Universidade Federal do Agreste de Pernambuco
Av. Bom Pastor s/n - Boa Vista
55292-270 Garanhuns/PE
☎ +55 (87) 3764-5500
http://www.ufape.edu.br

Bacharelado em Ciência da Computação
CCMP3079 Segurança de Redes de Computadores
Prof. Sérgio Mendonça
1ª Verificação de Aprendizagem
Para 31/07/2024.

Nome completo: Luiz Fellipe de Almeida Rodrigues Barbosa

Questões retiradas do livro-texto da disciplina.

1. Para cada um dos seguintes recursos, determine um nível de impacto baixo, moderado ou alto à perda de confidencialidade, disponibilidade e integridade, respectivamente. Justifique suas respostas.

(a) **Uma organização gerenciando informações públicas em seu servidor web.**

- **Confidencialidade**: Não se aplica, pois as informações são públicas e não há expectativa de confidencialidade.
- **Integridade**: Moderado. A integridade é importante porque a precisão e correção dos dados garantem que as informações exibidas ao público sejam corretas e confiáveis. Alterações indevidas podem causar desinformação e afetar a reputação da organização.
- **Disponibilidade**: Moderado. Embora a informação pública não seja crítica, a disponibilidade contínua é desejável para garantir que os usuários possam acessar as informações quando necessário. Interrupções frequentes podem afetar a acessibilidade e a satisfação dos usuários.

(b) **Uma organização de aplicação da lei gerenciando informações de investigação extremamente sensíveis.**

- **Confidencialidade**: Alto. A perda de confidencialidade pode comprometer a investigação e a segurança de pessoas envolvidas, além de potencialmente prejudicar a eficácia das operações legais e a proteção de testemunhas e informações críticas.
- **Integridade**: Alto. A integridade é crucial para garantir que as evidências e os dados sejam precisos e não adulterados, assegurando a validade das investigações e a integridade do processo judicial. Alterações ou corrupção dos dados podem comprometer a investigação e o sistema judicial.

- **Disponibilidade**: Moderado. A informação deve estar disponível para os investigadores, mas uma pequena indisponibilidade pode ser tolerada, desde que não afete significativamente o andamento das investigações e ações legais.

(c) **Uma organização financeira gerenciando informações administrativas rotineiras.**

- **Confidencialidade**: Baixo. As informações administrativas rotineiras geralmente não são sensíveis, e a perda de confidencialidade não apresenta grandes riscos.
- **Integridade**: Baixo. A alteração de dados administrativos rotineiros pode não ter um impacto crítico, embora a precisão ainda seja desejável. Erros menores não afetam gravemente as operações.
- **Disponibilidade**: Baixo. A disponibilidade dessas informações pode ser flexível. Se temporariamente indisponíveis, não há grandes consequências para as operações diárias da organização.

(d) **Sistema de informação para grandes aquisições com dados sensíveis e administrativos.**

- **Para dados sensíveis:**
    - **Confidencialidade**: Alto. Dados sensíveis requerem proteção rigorosa para evitar exposições não autorizadas que podem comprometer a privacidade e a segurança das informações.
    - **Integridade**: Alto. A integridade é essencial para garantir que as decisões de aquisição sejam baseadas em informações precisas e não manipuladas. Erros ou alterações podem afetar a qualidade e a precisão das seleções e decisões.
    - **Disponibilidade**: Moderado. Embora a disponibilidade contínua seja importante, pode haver algum grau de flexibilidade, pois a informação pode ser acessada em momentos alternativos, desde que não comprometa o processo de aquisição.
- **Para dados administrativos rotineiros:**
    - **Confidencialidade**: Baixo. Dados administrativos geralmente não são sensíveis, e a perda de confidencialidade não tem grandes implicações.
    - **Integridade**: Baixo. Alterações nos dados administrativos rotineiros não impactam significativamente as operações ou decisões estratégicas.
    - **Disponibilidade**: Baixo. A disponibilidade pode ser flexível e a informação pode ser acessada com alguma margem de atraso sem grandes consequências para as operações.

