Jean Bueno Karia - 211055290

github: https://github.com/Jibabk/2025-1-Lista1TAC

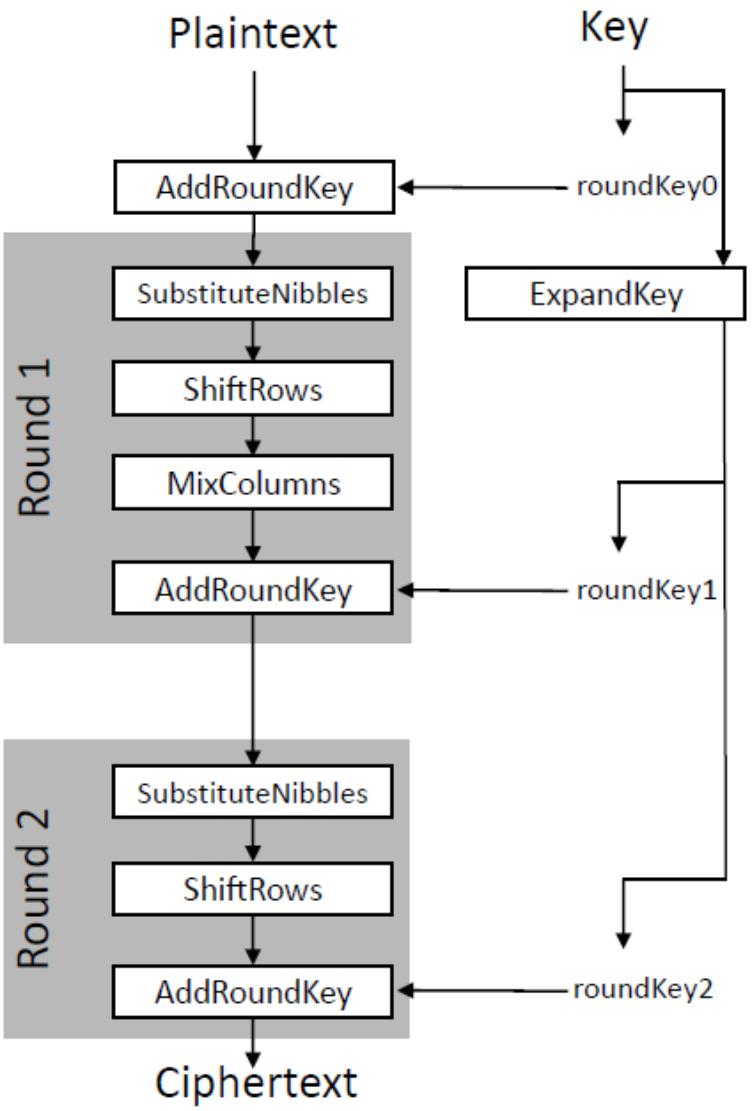
Parte 1 - Implementação do SAES

Introdução

O algoritmo SAES (Simplified AES) é uma versão simplificada do AES, criada para fins acadêmicos. Ele utiliza blocos menores e operações simplificadas, permitindo a compreensão didática dos conceitos de criptografia de blocos e do funcionamento interno do AES real.

Descrição do Algoritmo SAES

A implementação do SAES segue os seguintes passos principais:



- **Adição da Chave de Rodada (AddRoundKey):**
    - Realiza XOR entre a matriz de estado e a subchave correspondente da rodada.
- **Substituição de Nibbles (SubNibbles):**
    - Utiliza uma S-box para substituir cada nibble de 4 bits por outro nibble conforme uma tabela fixa.
- **Deslocamento de Linhas (ShiftRows):**
    - Realiza a troca da posição dos dois nibbles da segunda linha.
- **Mistura de Colunas (MixColumns):**
    - Multiplica cada coluna da matriz de estado por uma matriz fixa no campo GF(2^4), garantindo difusão
- **Expansão de Chave (KeyExpansion):**
    - Gera subchaves de 16 bits a partir da chave original, utilizando uma função g() que aplica a S-box e XOR e o número de rodadas

Descrição da Implementação

A implementação foi realizada em Python, utilizando funções separadas para cada operação principal. Além disso, há suporte para transformação entre representações hexadecimais e vetoriais para facilitar a manipulação de dados.

