Relatório do Trabalho Prático 2 – Simulador de HTTPS

Aluno: Gabriel Frigo Petuco
Disciplina: Segurança de Sistemas

1. Introdução

Este trabalho prático tem como objetivo simular parte do protocolo HTTPS, abrangendo duas etapas principais: a geração de uma chave segura utilizando o protocolo Diffie-Hellman e a troca de mensagens criptografadas de forma segura. A solução implementada visa demonstrar a aplicação desses conceitos utilizando criptografia simétrica AES no modo de operação CBC com padding.

O protocolo HTTPS (HyperText Transfer Protocol Secure) utiliza criptografia para garantir a segurança e privacidade na comunicação entre o cliente e o servidor. A segurança da comunicação é baseada em técnicas de troca de chaves e criptografia, o que é simulado neste trabalho. A solução envolve a utilização do algoritmo Diffie-Hellman para gerar uma chave compartilhada e a criptografia das mensagens com AES.

2. Etapa 1 – Geração da Chave com Diffie-Hellman

O protocolo Diffie-Hellman permite que duas partes (cliente e servidor) compartilhem uma chave secreta através de um canal público. A segurança deste processo é garantida pela dificuldade de resolver o problema do logaritmo discreto.

Passos executados:

1. **Parâmetros fornecidos**:
    - p: Número primo (fornecido pelo professor).
    - g: Número gerador (também fornecido pelo professor).
2. **Escolha do valor privado a**:
   O valor a foi escolhido como 12345678901234567890123456789O, com 30 dígitos conforme solicitado.
3. **Cálculo do valor público A**: A fórmula utilizada foi:

$$A = g^a \bmod p$$

A variável A foi calculada usando a função pow(g, a, p) no código, e o resultado foi convertido para a base hexadecimal para ser enviado ao professor.

4. **Recebimento do valor público B**: Após o cálculo de A, o professor forneceu o valor B (também em hexadecimal), que foi convertido para inteiro para ser utilizado no cálculo da chave compartilhada.
5. **Cálculo de V**: A chave compartilhada foi calculada utilizando o valor B recebido e a fórmula:

$$V = B^a \bmod p$$

O valor de V foi então utilizado para derivar a chave simétrica para a criptografia.

6. **Derivação da chave AES**: Utilizando o valor de V, foi aplicada a função SHA256 para derivar uma chave de 256 bits. Os 128 bits menos significativos dessa chave foram utilizados como a chave de sessão S_key, que será usada para criptografar e descriptografar as mensagens.

3. Etapa 2 – Troca de Mensagens Criptografadas

Na segunda etapa, o objetivo foi garantir a segurança na troca de mensagens utilizando a chave derivada com AES no modo CBC. A mensagem foi invertida e cifrada com AES, como solicitado.

Passos executados:

1. **Descriptografando a mensagem recebida**:
    - A mensagem fornecida foi criptografada com AES no modo CBC. Ela foi recebida no formato hexadecimal, contendo o IV (128 bits) seguido da mensagem cifrada.
    - A função descriptografaMensagem() foi utilizada para descriptografar a mensagem, utilizando a chave de sessão derivada anteriormente (S_key). O IV foi extraído dos primeiros 128 bits, e a mensagem foi descriptografada com a função unpad() para remover o padding aplicado durante a criptografia.
2. **Inversão da mensagem**:
    - Após a descriptografia da mensagem, foi realizado o processo de inversão dos caracteres da mensagem como parte da tarefa.
3. **Criptografando a mensagem invertida**:
    - A mensagem invertida foi então criptografada novamente utilizando o AES no modo CBC com um IV para garantir a segurança da mensagem retornada.
    - A função criptografaMensagem() foi responsável por criptografar a mensagem invertida. O IV foi gerado usando a função os.urandom(), e o padding foi aplicado para garantir que o tamanho da mensagem fosse múltiplo de 16 bytes.
4. **Envio da mensagem invertida criptografada**:
    - A mensagem criptografada, no formato hexadecimal (IV seguido da mensagem criptografada), foi então gerada e pronta para ser enviada de volta ao professor.