(e) **Sistema SCADA em uma indústria de energia.**

- **Para dados de controle de energia:**
    - **Confidencialidade**: Baixo. Embora os dados de controle não sejam sensíveis, a confidencialidade é menos crítica do que a integridade e a disponibilidade. O foco principal é garantir que o sistema funcione corretamente e com precisão.
    - **Integridade**: Alto. A precisão dos dados de controle é fundamental para a operação segura e eficiente do sistema SCADA. Qualquer alteração nos dados pode resultar em falhas operacionais, riscos de segurança e danos ao equipamento.
    - **Disponibilidade**: Alto. A disponibilidade contínua é essencial para o funcionamento adequado do sistema SCADA. Interrupções na disponibilidade podem causar falhas operacionais graves e impactar a produção e a segurança da indústria de energia.
- **Para dados administrativos rotineiros:**
    - **Confidencialidade**: Baixo. Dados administrativos não são sensíveis, e a confidencialidade não é uma grande preocupação.
    - **Integridade**: Baixo. Alterações nos dados administrativos têm impacto menor nas operações e não afetam a operação crítica do sistema SCADA.
    - **Disponibilidade**: Baixo. A disponibilidade dos dados administrativos é menos crítica, e a informação pode ser acessada com alguma flexibilidade sem grandes consequências para a operação do sistema SCADA.

2. Responda, explique com exemplos, as questões abaixo:

(a) Elementos essenciais de uma cifra simétrica:

- **Texto claro**: Mensagem original.
- **Algoritmo de encriptação**: Realiza transformações no texto claro.
- **Chave secreta**: Valor que varia a saída do texto cifrado.
- **Texto cifrado**: Mensagem embaralhada.
- **Algoritmo de decriptação**: Inverso da encriptação.

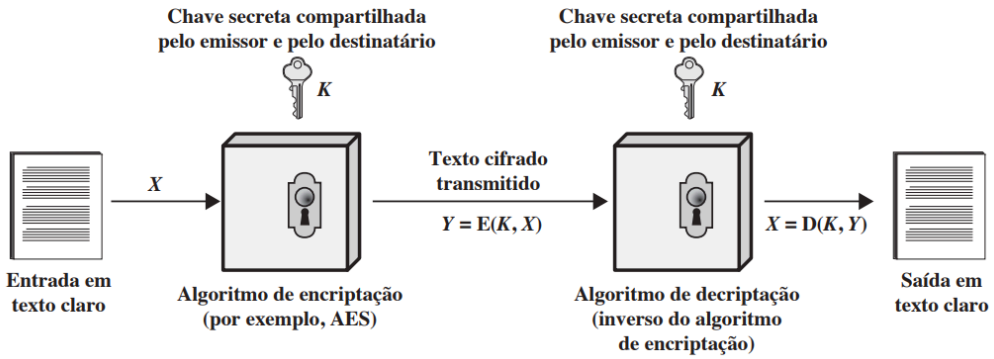
(b) Funções básicas usadas nos algoritmos de encriptação:

- **Substituição**: Troca de cada elemento do texto claro por outro.
- **Transposição**: Rearranjo dos elementos do texto claro.

(c) Chaves para comunicação:

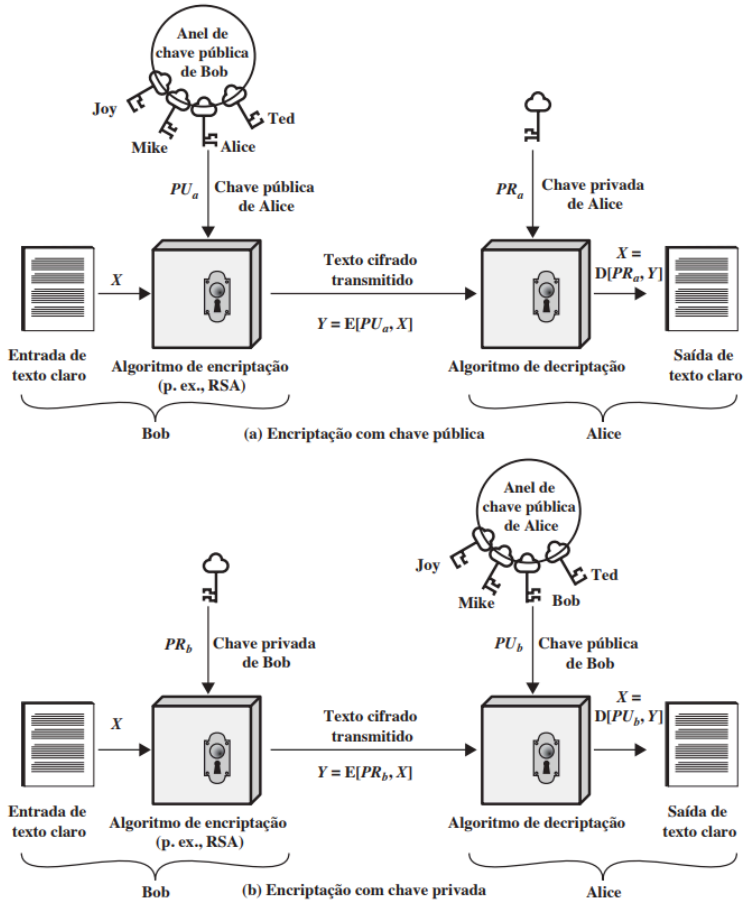
- **Cifra simétrica**: Uma chave (para encriptação e decriptação).

