UNIVERSIDADE DE BRASÍLIA
Departamento de Ciência da Computação
Segurança Computacional 2/2023
Prof. Dr. João Gondim

Alunos: Felipe Oliveira do Nascimento Florentino (202021767) e Lucas Amaral de Faria(211055316)

Trabalho de Implementação 2

Cifra AES DE 128 bits e CTR

Neste trabalho, foi desenvolvido um projeto para implementar a Cifra AES, incluindo cifragem, decifragem e modo de operação CTR. Para garantir que o programa funcione corretamente, é importante que o usuário siga as regras de processamento de dados que o programa aceita e siga as instruções dadas pelo programa.

Todas as informações de entrada, como mensagens, cifras e chaves, devem ser inseridas via teclado ou copiadas e coladas no programa.

.

Algoritmo AES

O AES é um algoritmo de criptografia simétrica amplamente utilizado para proteger informações confidenciais. Ele opera em blocos fixos de dados (geralmente 128 bits) e tem três tamanhos de chave possíveis: AES-128 (chave de 128 bits), AES-192 (chave de 192 bits) e AES-256 (chave de 256 bits). No nosso trabalho, foi feita a implementação de uma chave fixa de 128 bits. Possui como base as operações de soma e produto entre matrizes a teoria de corpos finitos de Galois.

Principais funções

As funções principais no AES que foram implementadas incluem:

Função subByte:

Esta função recebe um byte e a S-Box (sBox) como entrada e retorna o resultado da substituição da S-Box na entrada. Isso é usado para realizar a etapa "SubBytes" na criptografia AES.

Função subBytes:

Esta função aplica a operação subByte a cada byte na matriz block. Ela é usada para realizar a etapa "SubBytes" na criptografia AES em todo o bloco de dados.

Função g_function:

Esta função implementa a função g usada na expansão da chave. Ela recebe uma palavra de 4 bytes (w) e um número de rodada (round) e aplica uma série de operações, incluindo rotação, substituição da S-Box e operação XOR com uma constante de rodada. O resultado é usado na expansão da chave.

Função keyExpansion:

Esta função expande a chave de criptografia AES. Ela recebe a chave original e o número de rodada e retorna a chave expandida para essa rodada. A função realiza operações de expansão, incluindo a aplicação da função g_function e operações XOR.

Função mixColumns:

Esta função aplica a operação "MixColumns" à matriz block, usando a tabela columnTable fornecida. Essa operação envolve uma combinação linear de colunas da matriz.

Função addRoundKey:

Esta função realiza a operação "AddRoundKey" na matriz block. Ela aplica uma operação XOR entre a matriz block e a chave da rodada.

Função shiftRows:

Esta função executa a operação "ShiftRows" na matriz block, que envolve o deslocamento cíclico das linhas da matriz.

Função inverseShiftRows:

Esta função executa a operação inversa da "ShiftRows", restaurando as linhas à sua posição original.

Função aesEncript:

Esta função implementa a criptografia AES. Ela recebe um bloco de dados block, a chave key e o número de rodadas n_rounds. A função executa as etapas do algoritmo AES, incluindo a expansão da chave, as operações "SubBytes", "ShiftRows", "MixColumns" e "AddRoundKey" para cada rodada, bem como a rodada final.

Função aesDecript:

Esta função implementa a decifragem AES. Ela recebe um bloco de dados block, a chave key e o número de rodadas n_rounds. A função realiza a decifragem reversa executando as operações inversas na ordem reversa das etapas de cifragem.

Função generate_IV_matrix:

Esta função gera uma matriz IV (Vetor de Inicialização) para uso no modo de operação CTR. Ela usa uma chave de 64 bits (8 bytes) como base para gerar o IV.

Função getEncriptedCTRIV:

Esta função gera o bloco IV cifrado para o modo CTR usando o IV original, a chave e um contador. Ela aplica a cifragem AES ao IV com base no contador e na chave, retornando o IV cifrado.

Função getEncriptedCipherBlock:

Esta função gera o bloco de cifra final aplicando uma operação XOR entre o bloco cifrado do IV e o bloco de texto claro.

Função aesCTR:

Esta função implementa o modo CTR do AES. Ela recebe uma chave, o caminho de um arquivo a ser cifrado, o número de rodadas e um IV. A função lê o arquivo, divide-o em blocos, cifra cada bloco e retorna os blocos cifrados.