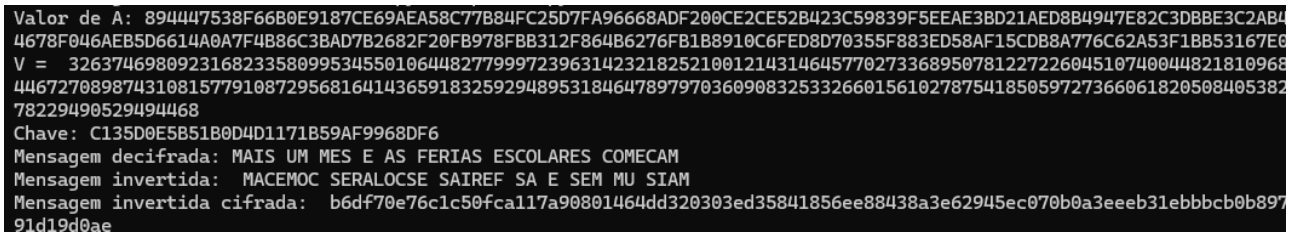
4. Código

O código está disponível no seguinte repositório no GitHub: **https://github.com/gpetuco/httpsGabrielPetuco** (necessário instalar bibliotecas como pycryptodome).

```python
import hashlib
import base64
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad, pad
from Crypto.Cipher import AES
import os

# Aluno: Gabriel Frigo Petuco

#Valores de p e g fornecidos pelo professor Edson
p_hex = (
    "B10B8F96A080E01DDE92DE5EAE5D54EC52C99FBCFB06A3C6"
    "9A6A9DCA52D23B616073E28675A23D189838EF1E2EE652C0"
    "13ECB4AEA906112324975C3CD49B83BFACCBDD7D90C4BD70"
    "98488E9C219A73724EFFD6FAE5644738FAA31A4FF55BCCC0"
    "A151AF5F0DC8B4BD45BF37DF365C1A65E68CFDA76D4DA708"
    "DF1FB2BC2E4A4371"
)
g_hex = (
    "A4D1CBD5C3FD34126765A442EFB99905F8104DD258AC507F"
    "D6406CFF14266D31266FEA1E5C41564B777E690F5504F213"
    "160217B4B01B886A5E91547F9E2749F4D7FBD7D3B9A92EE1"
    "909D0D2263F80A76A6A24C087A091F531DBF0A0169B6A28A"
    "D662A4D18E73AFA32D779D5918D08BC8858F4DCEF97C2A24"
    "855E6EEB22B3B2E5"
)

# Chave privada de 30 dígitos
a = 123456789012345678901234567890

# Converte p e g
p = int(p_hex, 16)
g = int(g_hex, 16)

#Diffie Hellman
# A = g^a mod p
A = pow(g, a, p)

# Abaixo, o valor de A informado ao professor no Moodle, convertido em hexadecimal
print(f"Valor de A: {hex(A)[2:].upper()}")

# B do professor Edson, fornecido no Moodle
B_hex = "008DC95194C5F1A0490A284686DEF42F8A03F33D590ECAFF9A273507118C0C88FC67748B1AE33001CC9C5D13
# Converte
B = int(B_hex, 16)

# V = B^a mod p
def calculaV(B, a, p):
    return pow(B, a, p)

V = calculaV(B, a, p)
print("V = ", V)

# Derivar chave AES de 128 bits
V_bytes = V.to_bytes((V.bit_length() + 7) // 8, byteorder="big")
S_full = hashlib.sha256(V_bytes).digest()
S_key = S_full[-16:]  # 128 bits menos significativos

print(f"Chave: {S_key.hex().upper()}")
```

```
60
61  # Descriptografa os dados usando AES (CBC).
62  # Entrada: string hexadecimal [128 bits de IV][mensagem].
63  def descriptografaMensagem(chaveSecao, mensagemCriptografada):
64      enc = bytes.fromhex(mensagemCriptografada)
65      iv = enc[:AES.block_size]
66      cipher = AES.new(chaveSecao, AES.MODE_CBC, iv)
67      plaintext = unpad(cipher.decrypt(enc[AES.block_size:]), AES.block_size)
68      return plaintext.decode('utf-8')
69
70  # Inverte a mensagem fornecida
71  def inverteMensagem(mensagem):
72      return mensagem[::-1]
73
74  def criptografaMensagem(chaveSecao, mensagem):
75      # IV (16 bytes)
76      iv = os.urandom(AES.block_size)
77
78      # Converte a mensagem para bytes
79      mensagem_bytes = mensagem.encode('utf-8')
80
81      # Aplica o padding para garantir que a mensagem tenha múltiplos de 16 bytes
82      mensagem_padded = pad(mensagem_bytes, AES.block_size)
83
84      # Objeto de criptografia AES com modo CBC
85      cipher = AES.new(chaveSecao, AES.MODE_CBC, iv)
86
87      # Criptografa a mensagem
88      mensagem_criptografada = cipher.encrypt(mensagem_padded)
89
90      # Retorna o IV e a mensagem criptografada como um único valor hexadecimal
```

```
74  def criptografaMensagem(chaveSecao, mensagem):
81      # Aplica o padding para garantir que a mensagem tenha múltiplos de 16 bytes
82      mensagem_padded = pad(mensagem_bytes, AES.block_size)
83
84      # Objeto de criptografia AES com modo CBC
85      cipher = AES.new(chaveSecao, AES.MODE_CBC, iv)
86
87      # Criptografa a mensagem
88      mensagem_criptografada = cipher.encrypt(mensagem_padded)
89
90      # Retorna o IV e a mensagem criptografada como um único valor hexadecimal
91      return (iv + mensagem_criptografada).hex()
92
93  # Mensagem cifrada fornecida em hexadecimal pelo professor Edson
94  mensagem_cifrada_hex = (
95  "e4964b319c6637f7a21cfc928494a9d3e8d947622c504703108b5eb7f1e9c80a1a284f9d61b67af086
96  )
97
98  # Descriptografa
99  mensagemDecifrada = descriptografaMensagem(S_key, mensagem_cifrada_hex)
100 print(f"Mensagem decifrada: {mensagemDecifrada}")
101
102 # Inverte mensagem decifrada
103 mensagemInvertida = inverteMensagem(mensagemDecifrada)
104 print("Mensagem invertida: " , mensagemInvertida)
105
106 # Criptografa mensagem invertida
107 mensagemInvertidaCriptografada = criptografaMensagem(S_key, mensagemInvertida)
108
109 print("Mensagem invertida cifrada: " , mensagemInvertidaCriptografada)
```