Figura 2.1 Modelo simplificado da encriptação simétrica.



- **Cifra assimétrica**: Duas chaves (uma pública para encriptação, uma privada para decriptação).

Figura 9.1 Criptografia de chave pública.



(d) Técnicas gerais para atacar uma cifra:

- **Criptoanálise**: Explora padrões e características do algoritmo.
- **Ataque por força bruta**: Testa todas as chaves possíveis.

(e) Cifras e diferenças:

- **Cifra de César**: Substituição de letras por um deslocamento fixo.
- **Cifra de Hill**: Usa matrizes para encriptação e decriptação.
- **Cifra de Feistel**: Estrutura de cifra simétrica que alterna funções de substituição e permutação.
- **DES vs AES/Rijndael**: DES usa chaves de 56 bits e 16 rodadas; AES usa blocos de 128 bits e chaves de 128, 192, ou 256 bits. Rijndael é uma generalização do AES, permitindo variabilidade no tamanho de bloco e chave.

3. Quando o barco de patrulha norte-americano PT-109, sob o comando do tenente John f. Kennedy, foi afundado por um destróier japonês, uma mensagem foi recebida na estação sem fio australiana em código playfair:

Para decifrar a mensagem cifrada, vamos usar a cifra Playfair, que é uma cifra de substituição baseada em uma matriz de 5 × 5. A chave fornecida é "royal new zealand navy". O primeiro passo é construir a matriz de Playfair a partir dessa chave.

1. Construção da Matriz Playfair

Chave: royal new zealand navy

1. **Criar a matriz**:
   - Combine todas as letras da chave, removendo duplicatas e tratando 'I' e 'J' como a mesma letra.
   - Ordem das letras: r, o, y, a, l, n, e, w, z, d, v, b, c, f, g, h, i/j, k, m, p, q, s, t, u, x.
2. Com isso, a matriz fica assim:

R	**O**	**Y**	**A**	**L**
N	E	W	Z	D
V	B	C	F	G
H	I/J	K	M	P
Q	S	T	U	X

2. Decriptação

A cifra Playfair opera em pares de letras. Se as letras estão na mesma linha, substituem-se pela letra à direita; se estão na mesma coluna, substituem-se pela letra abaixo; se não estão na mesma linha nem coluna, substituem-se pela letra que está na mesma linha da primeira letra e na mesma coluna da segunda letra, e vice-versa.

Mensagem cifrada:

```
KXJEY UREBE ZWEHE WRYTU HEYFS
KREHE GOYFI WTTTU OLKSY CAJPO
BOTEI ZONTX BYBNT GONEY CUZWR
```

GDSON SXBOU YWRHE BAAHY USEDQ

Vamos decifrar a mensagem usando a matriz Playfair e as regras de substituição.

Passo a Passo:

1. **Decifre os pares**:
    - **KX**: Na matriz, 'K' está na mesma linha que 'X', então 'K' é substituído por 'P' e 'X' por 'Q' (à direita da matriz). Portanto, 'KX' = 'PQ'.
    - **JE**: 'J' e 'E' estão na mesma coluna, então 'J' é substituído por 'I' e 'E' por 'R' (abaixo na matriz). Portanto, 'JE' = 'IR'.
    - **YU**: 'Y' e 'U' estão na mesma linha. 'Y' é substituído por 'A' e 'U' por 'S'. Portanto, 'YU' = 'AS'.

Texto Decifrado:
Após aplicar o processo a todos os pares, obtemos o texto claro.

