UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
COLEGIADO DE CIÊNCIA DA COMPUTAÇÃO

Relatório de Aula Prática – Redes de Computadores

Título: Criptografia
Aluno: Vitor Mayorca Camargo Data: 13/02/2025

---

1. INTRODUÇÃO

A internet surgiu na década de 1960 como ARPANET, inicialmente usada para comunicação militar e acadêmica nos EUA (LEINER, *et al*. 2009). Em 1989, a criação da World Wide Web democratizou o acesso e compartilhamento de informações por meio de navegadores web, permitindo que pessoas comuns usassem a internet de forma fácil e rápida (MONTEIRO, 2001).

As informações transmitidas na internet seguem protocolos rigorosos, como o TCP e o IP, que surgiram da necessidade de estabelecer regras e padrões para unificar as diferentes redes e máquinas que formam a internet (CORONA, 2004). O protocolo TCP (Transmission Control Protocol) foi projetado para fornecer uma comunicação confiável entre dois computadores, reduzindo ou aumentando a taxa de transmissão de dados de forma dinâmica, com o fim de evitar perda de dados. (CAMPISTA, *et al*. 2010).

Já o protocolo IPv4 (Internet Protocol) define a base para a transação, tráfego e reconhecimento de dados em uma rede. Ele é responsável por definir o "Endereço IP", que é um número único dado a cada máquina ou "host" na rede. Esse endereço é composto por quatro números (de 0 a 255) separados por pontos (de 000.000.000.000 a 255.255.255.255), e permite que uma máquina seja identificada e se comunique com outras na rede (CORONA, *et al*. 2004). Os autores também explicam que o protocolo TCP/IP divide os dados em pequenas porções para transferência na Internet. O protocolo IP organiza o envio e recebimento desses pacotes, enquanto o protocolo TCP se encarrega de dividir, ordenar e garantir que o fluxo de dados ocorra corretamente.

Duas versões do protocolo IP são usadas como padrão na internet. O IPv4 especifica as capacidades e protocolos básicos que toda máquina deve seguir, e usa um endereço de 32 bits (TANENBAUM, 2003). Já o IPv6, mais avançado, possui uma maior segurança e melhor comunicação entre as redes, fornecendo um endereço de 128 bits (CHANDRA, KATHING, KUMAR. 2013).

Com a popularização da internet, se tornou cada vez mais necessário garantir a segurança dos dados transmitidos na internet, contra ataques, manipulações, e acessos não autorizados. Uma das técnicas mais comuns de proteção de dados digitais é com o uso de *criptografia*. Kurose e Ross (2021) explicam que a criptografia resume-se ao processo de “disfarçar” informações, de forma que usuários indesejados não consigam entender os dados do pacote. Isso normalmente é feito com o uso de “chaves criptográficas”. Um algoritmo de criptografia e descriptografia vai usar essa chave, que só o remetente e o destinatário sabem, para poder “embaralhar” os dados transmitidos.

No geral, pode se dizer que existem duas formas de criptografar dados: pela criptografia simétrica, e pela criptografia assimétrica. Oliveira (2012) explica que a criptografia simétrica é mais antiga, e nela a chave é a mesma para ambas as partes, e deve permanecer um segredo. É um método mais simples e rápido. Porém, inicialmente a chave precisa ser previamente compartilhada entre origem e destino e, se ela for interceptada nesse momento, toda a comunicação estará comprometida. Por fim, a criptografia assimétrica não garante os princípios da autenticidade e

não-repudiação.

Um exemplo de algoritmo que usa chaves simétricas é o AES (Advanced Encryption Standard), uma cifra de bloco padrão do governo dos EUA desde 2003, que usa blocos de 128 bits e chaves de 128, 192 ou 256 bits, e é rápido, leve e amplamente usado. Também vale citar as criptografias da família Salsa20. Ela é consistentemente mais rápida que o AES, sendo mais indicada para aplicações de uso geral (BERNSTEIN, 2008).

Na comunicação com criptografia assimétrica, cada parte utiliza duas chaves complementares: uma pública e uma privada. A chave pública é disponibilizada para qualquer um que queira fazer parte da comunicação, mas a privada ficará em poder apenas do titular. A chave pública criptografa, e a privada descriptografa. Esse método permite que qualquer um envie uma mensagem secreta, a um usuário, que será o único capaz de descriptografá-la com a sua chave privada. Porém, o algoritmo é difícil de implementar, além de ser computacionalmente caro.

Um exemplo de algoritmo de criptografia assimétrica é o RSA, que foi criado em 1977. Ele baseia-se na dificuldade de fatorar números grandes, então a sua segurança depende do tamanho da chave. No Brasil, a ICP-Brasil usa chaves de 4.096 bits desde 2012 (OLIVEIRA, 2012).

