Universidade de Brasília - Instituto de Ciências Exatas
Departamento de Ciência da Computação

Segurança Computacional – Trabalho 2

Cifra AES

Prof. João Gondim
setembro de 2021

Alunos:
Vitor Vasconcelos de Oliveira
Álvaro Veloso Cavalcanti Luz

Matrículas:
180114778
180115391

1 - Informações relevantes sobre o Programa:

- Sistema Operacional usado: Windows 10
- Compilador/linguagem: Python 3.9.0
- IDE utilizada: Visual Studio Code (funcional sem a IDE também)
- São necessárias também algumas bibliotecas python:
  - OpenCV 4.5.1.48: usada para manipulação das imagens
  - Numpy 1.19.5: usada para facilitar a manipulação de vetores
    - OBS: caso não possua essas bibliotecas em sua máquina, basta instalá-las com o comando “pip install opencv” e “pip install numpy”.
    - OBS2: a biblioteca OpenCV em alguns computadores apresenta um bug no qual a função de leitura da imagem “imread()” sempre retorna NoneType. Nosso programa não está preparado para lidar com esse erro, uma vez que ele é causado pela biblioteca em si. Se não ocorrer esse bug, o programa funciona corretamente.

2 - Execução do Programa:

- Para executar o programa basta ter um diretório específico para o programa e uma pasta “fotos” onde devem ser armazenadas as fotos que vão ser cifradas e após serem cifradas.
- Feito isso basta executar o programa python em seu prompt de comando. Para isso pode-se usar “py trabSC2.py” ou “python3 trabSC2.py”
- Após esses passos o programa será executado e surgirá a seguinte interface. A partir daí basta seguir as direções dadas pela interface de acordo com seu desejo; encriptar com ECB, decriptar com ECB, encriptar com CTR e decriptar com CTR.
- É válido que seja notado que o ***Nonce*** gerado aleatoriamente para o algoritmo CTR tem seus valores salvos em um arquivo de texto chamado “randomKeyImage.txt” assim que se executa a codificação neste modo. Esse valor gravado no arquivo é sobrescrito toda vez que se roda o algoritmo, o que requer cuidado por parte do usuário para que não ocorram erros na decodificação.

3 - Problema Abordado:

Este projeto busca abordar uma implementação em python 3.9 do algoritmo AES voltado para a criptografia de imagens. Foram implementadas funções de encriptação e decriptação para dois modos de operação do algoritmo em questão: ECB e CTR.

O algoritmo AES é um algoritmo para encriptação de matrizes de 128 bits (16 bytes) com uma chave ,também 128 bits. Este algoritmo é composto de 4 principais processos:

- **AddRoundKey**: Se trata de um **XOR** simples entre cada byte da matriz com cada byte da chave;
- **SubBytes**: É um processo de substituição de cada byte da matriz, de acordo com seus nibbles, em uma tabela de inversões chamada S-Box;
- **ShiftRows**: Consiste em alterações nas colunas da matriz de forma a realizar a operação de shift com essas colunas, shifta 0 bytes na primeira coluna, 1 byte na segunda, 2 na terceira e 3 na quarta coluna.
- **MixColumns**: É uma multiplicação de polinômios que utiliza a aritmética de Campos Finitos de Galois.