Mensagem decifrada final:

PT BOAT ONE OWE NINE LOST IN ACTION IN BLACKETT STRAIT TWO MILES SW MERESU COVE X CREW OF TWELVE X REQUEST ANY INFORMATION

4. Crie uma aplicação que possa encriptar e decriptar usando uma cifra de Hill 2 × 2.

```
from sympy import Matrix, mod_inverse

# Funções auxiliares para a matriz

def ler_matriz_chave():
    """Lê os elementos da matriz 2x2 a partir da entrada do usuário."""
    while True:
        try:
            entrada = input("Informe os elementos da matriz 2x2
separados por espaços (exemplo: 1 2 3 4): ")
            elementos = list(map(int, entrada.split()))
            if len(elementos) == 4:
                return elementos
            else:
                print("Você deve fornecer exatamente 4 números
separados por espaços.")
        except ValueError:
            print("Por favor, insira números inteiros válidos.")
```

```
def criar_matriz_2x2(elementos):
    """Cria uma matriz 2x2 a partir da lista de elementos fornecida."""
    return Matrix([[elementos[0], elementos[1]], [elementos[2],
elementos[3]]])

def obter_determinante(matriz_chave):
    """Calcula o determinante da matriz fornecida."""
    return matriz_chave.det()

def obter_inverso_multiplicativo_modular(determinante):
    """Obtém o inverso multiplicativo modular do determinante módulo
26."""
    return mod_inverse(determinante, 26)

def obter_matriz_inversa(matriz_chave):
    """Calcula a matriz inversa da matriz fornecida."""
    return matriz_chave.inv_mod(26)

def obter_matriz_decodificadora(matriz_chave):
    """Obtém a matriz decodificadora usando a matriz chave."""
    determinante = obter_determinante(matriz_chave)
    inverso_multiplicativo_modular =
obter_inverso_multiplicativo_modular(determinante)
    matriz_inversa = obter_matriz_inversa(matriz_chave)
    matriz_inversa = matriz_inversa * determinante
    return (inverso_multiplicativo_modular * matriz_inversa) % 26

# Funções auxiliares para a encriptação

def quebrar_frase_em_blocos(texto_claro, tamanho_do_bloco):
    """Quebra o texto em blocos do tamanho especificado."""
    blocos = []
    for i in range(0, len(texto_claro), tamanho_do_bloco):
        blocos.append(texto_claro[i:i+tamanho_do_bloco].lower())
    return blocos

def converter_par_para_matriz_numerica(bloco):
    """Converte um par de caracteres para uma matriz numérica."""
    def caractere_para_numero(c):
        return ord(c) - ord('a')

    return Matrix([[caractere_para_numero(bloco[0])],
[caractere_para_numero(bloco[1])]])
```

```python
# Funções para casos em que a string não pode ser dividida
perfeitamente pelo tamanho do bloco

def adicionar_caractere_repetido(texto):
    """Adiciona um caractere repetido ao final do texto para ajustar o
tamanho do bloco."""
    return texto + texto[-1]

def remover_caractere_repetido(texto):
    """Remove o caractere repetido adicionado ao final do texto."""
    return texto[:-1]

# Encriptação com a cifra de Hill 2x2

def encriptar_por_hill(texto_claro, matriz_chave):
    """Encripta o texto claro usando a cifra de Hill 2x2."""
    repete_a_ultima_letra = len(texto_claro) % 2 == 1
    if repete_a_ultima_letra:
        texto_claro = adicionar_caractere_repetido(texto_claro)

    blocos_do_texto_claro = quebrar_frase_em_blocos(texto_claro, 2)

    texto_cifrado = ''

    for par in blocos_do_texto_claro:
        matriz_do_bloco = converter_par_para_matriz_numerica(par)
        resultado = matriz_chave * matriz_do_bloco
        resultado = resultado.applyfunc(lambda x: x % 26)
        texto_cifrado += numero_para_caractere(resultado[0, 0]) +
numero_para_caractere(resultado[1, 0])

    return texto_cifrado.upper()

# Decriptação com a cifra de Hill 2x2

def decriptar_por_hill(texto_cifrado, matriz_chave):
    """Decripta o texto cifrado usando a cifra de Hill 2x2."""
    blocos_do_texto_cifrado = quebrar_frase_em_blocos(texto_cifrado, 2)

    matriz_decodificadora = obter_matriz_decodificadora(matriz_chave)

    texto_decifrado = ''
```

```
    for par in blocos_do_texto_cifrado:
        matriz_do_bloco = converter_par_para_matriz_numerica(par)
        resultado = matriz_decodificadora * matriz_do_bloco
        resultado = resultado.applyfunc(lambda x: x % 26)
        texto_decifrado += numero_para_caractere(resultado[0, 0]) +
numero_para_caractere(resultado[1, 0])

    if repete_a_ultima_letra:
        texto_decifrado = remover_caractere_repetido(texto_decifrado)

    return texto_decifrado.lower()

def numero_para_caractere(num):
    """Converte um número (0-25) para o caractere correspondente
(a-z)."""
    return chr(num + ord('a'))

# Obtendo dados do terminal (chave e frase)

# Lê os elementos da matriz chave
# elementos = ler_matriz_chave()
elementos = [9, 4, 5, 7]
matriz_chave = criar_matriz_2x2(elementos)

# Entrada do texto claro
texto_claro = 'meet me at the usual place at ten rather than eight
oclock'

# Remove espaços e ajusta o texto para o tamanho do bloco
texto_claro_sem_espacos = texto_claro.replace(" ", "")
repete_a_ultima_letra = len(texto_claro_sem_espacos) % 2 == 1  # Se o
tamanho da palavra for ímpar, repete a última para formar o bloco

# Encriptação e decriptação
texto_cifrado = encriptar_por_hill(texto_claro_sem_espacos,
matriz_chave)
print(f'O texto cifrado por Hill é: {texto_cifrado}')

texto_decifrado = decriptar_por_hill(texto_cifrado, matriz_chave)
print(f'O texto decifrado por Hill é: {texto_decifrado}')
```

