

## Atividade 1 – Comunicação Segura

O código foi desenvolvido na linguagem Python (**versão 3.7**) usando o **Windows**. Para executá-lo, é preciso instalar as bibliotecas **cryptography** e **rsa**. Para que funcione corretamente, é preciso primeiro inicializar o arquivo **Server.py**. Depois, inicie o **Cliente1.py** e entre com o IP e com a porta que correspondem a **localhost** e **5535**, respectivamente. Depois, inicie **rapidamente** o **Cliente2.py** e insira os mesmos dados. Recomendo que os dados de ambos os clientes sejam preenchidos de forma simultânea para que o cliente 2 entre no server logo após o cliente 1 (em até 1 segundo). **É necessário que esse tempo seja respeitado. É possível apenas trocar mensagens entre dois usuários!**

No início do código de cada cliente, são geradas as chaves pública e privada de cada um. Também são obtidos os parâmetros **n** e **e** da chave pública. Quando os dois clientes se conectam, o cliente 1 começa enviando sua chave pública em 2 etapas. Primeiro ele envia o **n** e depois o **e**. Quando o cliente 2 recebe esses dados, ele monta a chave pública do cliente 1 e armazena em uma variável. Em seguida, o cliente 2 envia sua chave pública da mesma forma ao cliente 1. Depois, o cliente 1 gera uma chave simétrica aleatória, salva em um arquivo para uso próprio, a criptografa usando a chave pública do cliente 2 e a envia. Depois de alguns segundos, o cliente 1 cria o hash da chave simétrica, o assina com sua chave privada e envia a assinatura ao cliente 2. O cliente 2, ao receber a chave simétrica, a descriptografa usando sua chave privada e a armazena em um arquivo. Ao receber a assinatura, verifica usando a chave pública do cliente 1 se ela confere. Se sim, dispara uma mensagem de que a chave simétrica foi recebida de forma íntegra, a armazena em um arquivo e envia um “ok” ao cliente 1 para que ele entenda que a chave simétrica foi recebida corretamente. Quando o cliente 1 recebe o “ok”, ele a armazena em um arquivo chamado “ok1.txt”. Esse arquivo é extremamente importante porque é a partir da existência dele que o cliente 1 entende que o handshake foi finalizado e que já pode começar o envio de mensagens. O cliente 2 também checa se o handshake foi finalizado, porém, seu parâmetro é a existência do arquivo “keysimetrica2.key” que é o arquivo onde ele armazenou a chave simétrica recebida. Assim que os arquivos em questão são identificados, é exibido um “:” na tela de cada cliente que indica que a troca de mensagens já pode ser feita. A troca de mensagens pode ser iniciada por qualquer um dos clientes.

O processo de envio e de recebimento de mensagens é o mesmo para os dois clientes. Nesse exemplo, vamos considerar que o cliente 1 que iniciou a troca de mensagens. O cliente escreve sua mensagem e ela apenas pode ser enviada caso tenha até 255 caracteres, do contrário, é exibida uma mensagem de erro. Supondo que a mensagem enviada pelo cliente 1 esteja dentro dos parâmetros, sua mensagem é criptografada usando a chave simétrica e enviada. Alguns segundos depois, é feito o hash e a assinatura da mensagem usando sua chave privada. Por fim, a assinatura é enviada.

Quando o cliente 2 recebe a mensagem, ele a descriptografa usando a chave simétrica recebida e a salva em uma variável. Quando recebe a assinatura, ele confere sua assinatura usando a chave pública do cliente 1. Caso a assinatura seja conferida com sucesso, a mensagem junto com o nome do usuário que a enviou é exibida na tela. Caso não, é disparada uma mensagem que diz que a mensagem não foi recebida de forma íntegra e, assim, o programa é fechado.

Ao final do handshake, devem existir esses 3 arquivos na pasta na qual os clientes e o servidor foram executados, eles podem ser visualizados na Figura 1. É importante ressaltar que

eles devem ser **excluídos** a cada nova execução do chat, caso o contrário, ele não funcionará! Também é necessário reiniciar o **Server.py** nessa situação.




 keysimetrica2.key	23/08/2020 22:35	Arquivo KEY	1 KB
 ok1.txt	23/08/2020 22:35	Documento de Te...	1 KB
 keysimetrica.key	23/08/2020 22:35	Arquivo KEY	1 KB

Figura 1 – Arquivos gerados durante o handshake

A segurança do teste foi testada usando o Wireshark e esta foi aprovada. Os resultados podem ser vistos na Figura 2. A aparência do chat usado como o teste para o sniffer de rede pode ser vista na Figura 3. É importante ressaltar que o “ok” visto na Figura 2 refere-se ao “ok” que o cliente 2 envia ao cliente 1 quando recebe a chave simétrica de forma íntegra. Ele não corresponde a uma mensagem vazada!