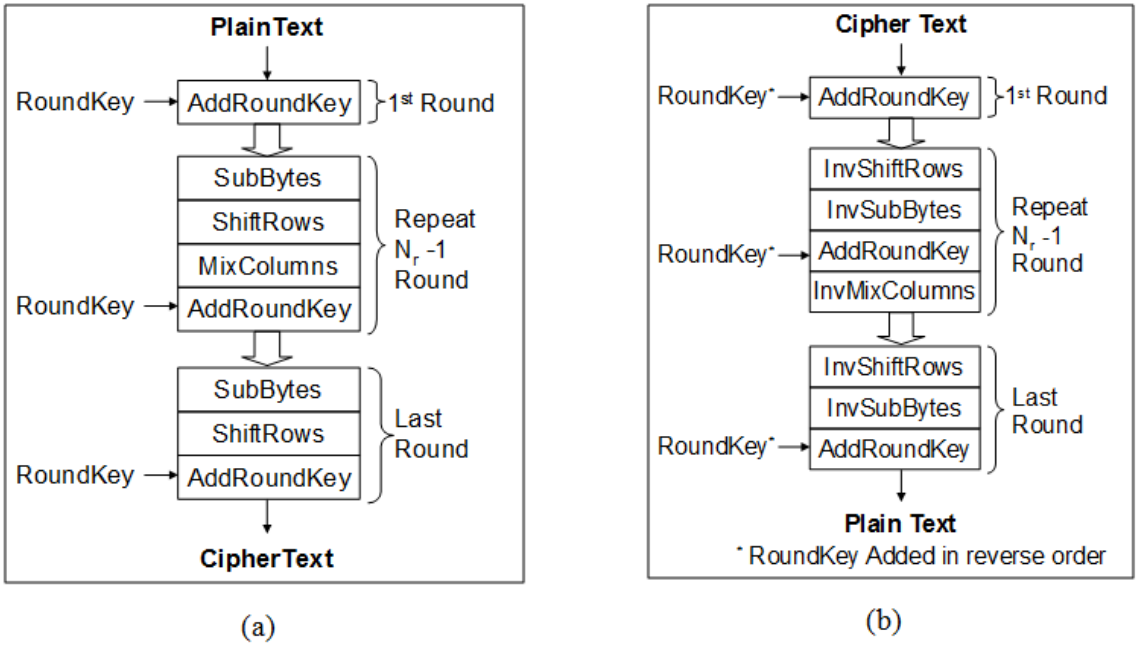
Também existe um processo para a gestão da chave de modo que ela se altere com passar de iterações. Esse processo pode ser chamado de Key Schedule e consiste em isolar a última coluna da chave anterior, mover o primeiro elemento desta coluna para a última posição, realizar um SubBytes com seus valores, e realizar um **XOR** entre o vetor obtido, a primeira coluna da chave anterior e um vetor de quatro posições composto pelo valor correspondente de R-CON para esta iteração seguido de zeros. O vetor obtido através dessa operação torna-se a primeira coluna da nova chave.

As próximas três colunas (segunda, terceira e quarta) da nova chave são obtidas, em ordem, a partir de uma sucessão de **XOR**s entre: a segunda coluna da chave anterior e a primeira coluna gerada na etapa anterior, a terceira coluna da chave anterior e a segunda coluna da nova chave, e , por fim, a quarta coluna da chave anterior e a terceira coluna da nova chave.

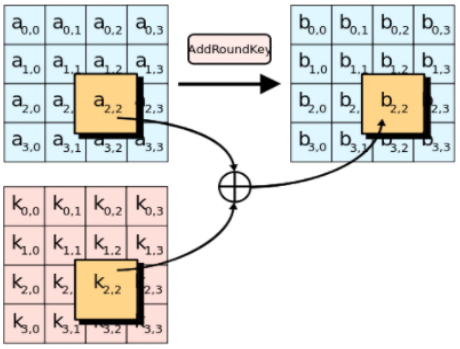
Para decriptar um trecho codificado pelo algoritmo AES precisa-se realizar os processos de forma inversa, o que implica implica em criar os seguintes procedimentos:

- **invShiftRows**: realiza o shiftrows voltando às coluna para suas posições iniciais.
- **invSubBytes**: realiza a substituição do subBytes, porém em um tabela inversa que realiza o processo contrário.
- **invMixColumns**: realiza operação inversa da multiplicação de Campos Finitos de Galois.

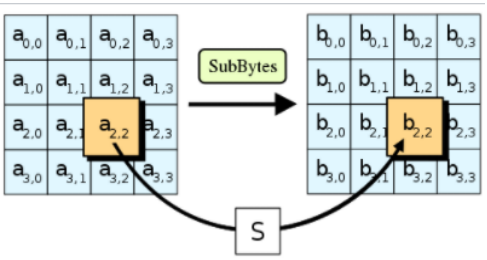
No fluxograma a seguir (**3.1**), exibe-se como ocorre o ciclo do algoritmo AES normal (“**a)**”) e o ciclo do algoritmo de decriptação (”**b)**”) :



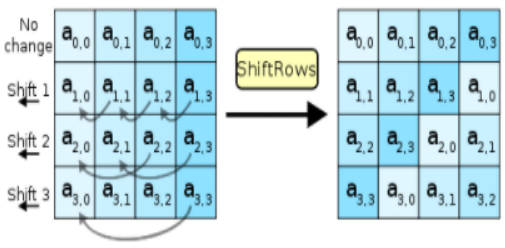
3.1) Fluxograma da encriptação dos blocos de dados feita pelo algoritmo AES.