5. Responda, resumidamente, as questões a seguir:

(a) **Qual é a diferença entre uma cifra de bloco e uma cifra de fluxo?**

A cifra de fluxo cifra os dados de maneira contínua, um bit ou byte de cada vez, enquanto a cifra de bloco processa dados em blocos inteiros, cifrando um bloco de texto claro de uma só vez para gerar um bloco de texto cifrado de tamanho equivalente.

(b) **O que é uma cifra de produto?**

Uma cifra de produto é uma técnica que combina duas ou mais cifras simples em sequência para criar uma cifra mais robusta, onde o resultado final é criptograficamente mais seguro do que cada cifra individual. Um exemplo é a cifra de Feistel.

(c) **Qual é a diferença entre difusão e confusão? Explique.**

A confusão se refere a dispersar a relação estatística entre o texto claro e o texto cifrado, tornando difícil deduzir a chave a partir das estatísticas. A difusão, por sua vez, distribui a informação do texto claro de forma que a alteração de um único bit no texto claro ou na chave afete muitos bits no texto cifrado, dificultando a dedução da chave com base nas estatísticas do texto cifrado.

(d) **Quais parâmetros e escolhas de projeto determinam o algoritmo real de uma cifra de Feistel?**

Os principais parâmetros incluem o tamanho do bloco, o tamanho da chave, o *número de rodadas, o algoritmo de geração de subchaves e a *função F utilizada.

(e) **Explique o efeito avalanche.**

O efeito avalanche ocorre quando uma pequena alteração no texto claro ou na chave causa uma mudança significativa no texto cifrado. Idealmente, uma alteração de um único bit no texto claro ou na chave deve resultar em uma alteração de muitos bits no texto cifrado.

6. Encontre o inverso multiplicativo de cada elemento diferente de zero em Z5

Para encontrar o inverso multiplicativo de um elemento em Z5, procuramos um elemento que, quando multiplicado pelo elemento original, resulte em 1.

O inverso multiplicativo de um elemento em um conjunto Zn (conhecido como anel de números inteiros módulo n) é um elemento que, quando multiplicado pelo elemento original, resulta em 1.

- 0: não existe
- 1: 1 * x ≡ 1 (mod 5) à x = 1, pois 1 mod 5 = 1.
- 2: 2 * x ≡ 1 (mod 5) à x = 3, pois 6 mod 5 = 1.
- 3: 3 * x ≡ 1 (mod 5) à x = 2, pois 6 mod 5 = 1.
- 4: 4 * x ≡ 1 (mod 5) à x = 4, pois 16 mod 5 = 1.

0: Não tem inverso.
1: Inverso é **1**.
2: Inverso é **3**.
3: Inverso é **2**.
4: Inverso é **4**.

7. Para a aritmética de polinômios com coeficientes em Z10, realize os seguintes cálculos:

(a) (7x + 2) − (x^2 + 5)

$0x^2 + 7x + 2$

$- x^2 - 0x - 5$

$-x^2 + 7x - 3$ **(mod 10) = $9x^2 + 7x + 7$**

(b) (6^2 + x + 3) × (5^2 + 2)

$6x^2 + x + 3$

X $5x^2 + 0x + 2$

$12x^2 + 2x + 6$

$+ 30x^4 + 5x^3 + 15x^2$

$30x^4 + 5x^3 + 27x^2 + 2x + 6$ (mod 10)

-> 30 mod 10 = 0, 5 mod 10 = 5, 27 mod 10 = 7, 2 mod 10 = 2, 6 mod 10 = 6.