Para a implementação de clientes e servidores web, a linguagem Python 3 se destaca devido à sua simplicidade, fácil uso, e grande variedade de bibliotecas que ajudam no processo. Além disso, a biblioteca PyCryptodome também oferece inúmeras ferramentas que facilitam a implementação de diferentes algoritmos de criptografia.

2. OBJETIVOS

A atividade prática realizada no dia 13/02/2025 teve como objetivo implementar os algoritmos de criptografia Salsa20, AES, e RSA. O professor também forneceu 3 códigos, que então foram modificados com o fim de implementar funcionalidades novas. Foi utilizada a linguagem de programação Python 3, com a ajuda da biblioteca PyCryptoDome.

3. MATERIAL UTILIZADO

Os programas e testes foram feitos num notebook da marca DELL, modelo Vostro 3520, com um CPU Intel Core i7-1255U 1.70 GHz, 16Gb de memória RAM DDR4, placa de vídeo integrada Intel Iris Xe Graphics, placa de vídeo dedicada NVIDIA GeForce MX550, adaptador de rede modelo Intel(R) Wi-Fi 6 AX201 160MHz, unidade de disco SSD NVMe ADATA 512Gb.

Dentro dele, foi executado o sistema operacional Windows 11 Home, versão 10.0.26100. Mais especificações da máquina estão explícitas na figura 1. Na máquina foi instalado o ambiente virtual do Python, diretamente do website oficial, e o PyCryptoDome, com a ajuda do instalador de pacotes do python, pip. Os testes foram feitos durante a aula prática do dia 13 de fevereiro de 2025.

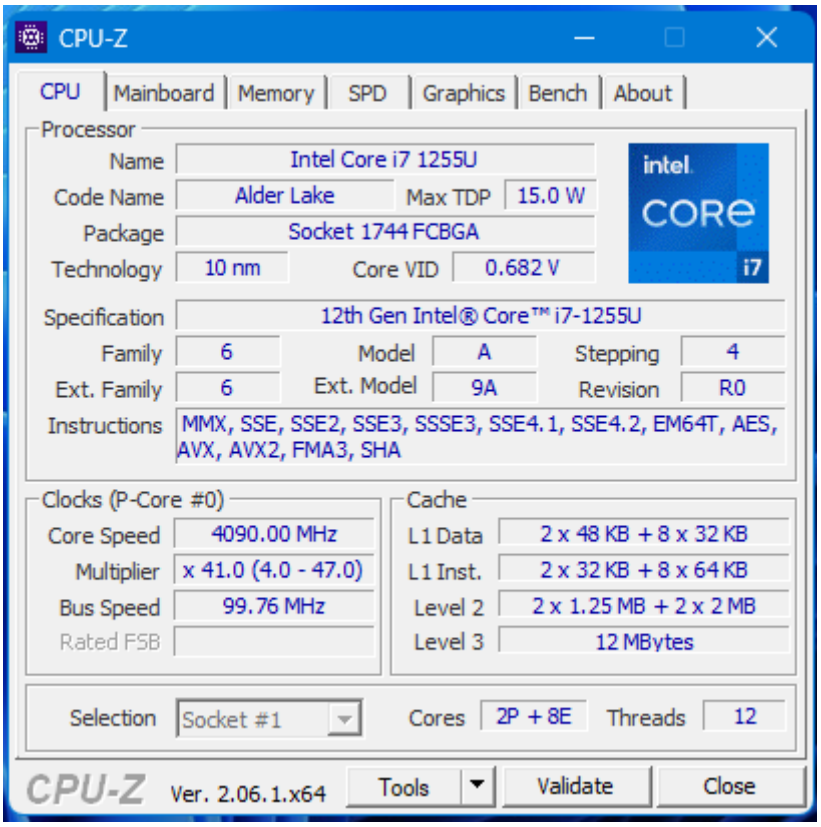

Figura 1: Especificações do sistema segundo o software *CPU-Z*.

4. METODOLOGIA

4.1 Cifra de Fluxo Salsa20:

Inicialmente, foi implementado o algoritmo Salsa20, de acordo com o sugerido na documentação oficial do PyCryptodome (https://pycryptodome.readthedocs.io/en/latest/src/cipher/salsa20.html). Depois, o código foi testado para diferentes inputs.

4.2 Cifra de Bloco AES:

Foi implementado o algoritmo AES, de acordo com o sugerido na documentação oficial do PyCryptodome (https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html). Depois, o código foi testado para diferentes inputs.

4.3 Atividade 1:

Durante a prática foram disponibilziados os arquivos aes_cliente e aes_servidor. Eles foram executados, com o fim de testar na prática o funcionamento de uma troca de pacotes encriptados por AES.