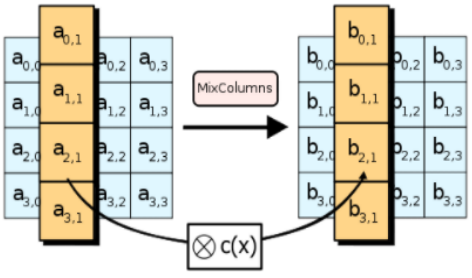
3.2) AddRoundKey.



3.3) SubBytes.



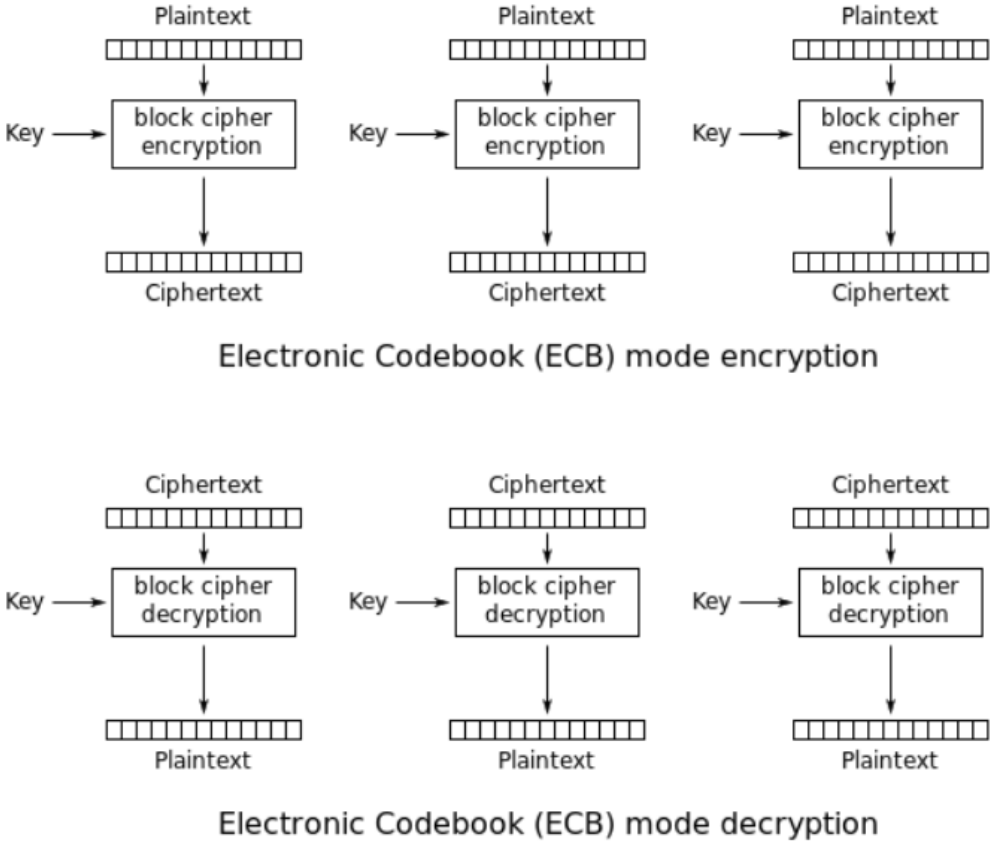
3.4) ShiftRows.



3.5) MixColumns.