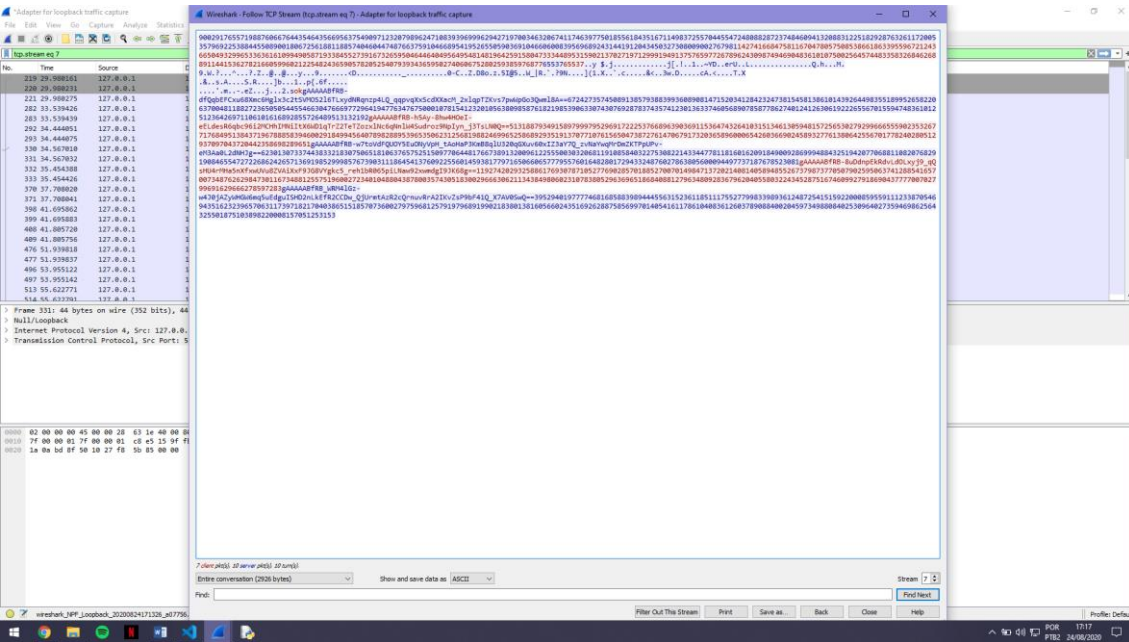


Figura 2 – Coletando pacotes enviados

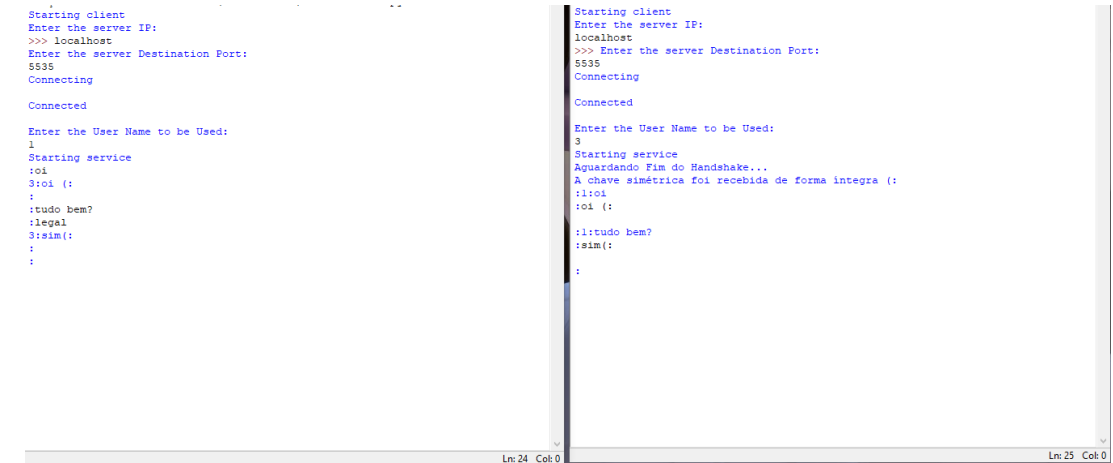


Figura 3 – Troca de mensagens feita pelos usuários “1” e “3”