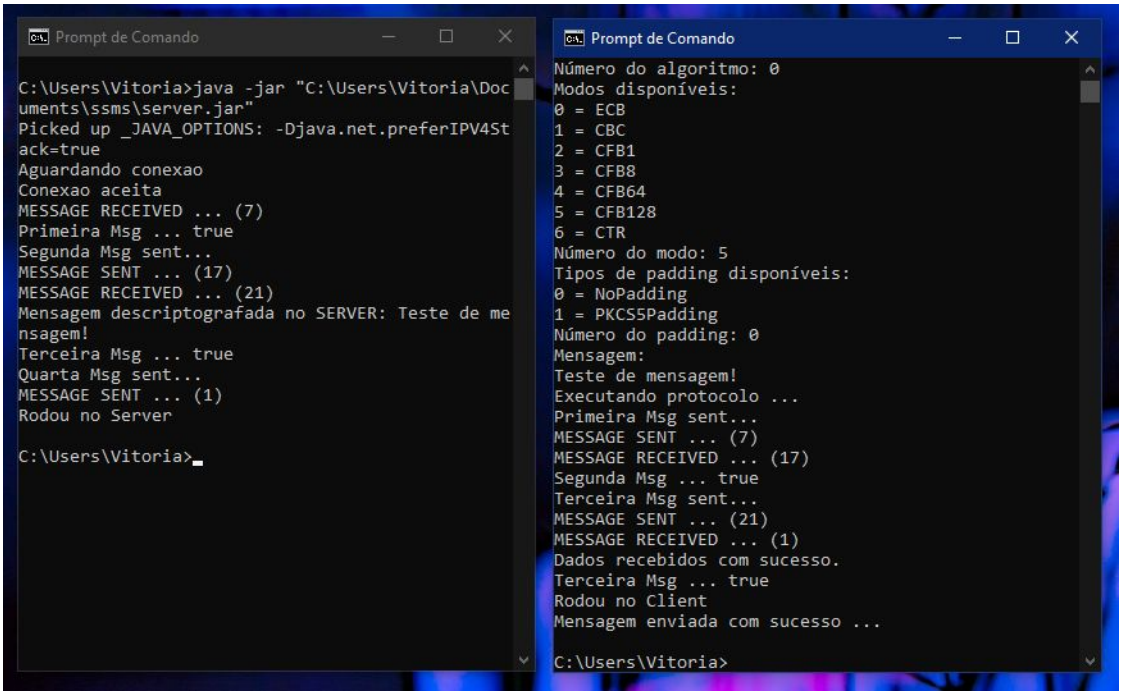

Figura 2 - exemplo de execução

Como exemplo de execução, selecionamos as opções abaixo:

- **0:** algoritmo AES128
- **5:** modo CFB128
- **0:** NoPadding
- **Teste de mensagem!:** mensagem a ser criptografada e enviada ao servidor de acordo com o vetor de inicialização retornado por ele na segunda mensagem (Par_conf, caso o servidor suporte os parâmetros anteriores).

O resultado dessa execução é o apresentado na Figura 2. Como podemos ver na janela da esquerda (**cmd-server**), o servidor descriptografou a mensagem enviada.

Fluxo de execução

As principais classes da implementação estão descritas abaixo :

- **RunProtocol:** classe responsável pela execução das etapas do protocolo, pois, para cada instância das classes *Client* e *Server*, ela realiza a alternância entre o modo de envio e de recebimento para ambas as classes. Portanto, quando o Server está no modo de recebimento, o *Client* está no modo de envio e vice-versa.
- **ProtocolSSMS:** classe responsável pela implementação do protocolo. Contém todos os métodos responsáveis por gerar as mensagens que serão enviadas e verificar as mensagens recebidas.
- **SecureSuite:** classe responsável por armazenar os dados relacionados aos algoritmos de criptografia (tanto pelo cliente, como pelo servidor). Os métodos disponibilizados nesta classe foram bastante úteis para a implementação dos métodos de criptografia.
- **Client:** responsável por instanciar os objetos necessários para a execução (das classes *ProtocolSSMS* e *RunProtocol*), bem como, requisitar a conexão com o server.
- **Server:** responsável por executar os métodos do lado do servidor e instanciar as classes necessárias para executar a troca de mensagens, além de aceitar a conexão por meio do socket.