5. Dados e resultados

A mensagem decifrada foi **MAIS UM MÊS E AS FÉRIAS ESCOLARES COMECAM**. A mensagem decifrada foi invertida para MACEMOC SERALOCSE SAIREF SA E SEM MU SIAMMACEMOC SERALOCSE SAIREF SA E SEM MU SIAM e cifrada para **b6df70e76c1c50fca117a90801464dd320303ed35841856ee88438a3e62945ec070b0a3eeeb31ebbbcb0b8972a7e5d793f9789adfd6cd26ca4f051791d19d0ae**.

O meu valor de A informado no Moodle era:
0X894447538F66B0E9187CE69AEA58C77B84FC25D7FA96668ADF200CE2CE52B423C59839F5EEAE3BD21AED8B4947E82C3DBBE3C2AB457DBD53696F652C6DD071394C78C44D4678F046AEB5D6614A0A7F4B86C3BAD7B2682F20FB978FBB312F864B6276FB1B8910C6FED8D70355F883ED58AF15CDB8A776C62A53F1BB53167E054A.

O valor de B informado pelo professor era:
008DC95194C5F1A0490A284686DEF42F8A03F33D590ECAFF9A273507118C0C88FC67748B1AE33001CC9C5D13E5C12C5EA7920FC1F50D1E3F9E43ACC61950FE69004BD7AE763FB5F7D6DD8F6684C9335F5B158416722FE31389BAD9FCF086C7156F047FA087BB635E024BB0344503CC12108881846A26AA2677F95C052D8CFB5D9D.

A mensagem criptografada (IV + EMsg) enviada pelo professor era:

e4964b319c6637f7a21cfc928494a9d3e8d947622c504703108b5eb7f1e9c80a1a284f9d61b67af086d035cb1ef433bdb877db37dfc629f1bcd0c93beb466880

Foi possível decifrar a mensagem enviada pelo professor, obtendo sucesso na realização da solução do trabalho proposto.

6. Conclusão

A solução desenvolvida implementa corretamente as etapas solicitadas no trabalho prático, utilizando os protocolos de Diffie-Hellman para a troca de chaves e AES no modo CBC para a criptografia e descriptografia de mensagens. O código está modularizado, permitindo fácil compreensão e manutenção das funções específicas para cada parte do processo. A troca segura de mensagens foi garantida, e a implementação seguiu as especificações detalhadas no enunciado.