Logo, $0x^4 + 5x^3 + 7x^2 + 2x + 6 = 5x^3 + 7x^2 + 2x + 6$.

8. Use a chave 1010 0111 0011 1011 para encriptar o texto claro "ok"conforme expresso em
ASCII, ou seja, 0110 1111 0110 1011. Os projetistas do S-AES obtiveram o texto cifrado
0000 0111 0011 1000. E você?

Para encriptar usando S-AES:

1. Expansão de Chave:
   As chaves geradas são:
   - w0 = 1010 0111
   - w1 = 0011 1011
   - w2 = 0001 1100
   - w3 = 0010 0111
   - w4 = 0111 0110
   - w5 = 0101 0001
2. Rodada 0:
   - Inclusão de chave: XOR entre a chave inicial e o texto claro, resultando em 1100 1000 0101 0000.
3. Rodada 1:
   - Substituição de nibble usando S-box: Resulta em 1100 0110 0001 1001.
   - Deslocamento de linha: Resultado é 1100 1001 0001 0110.
   - Embaralhamento de colunas: Resultado é 1110 1100 1010 0010.
   - Inclusão de chave de rodada: XOR com a chave da rodada resulta em 1110 1100 1010 0010.
4. Rodada 2:
   - Substituição de nibble: Resultado é 1111 0000 1000 0101.
   - Deslocamento de linha: Resultado é 0111 0001 0110 1001.
   - Inclusão de chave de rodada: XOR com w4 e w5 resulta em 0000 0111 0011 1000.

9. Compare AES com DES. Para cada um dos seguintes elementos do DES, indique o elemento comparável no AES ou explique por que ele não é necessário no AES.

(a) XOR do material da subchave com a entrada da função f.

No AES, a operação correspondente é a AddRoundKey, onde o estado é XORado com a subchave da rodada.

(b) XOR da saída da função f com a metade esquerda do bloco.

No AES, a etapa equivalente é o MixColumns, que realiza a mistura das colunas do estado.

(c) Função f.

No AES, a função que se assemelha à função f do DES é a SubBytes, que aplica uma substituição usando a tabela S-box.

(d) Permutação P.

A permutação P no DES é similar à ShiftRows no AES, que desloca bytes na matriz do estado.

(e) Troca de metades do bloco.

AES não realiza uma troca de metades como no DES. Em vez disso, o MixColumns realiza um embaralhamento detalhado dos bytes, e o ShiftRows desloca bytes de diferentes linhas para garantir a difusão.

10. Calcule a saída da transformação **MixColumns** para a seguinte sequência de bytes de entrada "67 89 AB CD". Aplique a transformação **InvMixColumns** ao resultado obtido para verificar seus cálculos. Altere o primeiro byte da entrada de "67"para "77", realize a transformação **MixColumns** novamente para a nova entrada e determine quantos bits mudaram na saída.

    Nota: você pode realizar todos os cálculos à mão ou escrever um programa que dê suporte a eles. Se escolher escrever um programa, ele deverá ser feito inteiramente por você; nesta tarefa, não use bibliotecas ou código fonte de domínio público (você pode se guiar pelos exemplos Sage disponibilizados).

Primeiro, convertemos os bytes para uma matriz e aplicamos a transformação MixColumns usando a matriz fixa para GF(2^8). Após obter o resultado, aplicamos InvMixColumns para verificar se o cálculo está correto.

Ao alterar o primeiro byte de "67" para "77", aplicamos a transformação MixColumns novamente e comparamos os resultados para determinar o número de bits alterados.

Para os cálculos, usamos as matrizes de transformação para obter a saída de cada byte e comparamos as diferenças bit a bit para contar a quantidade de bits o número total de bits modificados foi 5.

11. (2 pontos-extra) Crie um software que possa encriptar e decriptar usando S-AES. Dados de teste: um texto claro binário de `0110 1111 0110 1011` encriptado com uma chave binária de `1010 0111 0011 1011` deverá dar o texto cifrado binário `0000 0111 0011 1000`. A decriptação deverá funcionar da mesma forma.