Exemplos de uso:

Utilizando a chave 1010011100111011 e mensagem “ok” temos como saida:
Texto cifrado em hexadecimal: 738
Texto cifrado em base64: Bzg=
Texto cifrado em binário: 0000011100111000

e valores intermediários de cada round:
Round 0: c850
Round 1: f085
Round 2: 0738

Comparativo: S-AES vs AES

O S-AES (Simplified AES) é uma versão educacional simplificada do algoritmo AES, projetada para facilitar o entendimento dos conceitos fundamentais da criptografia de blocos. Enquanto o S-AES opera com blocos e chaves de 16 bits e utiliza apenas duas rodadas, o AES oficial (definido pelo NIST na FIPS 197) trabalha com blocos de 128 bits e permite chaves de 128, 192 ou 256 bits, realizando até 14 rodadas de transformação. O S-AES utiliza o campo finito $GF(2^4)$, com S-boxes pequenas e operações simplificadas, tornando-o inadequado para aplicações reais. Em contraste, o AES oficial usa $GF(2^8)$, possui estruturas mais complexas como a S-box baseada em inversos multiplicativos e uma expansão de chave robusta, garantindo alta segurança e desempenho, sendo amplamente utilizado em sistemas criptográficos modernos.

Parte 2 - Simulação com SAES usando ECB

Introdução

Nesta etapa, simulamos o uso do SAES que criamos na parte 1 com o modo de operação ECB (Electronic Codebook), onde o texto claro é dividido em blocos independentes de 16 bits. Cada bloco é cifrado separadamente utilizando a mesma chave.

Etapas da implementação

A implementação segue a mesma lógica apresentado anteriormente, sendo a única diferença, o suporte para mensagens maiores. Assim as diferenças principais são:

- **Leitura da Mensagem:**
  - A mensagem lida do terminal não possui mais um limite de tamanho e convertida para uma lista de blocos de 16 bits
- **Ciframento:**
  - Cada bloco da mensagem é cifrado individualmente usando a função encrypt_saes().

Limitações do Modo ECB

Apesar de sua simplicidade, o modo ECB apresenta sérias limitações de segurança, pois blocos idênticos de texto claro geram blocos idênticos de texto cifrado. Isso compromete a aleatoriedade e permite inferências sobre o conteúdo da mensagem cifrada.

Demonstração da geração de cifras iguais para blocos iguais:

Ao utilizar a mensagem “Eu aaaaaaaaaaaaaaaaaaaaaaaaaaaaamo criptografia” com a senha 0000011100111000,temos com resultado do S-AES com modo de operação ECB o seguinte valor em base64 como saída:
9S3D7/TQ9ND00PTQ9ND00PTQ9ND00PTQ9ND00PTQ9NCk1eZ6hJbi+sFZjnZ01/XA

Note que a sequencia “Q9ND00PT” se repete várias vezes e isso se deve justamente pelo fato dos blocos manipulados para serem iguais.

Parte 3 - Simulacao com AES Real usando Bibliotecas Criptograficas Introducao

Introdução

A criptografia AES (Advanced Encryption Standard) e um dos algoritmos mais utilizados no mundo para garantir a seguranca de dados digitais. Ela opera sobre blocos de 128 bits com chaves de 128, 192 ou 256 bits. Este projeto tem como objetivo simular o funcionamento do AES utilizando a biblioteca PyCryptodome, testando os principais modos de operacao: ECB, CBC, CFB, OFB e CTR. As mensagens sao cifradas a partir de um arquivo de entrada, com os resultados sendo salvos em base64, alem da medicao de tempo de execucao para cada modo.