Seguem imagens dos blocos de código usados para cifração e decifração.

```
array<array<uint8_t, 4>, 4> aesEncript(array<array<uint8_t, 4>, 4> block, array<array<uint8_t, 4>, 4> key, int n_rounds){

    //Initial round
    addRoundKey(key,block);



    //1 : n-1 rounds
    for(int i = 1;i<n_rounds;i++){
        subBytes(block, sBox);
        shiftRows(block);
        mixColumns(block, columnTable);


        key = keyExpansion(key,i);
        addRoundKey(key,block);
    }

    //Final round
    subBytes(block, sBox);
    shiftRows(block);
    key = keyExpansion(key,n_rounds);
    addRoundKey(key,block);

    return block;
}
```

```
array<array<uint8_t, 4>, 4> aesDecript(array<array<uint8_t, 4>, 4> block, array<array<uint8_t, 4>, 4> key, int n_rounds){
    vector<array<array<uint8_t, 4>, 4>> roundKeys;
    roundKeys.push_back(key);

    for(int i = 1;i<=n_rounds;i++){
        roundKeys.push_back(keyExpansion(roundKeys[i-1],i));
    }

    addRoundKey(roundKeys[roundKeys.size()-1],block);
    roundKeys.pop_back();

    for(int i = 1;i<n_rounds;i++){
        inverseShiftRows(block);
        subBytes(block,invSBox);
        addRoundKey(roundKeys[roundKeys.size()-1],block);
        mixColumns(block,invColumnTable);
        roundKeys.pop_back();
    }

    inverseShiftRows(block);
    subBytes(block,invSBox);
    addRoundKey(roundKeys[roundKeys.size()-1],block);
    roundKeys.pop_back();

    return block;
}
```

Modo de Operação CTR

Modo de Operação CTR (Counter):

O modo de operação CTR (Counter) é um modo de operação simétrica de cifra por blocos. Ele transforma um cifrador por blocos em um cifrador de fluxo, permitindo a cifragem de dados arbitrariamente longos.

Nesse modo, um contador é criptografado usando a chave do cifrador AES, e o resultado é usado como uma máscara de bits para criptografar (ou descriptografar) os dados. Cada bloco de texto simples é combinado com um valor exclusivo do contador (que normalmente começa em zero e é incrementado a cada bloco), e a operação de OU exclusivo (XOR) é usada para produzir o bloco de texto cifrado.

As principais características do modo CTR incluem:

Paralelismo: O modo CTR permite a cifragem paralela dos blocos de texto simples, o que o torna eficiente para criptografar grandes quantidades de dados.
Não requer preenchimento: Diferentemente dos modos de operação de bloco, como o modo CBC, o modo CTR não requer preenchimento, pois opera como um cifrador de fluxo.
Determinismo: A mesma chave e o mesmo contador sempre produzirão a mesma sequência de cifra, tornando-o determinístico.

Para usar o modo CTR, você geralmente precisa inicializar um contador (normalmente chamado de "nonce" ou "IV" - vetor de inicialização) e garantir que ele seja exclusivo para cada mensagem cifrada. O contador é incrementado para cada bloco de texto simples.

Funcionamento do programa

Entrada do Usuário:

O código começa solicitando ao usuário que escolha uma operação (cifrar um bloco único, decifrar um bloco único ou cifrar um arquivo) e forneça o número de rodadas e a chave de criptografia.

Operações com Bloco Único:

Se o usuário escolher cifrar ou decifrar um bloco único, o código solicitará um bloco de 128 bits em hexadecimal, que é convertido em uma matriz 4x4 chamada block. Depois, ele realiza a operação selecionada e exibe o resultado.

Operação de Cifragem de Arquivo:

Se o usuário escolher cifrar um arquivo, o código solicita o caminho do arquivo de entrada, o caminho do arquivo de saída e um IV (Vetor de Inicialização) em hexadecimal.

O IV é convertido em uma matriz 4x4 chamada IV.

O código usa a função aesCTR para cifrar o arquivo de entrada usando AES-CTR e a chave fornecida. Os blocos cifrados resultantes são armazenados em vector<array<array<uint8_t, 4>, 4>> ciphered.

O tamanho do arquivo de entrada é obtido usando a função fileSize.

O resultado é escrito no arquivo de saída e os blocos gerados são exibidos na saída padrão.

Verificação de Resultados:

O código lê e exibe os dados do arquivo de saída gerado para verificar os resultados.

No código, as funções implementadas em modules/aesOperations.hpp são responsáveis pela cifragem e decifragem AES, enquanto as funções em modules/fileManipulations.hpp são usadas para ler e escrever em arquivos. A função aesCTR é responsável pela cifragem de arquivos usando o modo AES-CTR.

Para um entendimento completo do código, é importante analisar os detalhes das funções definidas nos arquivos de cabeçalho aesOperations.hpp, fileManipulations.hpp e blocks.hpp. Cada uma dessas funções contém a implementação das operações necessárias para a cifragem e decifragem de dados.

Conclusão

Em resumo, o AES é um algoritmo de criptografia robusto e amplamente aceito que oferece segurança confiável para proteger informações confidenciais. Sua flexibilidade, eficiência e histórico de segurança o tornam uma escolha sólida para uma ampla variedade de aplicações de segurança e privacidade. Poder implementar uma solução em C++ é uma experiência boa pois conseguimos programar em um nível mais baixo para manipulação de bytes e ter também um bom uso da aplicação de álgebra no projeto.

Referências:

https://www.ime.usp.br/~rt/cranalysis/AESSimplified

https://en.wikipedia.org/wiki/Rijndael_S-box

csrc.nist.gov/publications/fips/fips197/fips-197.pdf

https://www.sciencedirect.com/science/article/pii/S1875389212005822?ref=pdf_download&fr=RR-2&rr=81e5c397f9841d17

https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf#page=1

https://www.includehelp.com/cryptography/counter-ctr-mode-in-cryptography.aspx

https://xilinx.github.io/Vitis_Libraries/security/2020.1/guide_L1/internals/ctr.html