4.4 Atividade 2:

Com base nas implementações do algoritmo RSA disponibilizados pelo PyCryptodome (https://pycryptodome.readthedocs.io/en/latest/src/examples.html#generate-public-key-and-private-key), que realiza a criptografia com a chave pública e a decriptografia com a provada, foi feita uma modificação no código com o fim de fazer com que a chave privada seja responsável pela criptografia, e a pública a descriptografia, garantindo autenticidade das mensagens.

5. RESULTADOS E DISCUSSÃO

5.1 Cifra de Fluxo Salsa20:

Com base na documentação oficial do PyCryptodome, foi criado um algoritmo básico que recebe uma mensagem e uma chave como input, e depois realiza o processo de criptografia e decriptografia na mensagem. Observa-se que o algoritmo funciona corretamente, conforme as imagens 2 e 3.

```
from Crypto.Cipher import Salsa20

#Encriptando
print("Insira a mensagem a ser codificada")
plaintext = input()
plaintext = plaintext.encode()

print("Insira a chave (Deve conter 32 caracteres)")
secret = input()
secret = secret.encode()
#secret = b'*Thirty-two byte (256 bits) key*'
cipher = Salsa20.new(key=secret)
msg = cipher.nonce + cipher.encrypt(plaintext)

print("Encriptado: ", msg)

#Desencriptando
secret = secret
msg_nonce = msg[:8]
ciphertext = msg[8:]
cipher = Salsa20.new(key=secret, nonce=msg_nonce)
plaintext = cipher.decrypt(ciphertext)
print("Decriptado: ", plaintext.decode())
```

Figura 2: Algoritmo Salsa20 com a ajuda do PyCryptodome.

```
Insira a mensagem a ser codificada
Teste 123
Insira a chave (Deve conter 32 caracteres)
12345678901234567890123456789012
Encriptado:  b'Xz\xd3\x92\xddZ|\x9f\xcc9\xcf\xa3\xde\x85\x11\xe8\x1c'
Decriptado:  Teste 123
```

Figura 3: Teste rápido do algoritmo Salsa20.

5.2 Cifra de Bloco AES:

De forma similar ao algoritmo Salsa20, o AES também foi implementado conforme a documentação do PyCryptodome. O código recebe uma mensagem e uma chave como input e, para poder decriptografar a mensagem, é necessário usar a mesma chave usada na criptografia.

```python
from Crypto.Cipher import AES

# Criptografia
print("Insira uma mensagem a ser criptografada (Apenas caracteres ASCII)")
data = input()
data = data.encode()

print("Insira uma chave (Deve conter 16 caracteres ASCII)")
key = input()
key = key.encode()

cipher = AES.new(key, AES.MODE_EAX)

nonce = cipher.nonce
ciphertext, tag = cipher.encrypt_and_digest(data)

print("Mensagem criptografada: ", ciphertext)
print("Tag?: ", tag)

# Decriptografia
print("\n-------------------------\nHora de decriptografar!")
print("Insira a chave de decriptografia (Deve conter 16 caracteres ASCII)")
key = input()
key = key.encode()

cipher = AES.new(key, AES.MODE_EAX, nonce=nonce)
plaintext = cipher.decrypt(ciphertext)
try:
    cipher.verify(tag)
    print("The message is authentic:", plaintext)
except ValueError:
    print("Key incorrect or message corrupted")
```

Figura 4: Algoritmo AES com a ajuda do PyCryptodome.

```
Insira uma mensagem a ser criptografada (Apenas caracteres ASCII)
Teste 1 2 3
Insira uma chave (Deve conter 16 caracteres ASCII)
1234567890123457
Mensagem criptografada:  b"\xa8Qn'O\x19\xe3\xc8a\x97\xba"
Tag?:  b'\xaa\x953\xa5E\x03\x99\x18j\x80NgW\x9f\xaa\x95'

-------------------------
Hora de decriptografar!
Insira a chave de decriptografia (Deve conter 16 caracteres ASCII)
1234567890123457
The message is authentic: b'Teste 1 2 3'
```

Figura 5: Teste rápido do algoritmo AES.

5.3 Atividade 1:

Ao tentar executar o cliente, percebeu-se que a chave era “minha_chave_secreta1234”. Isso é um problema, pois a chave deveria ter 16 bytes (16 caracteres). Devido a isso, o algoritmo não foi executado corretamente. Para resolver esse problema, foi necessário mudar a chave para uma de 16 bytes, “chave_secreta123” foi a chave escolhida. Após essa pequena alteração, o cliente conseguiu se conectar corretamente ao servidor, conforme a figura 6.

```
Aguardando conexão...
Conectado a ('127.0.0.1', 55255)
Mensagem recebida: Olá, servidor! Aqui é o cliente!
```

Figura 6: Execução correta do aes_servidor.