Antes que a execução comece, o *Client* configura cada um dos parâmetros dos algoritmos na classe *SecureSuite* para que esses dados possam ser acessados posteriormente durante a execução dentro da classe *ProtocolSSMS*:

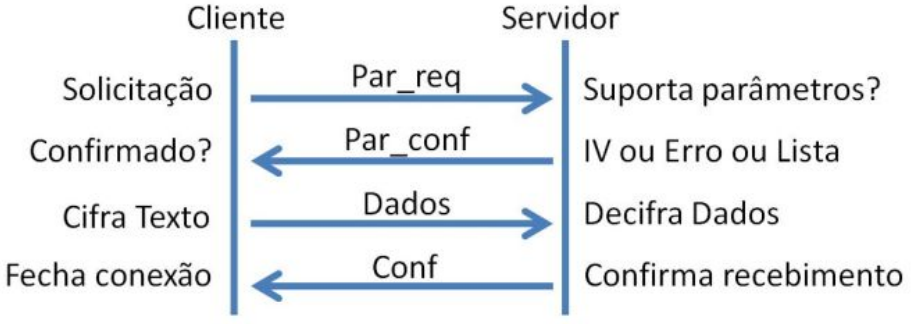

Figura 1 – Troca de Mensagens do Protocolo SMSS

Figura 3 - estrutura do protocolo

A execução inicia na classe *Server*, que espera a conexão ser estabelecida (solicitação da classe *Client*). Assim que a conexão é estabelecida, o *Client* gera a primeira mensagem (como descrito na Figura 3) por meio do método abaixo, onde cada um dos campos do cabeçalho é inicializado com os dados fornecidos pelo usuário no início da execução do *Client*.

```
private byte[ ] genParReq()
```

Em seguida, a mensagem é transmitida ao Server (através da instância de *RunProtocol*) e verificada através do método abaixo.

```
private boolean verifyParReq(byte[ ] receivedMsg)
```

Para facilitar a identificação dos métodos, eles foram nomeados de acordo com a estrutura do protocolo, por isso, os que são responsáveis pela geração das mensagens iniciam com “gen” e possuem o nome da mensagem sendo trocada:

- *genParReq*
- *genParConf*
- *genDadosMsg*
- *genConf*

E os que são responsáveis pelo recebimento e verificação das estruturas das mensagens (além dos códigos de erros) iniciam com “verify”:

- *verifyParReq*
- *verifyParConf*
- *verifyData*
- *verifyConf*

Após a verificação da primeira mensagem, o *Server* gera a segunda mensagem, o *Client* a verifica (por meio de *verifyParConf*) e, assim sucessivamente, até que o processo completo do protocolo esteja concluído.

Para os campos dos cabeçalhos com menos de 1 byte (tipo e código de erro, por exemplo), os bits foram armazenados por meio do tipo primitivo byte do java,

através da definição manual dos bits de cada campo (0b00000000 para tipo e código de erros iguais a zero, por exemplo). E os campos de cabeçalho com tamanho maior do que 1 byte foram manipulados por meio do método *arraycopy*.

Objetivos do projeto

O desenvolvimento do protocolo SSMS tem como principal objetivo a aplicação dos conhecimentos adquiridos na disciplina de Segurança e Auditoria de Sistemas, principalmente com relação às cifras de bloco e a aplicação dos algoritmos de criptografia para troca de mensagens dentro de uma arquitetura Cliente/Servidor.

Problemas encontrados na execução

Durante o primeiro teste dos arquivos ***.jar***, a execução gerou a *Exception "java.net.SocketException: Connection reset"*. A solução encontrada foi executada a partir dos seguintes passos:

- Abrir o Prompt de Comando
- Clicar com o botão direito do mouse na janela > *Propriedades*
- Em seguida, verificar se a opção *"Modo de Edição Rápida"* estava marcada, marcar essa opção caso não esteja selecionada e clicar em *OK*.
- Digitar *setx _JAVA_OPTIONS -Djava.net.preferIPV4Stack=true* na linha de comando e apertar *Enter*

Após a execução dessa configuração, os arquivos foram executados normalmente sem que a exceção fosse gerada.