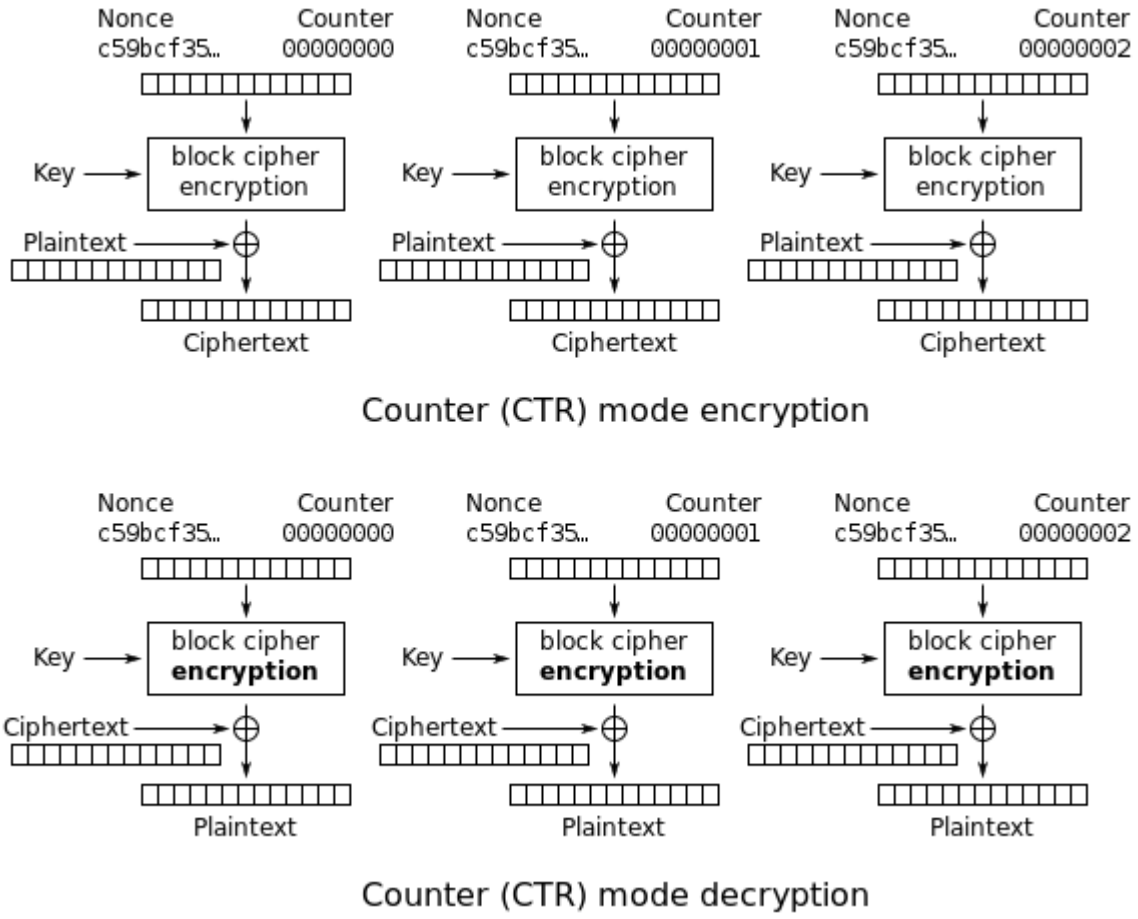
Agora que já sabe-se como funciona o AES pode-se passar para seus dois modos de operação, ECB e CTR, aplicados na encriptação de imagens.

O algoritmo AES-ECB consiste em dividir a imagem em várias matrizes de dimensionalidade 4x4, de 16 bytes cada, e aplicar em cada matriz o algoritmo AES com uma mesma chave, repetindo o mesmo procedimento com todas as matrizes. Por essa simplicidade, o ECB lida com um problema de falta de difusão da criptografia. Para decriptar este modo basta realizar o processo inverso do AES para todos os valores da imagem como feito no processo de encriptação, utilizando os procedimentos inversos de cada etapa: invShiftRows, invSubBytes e invMixColumns, tal como exibido no fluxograma **b)** da imagem **3.1**.



3.6) Fluxograma de funcionamento do algoritmo AES-ECB.

Já para o algoritmo AES-CTR, há uma encriptação que leva em conta a utilização de um fator randômico. Tal algoritmo consiste também da divisão da imagem em várias matrizes de 16 bytes, agora havendo a criação de uma chave randômica de 16 bytes, denominada ***Nonce***, e também de um contador ***counter*** de 16 bytes zerado, que será incrementado em 1 para cada matriz da imagem à medida que executa-se o algoritmo. Implementa-se a seguinte lógica: realiza-se um **XOR** entre o ***Nonce*** e o ***counter***, após isso utiliza-se o produto desse **XOR** na forma de uma matriz de dimensionalidade 4x4 para executar o algoritmo AES sobre esse valor o número de vezes desejado. Em seguida executando outro **XOR** entre o resultado desse procedimento e a matriz de imagem. Repete-se esse processo para todas as matrizes da imagem.