5.4 Atividade 2:

Com base na implementação já disponível, foram feitas duas funções simples: uma de descriptografar com a chave pública, e outra de criptografar com a chave privada (Figura 7). As novas funções então foram testadas (Figura 8), e o resultado foi um sucesso (Figura 9). Percebe-se que a mensagem criptografada com a chave privada possui um tamanho muito menor que a mensagem criptografada com a chave pública.

```python
# Função de descriptografia com a chave pública
def descriptografar_com_publica(mensagem, chave_publica):
    cipher = PKCS1_OAEP.new(chave_publica)
    return cipher.decrypt(base64.b64decode(mensagem)).decode()

# Função de criptografia com a chave privada
def criptografar_com_privada(mensagem, chave_privada):
    cipher = PKCS1_OAEP.new(chave_privada)
    return base64.b64encode(cipher.encrypt(mensagem.encode())).decode()
```

Figura 7: Funções implementadas na Atividade 2.

```python
# Teste 2
mensagem_original = "Mensagem secreta Criptografada com Privada"
criptografada = criptografar_com_privada(mensagem_original, chave_publica)
print(f"Criptografada:\n{criptografada}")
descriptografada = descriptografar_com_publica(criptografada, chave_privada)
print(f"Descriptografada:\n{descriptografada}")
```

Figura 8: Chamadas das funções implementadas na Figura 7.

```
Criptografada:
tIBWuJjNnmSDaC+UvrDKU7mr5Gvrk6/QwVbfYiGnupZv+G/YPRIsKgW
wtpTALq0qu4+TAbjdIIMjRmgMJsT1P14U3nijdopi1mAiXrafDx5uK4
EoUw37wsKqWBFICcgS7kEFeY4O+Iuzvh7hiHRWIzCYwU4fdpaFXqM9j
YCP1buWzdIKknVAJy861eUvgbM9U11A6S89ZIiUbC6h1dFzEJMqxFet
wuHXtk+PdEX+T4yB9ZpkTL+/AihXXb1oSQddVyo+C4MhqWDXp5Vu/lg
00YpXxerzjeDRAMwnmcSqjbm4WlKioL++4wkETuenyJiGC7RXI3PBu7
U6yh7MQhf7Eg==
Descriptografada:
Mensagem secreta Criptografada com Privada
```

Figura 9: Resultado das execuções da Atividade 2.

6. CONCLUSÕES

A aula prática proporcionou uma boa compreensão sobre a parte prática da implementação e funcionamento dos algoritmos de criptografia, e ajudou na visualização de como uma chave e uma mensagem criptografada realmente se parecem. O uso do pacote PyCryptodome foi de imensa ajuda no processo, pois as suas funções facilitam muito o processo, e oferecem ao usuário uma interface simples e intuitiva.

Conclui-se, portanto, que a prática foi bem-sucedida em atingir seus objetivos, consolidando o entendimento sobre algoritmos de criptografia, e mostrando que a linguagem Python com o PyCryptodome de fato são uma ótima dupla quando o assunto é implementar sistemas de criptografia de forma rápida e eficiente.

BIBLIOGRAFIA

BERNSTEIN, Daniel J. The Salsa20 family of stream ciphers. In: **New stream cipher designs: the eSTREAM finalists**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 84-97.

CAMPISTA, Miguel Elias M. et al. Interconexão de Redes na Internet do Futuro: Desafios e Soluções. **Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC**, v. 2010, p. 47-101, 2010.

CHANDRA, Deka Ganesh; KATHING, Margaret; KUMAR, Das Prashanta. A comparative study on IPv4 and IPv6. In: **2013 International Conference on Communication Systems and Network Technologies**. IEEE, 2013. p. 286-289.

CORONA, Adrián Estrada. *et al*. **Protocolos TCP/IP de internet**. 2004.

GOYAL, Piyush; GOYAL, Anurag. Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark. In: **2017 9th International Conference on Computational Intelligence and Communication Networks (CICN)**. IEEE, 2017. p. 77-81.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **IEEE 802.11:** Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. New York: IEEE, 1997.

KUROSE, James F.; ROSS, Keith W. **Redes de Computadores e a Internet: Uma abordagem top-down**. Trad. 8 ed. Sao Paulo: Francisco Araújo da Costa, 2021.

LEINER, Barry M. et al. **A brief history of the Internet**. ACM SIGCOMM computer communication review, v. 39, n. 5, p. 22-31, 2009.

MONTEIRO, Luís. **A internet como meio de comunicação: possibilidades e limitações**. In: Congresso Brasileiro de Comunicação. sn, 2001.

OLIVEIRA, Ronielton Rezende. **Criptografia simétrica e assimétrica-os principais algoritmos de cifragem**. Segurança Digital [Revista online], v. 31, p. 11-15, 2012.

ROSS, Julio. **Redes de computadores**. Julio Ross, 2008.

TANENBAUM, Andrew S. **Redes de Computadores**, 7ª Edição, Editora Campus, Rio de Janeiro – RJ, 2003.