Descricao dos Modos de Operacao AES

A implementação do SAES segue os seguintes passos principais:

- **ECB (Electronic Codebook):**
    - Cada bloco de texto e cifrado independentemente. Rapido, mas inseguro em mensagens longas.
- **CBC (Cipher Block Chaining):**
    - Cada bloco é combinado com o anterior antes de ser cifrado, adicionando segurança ao aumentar a aleatoriedade do processo. Mas isso causa que o processo seja sequencial e mais lento. E utiliza um vetor de inicialização para iniciar a corrente de cifras.
- **CFB (Cipher Feedback):**
    - É processado uma quantidade limitada de bits por vez, e o texto cifrado anterior é usado como entrada para produzir uma saída pseudo-aleatória, que é o XOR com o texto claro para produzir a próxima unidade de texto cifrado. Esse modo é utilizado para a criptografia de mensagens em fluxo de bits, mas ainda não é possível o paralelismo.
- **OFB (Output Feedback):**
    - O output feedback trás uma característica de que apesar dele ainda depender do bloco anterior para realizar a criptografia do próximo bloco, ele não depende da saída desse. Assim permitindo que o cálculo da cifra seja separado em duas partes.
- **CTR (Counter):**
    - Por fim, o Counter mode mostra uma solução que garanta a aleatoriedade para mensagens grandes, mesmo não dependendo de partes anteriores do processo. Assim, permitindo o paralelismo da operação. Isso é possível pelo fato desse modo de operação utilizar um contador para introduzir essa aleatoriedade sequencial.

Descrição da Implementação

A implementacao foi feita em Python utilizando a biblioteca PyCryptodome. As seguintes etapas foram realizadas:

1. Leitura da mensagem do arquivo 'mensagem.txt'. Que contem a imagemOriginal.jpg em base64.
2. Padding com bytes nulos para multiplos de 16.
3. Geracao da chave de 128 bits e IV usando 'secrets.token_bytes(16)'.
4. Implementacao dos modos AES: ECB, CBC, CFB, OFB e CTR
5. Cada resultado foi salvo no diretorio 'cifras' em codificacao base64.
6. O tempo de execucao de cada modo foi medido utilizando 'time.perf_counter()'
7. Foi calculado a entropia de Shannon. Que é uma medida da aleatoriedade de um conjunto de dados.

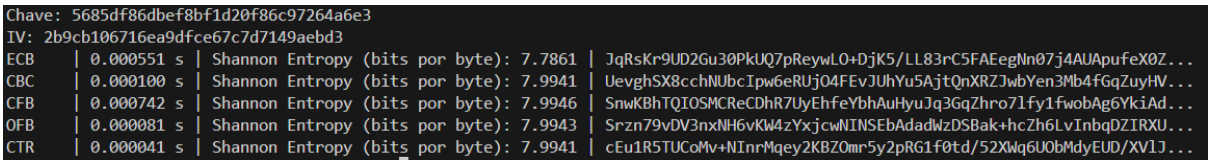
Resultados e Comparacoes

Grau de Aleatoriedade e Seguranca:

Os modos de operacao que utilizam IV ou contador (CBC, CFB, OFB e CTR) produzem resultados diferentes mesmo com a mesma entrada e chave, aumentando

a aleatoriedade. Foram utilizados histogramas e entropia de Shannon para demonstrar esse comportamento. O modo ECB apresentou uma menor aleatoriedade, repetindo padroes da entrada. Sendo que o modo que demonstrou maior eficiência foi o CTR com uma alta taxa de aleatoriedade e baixo tempo de execução.

Tabela comparativa entre modos de operações:



Conclusão:

A simulacao permitiu comparar a eficacia dos diferentes modos do AES em termos de seguranca e desempenho. Foi possivel observar que modos como ECB, embora simples, nao sao recomendados para dados grandes. Modos com IV ou contador, como CBC, OFB e CTR, demonstraram seguranca significativamente superior.