3.7) Fluxograma de funcionamento do algoritmo AES-CTR para encriptação e decriptação.

4 - Elaboração do código:

Para a elaboração do código foi utilizada a metodologia de pair programming para a divisão de tarefas no desenvolvimento. Iniciou-se o desenvolvimento do programa ao definir-se o método de input da chave de codificação da imagem pelo usuário. O método escolhido consiste em gerar uma chave hash sha256 a partir do input de uma string pelo usuário. Esta chave hash de 32 bytes é então dividida em duas metades de 16 bytes, sendo feito, então, um **XOR** entre as duas, resultando em uma só chave de 16 bytes que será convertida em uma matriz de dimensionalidade 4x4. Essa chave gerada, então, será utilizada para a codificação.

Após a obtenção da chave, parte-se para o processamento da imagem, que inicia-se extraindo os valores de RGB para cada pixel da imagem que deseja-se processar. Tais valores são organizados em matrizes de dimensionalidade 4x4 e armazenados em uma lista para futuro processamento.

Separados os valores da imagem, prossegue-se com a elaboração em código da etapa de encriptação. Para tal, foram desenvolvidas funções em python para cada etapa do algoritmo AES descrita no fluxograma 3.1, sendo suas respectivas funções vistas no código-fonte pelos nomes:

- **AddRoundKey**: addRoundKey();

- **ByteSub**: subBytes();
- **ShiftRows**: shiftRows();
- **MixColumns**: mixColumns().

Utilizando-se destas funções, então, busca-se processar os valores das matrizes nas quais estariam contidos os valores de RGB seguindo os modos ECB e CTR. Para tal, foram desenvolvidas as seguintes funções: “cryptECB()” e “CTR()”. Elaboradas de forma a seguir a ordem de execução do AES vista no fluxograma **3.1a)** e também de forma a seguir respectivamente os fluxogramas de encriptação em **3.6** e **3.7**. A lista de matrizes obtida após a execução das etapas de encriptação em todas as iterações é então reorganizada de forma a exibir a imagem ao usuário do programa. A imagem em questão é, em seguida, salva em um arquivo “.jpg”.

Após construídas ambas as funções de encriptação, o desenvolvimento voltou-se para as funções de decriptação para ambos os modos do algoritmo, etapa abstraída em código através das funções “decryptECB()” e “CTR()”. Em “decryptECB()” utiliza-se as funções inversas de encriptação do AES, como mencionado na seção anterior, para isso elaborando as funções invSubBytes(), invShiftRows() e invMixColumns() e as executando conforme a ordem descrita em no fluxograma **3.1b)**. Já para a decriptação do algoritmo AES-CTR, como visto no procedimento da imagem **3.7**, é utilizada a mesma função “CTR()” que foi utilizada na encriptação, apenas com a diferença de estar repassando a imagem já encriptada pelo CTR e o ***Nounce*** armazenado em um arquivo de texto chamado “randomKeyImage.txt”, dessa encriptação CTR anterior.

5 - Resultado e Considerações finais:

Finalizando, pode-se considerar o trabalho como um sucesso. Todas as funções do algoritmo AES foram implementadas e estão funcionais, assim como os seus modos de operação o ECB e CTR foram corretamente implementados e estão funcionando. Um adendo importante entretanto é a possível demora na execução do processamento das imagens principalmente em casos onde a imagem possui uma resolução muito grande ou o número de iterações do AES for alto.

Já com relação a dificuldades na produção do trabalho, as dificuldades centrais encontradas ao longo do desenvolvimento foram voltadas à complexidade de implementação do modelo, dadas as inúmeras formas de manipulação de dados em matrizes presentes no modelo. O que além de ter gerado uma dificuldade em escrever o algoritmo, tornou difícil a otimização do programa.

É válido informar também, que foram incluídos os casos de teste pedidos no relatório do trabalho (ECB - 01,05,09,13 iterações e CTR - 01,05,09,13 iterações) no arquivo “.zip” no qual foi enviado o código-fonte.

6 - Referências:

- https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation
- https://pt.wikipedia.org/wiki/Advanced_Encryption_Standard
- https://www.youtube.com/watch?v=04HK1UHmxcs
- https://www.researchgate.net/figure/The-Diagram-of-AES-Algorithm-a-Encryption-b-Decryption_fig2_331589229

Obrigado!!!