Relatório da disciplina de Segurança de Redes - IMD0703

Abraão Dantas e Samuel Cavalcanti

Setembro 2019

1 Cifras Modernas: Simple DES e RC4

Implemente o algoritmo de criptografia S-DES apresentado em aula para cifrar/ decifrar um arquivo texto passado por parâmetro em linha de comando.

Implementar o protótipo do algoritmo RC4 para cifra qualquer texto usando uma chave de tamanho variável entre 1 a 256 bytes.

1.1 Solução apresentada

Para essa atividade foi necessário criar duas classes: SimpleDes e RC4. Para cifrar textos a partir desses algorítmos foi necessário criar uma interface por linha de comando chamado de **encrypt**. Ele é um script python que recebe 4 argumentos:

- Primeiro argumento você escolher se vai usar o Simple DES ou RC4, caso seja escolha o Simple DES digite: **des**, caso seja RC4 digite: **rc4**.
- Segundo argumento é o aquivo que contém a mensagem cifrada ou não.
- Terceiro argumento é a chave, os critérios da chave variam dependendo do algorítmo a ser utilizado caso seja o Simple Des a chave só pode ter um carácter, o RC4 pode ter até 256 caracteres , o resto será descartado.
- Quarto argumento indica se você vai encriptar a mensagem ou vai descriptografar a mensagem. Digite **e** para encriptar ou **d** para descriptografar.

Exemplo: utilizando o des para encriptar a mensagem:

```
$ cd Python_Chat/
# Python 3.7.4 caso esteja utilizando ubuntu
$ python encrypt.py des message.txt e e
```

Exemplo: utilizando o RC4 para encriptar a mensagem:

```
$ cd Python_Chat/
# Python 3.7.4 caso esteja utilizando ubuntu
$ python encrypt.py rc4 message.txt segredo e
```

Tando as classes quanto a interface pode ser encontrado na pasta **Python_Chat**. Nos arquivos: **s_des.py**, **rc4.py** e **encrypt.py**.

2 Chat seguro

Desenvolva uma aplicação para troca de mensagens de texto (estilo chat) entre você e seus colegas de maneira que seja possível trocar mensagens de texto entre pares utilizando criptografia com o S-DES e o RC4, desenvolvidos por você

- Assuma que a comunicação será feita pela porta 5354, com socket TCP
- Assuma também que ambos os pares já conhecem a chave de segurança e essa poderá ser modificada pelo usuário em tempo de execução da aplicação, bem como o algoritmo de criptografia utilizado (S-DES ou RC4)

2.1 Solução apresentada

Para essa Atividade foi necessário criar duas classes: Server e Client. A classe server é responsável por transmitir a mensagem para todos os clientes. A classe cliente é responsável por receber, enviar, cifrar e descifrar a mensagem.

Exemplo de utilização em localhost:

```
$ cd Python_Chat
# Python 3.7.4 caso esteja utilizando ubuntu
$ python socket_server.py
```

Abra mais dois terminais e execute os seguintes comandos.

```
$ cd Python_Chat
# Python 3.7.4 caso esteja utilizando ubuntu
$ python socket_client.py 127.0.0.1
```

O chat possui alguns comandos que são apresentados ao usuários assim que eles se conectam. Esses comandos são:

- \exit para sair do chat
- \crypt sdes "chave" para cifrar a conversa usando o simple DES, onde a chave deve ser escrita entre aspas duplas e permitindo apenas zeros e uns
- \crypt rc4 "segredo" para cifrar a conversa usando o RC4, onde a chave pode ter no máximo 256 caracteres

Exemplo de utilização do **Simple DES**: \crypt sdes "1010101010".
Observe que a chave só tem 10 dígitos, onde cada digito representa um byte, a chave do Simple DES tem o tamanho máximo de 10 bytes

Exemplo de utilização do **RC4**: \crypt rc4 "segredo".
No RC4 a chave ou frase que digitar entre as aspas duplas devera ser menor ou igual a 256 caracteres se a chave dada for maior que 256 caracteres o resto da frase ou da palavra será descartado

3 Troca de chaves usando Diffie-Hellman

Usando comunicação via socket, implemente a troca de chaves de sessão baseado no modelo de Diffie-Hellman.

Adicione esta funcionalidade ao seu chat seguro:

- Com esta atualização, seu programa deverá permitir configurar os valores de q e α; e trocar AUTOMATICAMENTE as chaves a serem usadas na sessão de criptografia simétrica

3.1 Solução apresentada

Para esta atividade foi necessário criar uma classe Diffie-Hellman e modificar o Client do nosso chat. A classe Diffie-Hellman realiza os cálculos de chave pública e de sessão dados um q e α. Já na classe cliente está precisou ser modificada pra realizar as trocas de chaves e alterar os comandos de criptografia/descriptografia (não é mais necessário digitar chaves já que ela são geradas pelo Diffie-Hellman).

Como executar o chat, Client (Foi o único com mudança):

```
$ cd Python_Chat
# Python 3.7.4 caso esteja utilizando ubuntu
# onde q e alpha devem ser valores que atendam ao proposto no
# algoritmo de Diffie-Hellman e deve ser de conhecimento de
# ambas as partes
$ python socket_client.py 127.0.0.1 q alpha
```

Os novos comandos do chat ficaram assim:

- \exit para sair do chat
- \dh_begin para iniciar a trocar de chaves entre os clientes, uma vez que um cliente solicite automaticamente as chaves serão trocadas entre todos.
- \crypt sdes para cifrar a conversa usando o simple DES
- \crypt rc4 para cifrar a conversa usando o RC4

4 Dificuldades encontradas

Uma das grandes dificuldades encontradas em todas as atividades foi a implementação do RC4, ao contrário dos outros algoritmos ele possui algumas operações aritméticas que não estavam claras no início e que por isso demandaram uma pesquisa mais profunda. A implementação da troca de chaves através da técnica de Diffie-Hellman parecia bem complicada à princípio mas a mesma foi mais simples do que o esperado, as mudanças no chat foram mínimas, de forma que, a refatoração do código pôde ser feita com uma certa velocidade. As demais atividades exigiram bastante planejamento e organização para que pudessem ser realizadas em tempo hábil.