Trabalho 2 - AES
Segurança Computacional - Turma 01
Gabriel Borges - 202006401
Maycon Fabio - 200059742

1. Introdução

O trabalho pode ser dividido em três partes: AES modo ECB, AES modo CTR e testes. A primeira parte está relacionada ao modo de operação da cifra em seu estado puro, enquanto a segunda inclui um contador que evita a repetição de chave para diferentes blocos. O programa pode ser testado tanto em forma de entrada escrita no terminal, quanto para arquivos.

2. Implementação

Existem 4 subprogramas essenciais para implementação da cifração e decifração, são eles: *add_round_key*, *sub_bytes*, *shift_rows* e *mix_columns*. Também existem 5 subprogramas essenciais para o processo de expansão de chave, são eles: *rot_word*, *sub_word*, *rcon*, *ek* e *k*. Cada um recebeu uma função separada dentro do programa para facilitar a legibilidade e organização.

2.1. Expansão de chave

Começando com as mais simples as funções *ek* e *k* basicamente pegam trechos dos bytes da chave, com *k* retornando 4 bytes de um determinado trecho da chave original e *ek* retornando 4 bytes da chave expandida.

Rot_word faz com que um trecho de bytes rotacione para a esquerda, por exemplo, se temos uma sequência “0123”, ao aplicarmos a função teremos “1230”.

Sub_word usa da função *sub_bytes*, que basicamente substituirá os bytes de um trecho por seus correspondentes na tabela *SBOX*.

Rcon retorna um valor de 4 bytes da tabela *RCON* de acordo com o atual *round* em que o algoritmo está.

Rcon(0) = 01000000
Rcon(1) = 02000000
Rcon(2) = 04000000
Rcon(3) = 08000000
Rcon(4) = 10000000
Rcon(5) = 20000000
Rcon(6) = 40000000
Rcon(7) = 80000000
Rcon(8) = 1B000000
Rcon(9) = 36000000
Rcon(10) = 6C000000
Rcon(11) = D8000000
Rcon(12) = AB000000
Rcon(13) = 4D000000
Rcon(14) = 9A000000

Agora o algoritmo passo a passo para uma chave de 16 bytes, seguindo o padrão do AES 128 bits. Os 16 primeiros bytes vem da chave original por meio da função *k*, já contabilizando 4 rodadas, e agora a cada 4 rodadas todas as próximas seguem esse padrão:

…
Rodada 12:
sub_word(rot_word(ek((12-1)*4))) ^ rcon((12/4)-1) ^ ek((12-4)*4)
Rodada 13:
ek((13-1)*4) ^ ek((13-4)*4)
Rodada 14:
ek((14-1)*4) ^ ek((14-4)*4)
Rodada 15:
ek((15-1)*4) ^ ek((15-4)*4)
…

E assim por diante, cada rodada depende da anterior para ser gerada e ao final disso temos nossa chave expandida.

2.2. ECB

Add_round_key nada mais é do que o resultado da operação *XOR* entre os bytes de entrada com os bytes da chave atual, ou seja, a que está sendo usada na respectiva rodada.

Sub_bytes apenas troca os bytes da matriz state com seu correspondente na tabela *SBOX* (cifração) ou *INV_SBOX* (decifração).

Shift_rows realiza o shift circular a esquerda (cifração) ou shift circular a direita (decifração). O deslocamento depende da linha e varia de 0 a 3, sendo 0 para a primeira e 3 para a última.

Mix_columns espalha os bytes da matriz usando operações matemáticas. É feito uma multiplicação utilizando o campo de Galois entre as colunas da matriz state com as linhas da tabela *MIX_MUL_ENC* (cifração) ou *MIX_MUL_DEC* (decifração). Depois, é feita uma operação XOR entre cada resultado, gerando um novo byte para todas as posições da matriz.

2.3. CTR

O modo de operação CTR utiliza o AES puro combinado com um contador somado a um número aleatório denominado *nonce*. Para cifrar, o *nonce* é gerado aleatoriamente como um número entre 0 e 2^64, somado ao contador que também varia nessa faixa e é incrementado de acordo com a quantidade de blocos. O resultado dessa soma serve como entrada para o algoritmo apresentado acima. Então é feita a operação XOR entre o bloco de texto e o criptograma gerado. Para decifrar, basta fazer o mesmo processo utilizando o mesmo *nonce* da cifração, visto que o XOR é uma operação reversível.

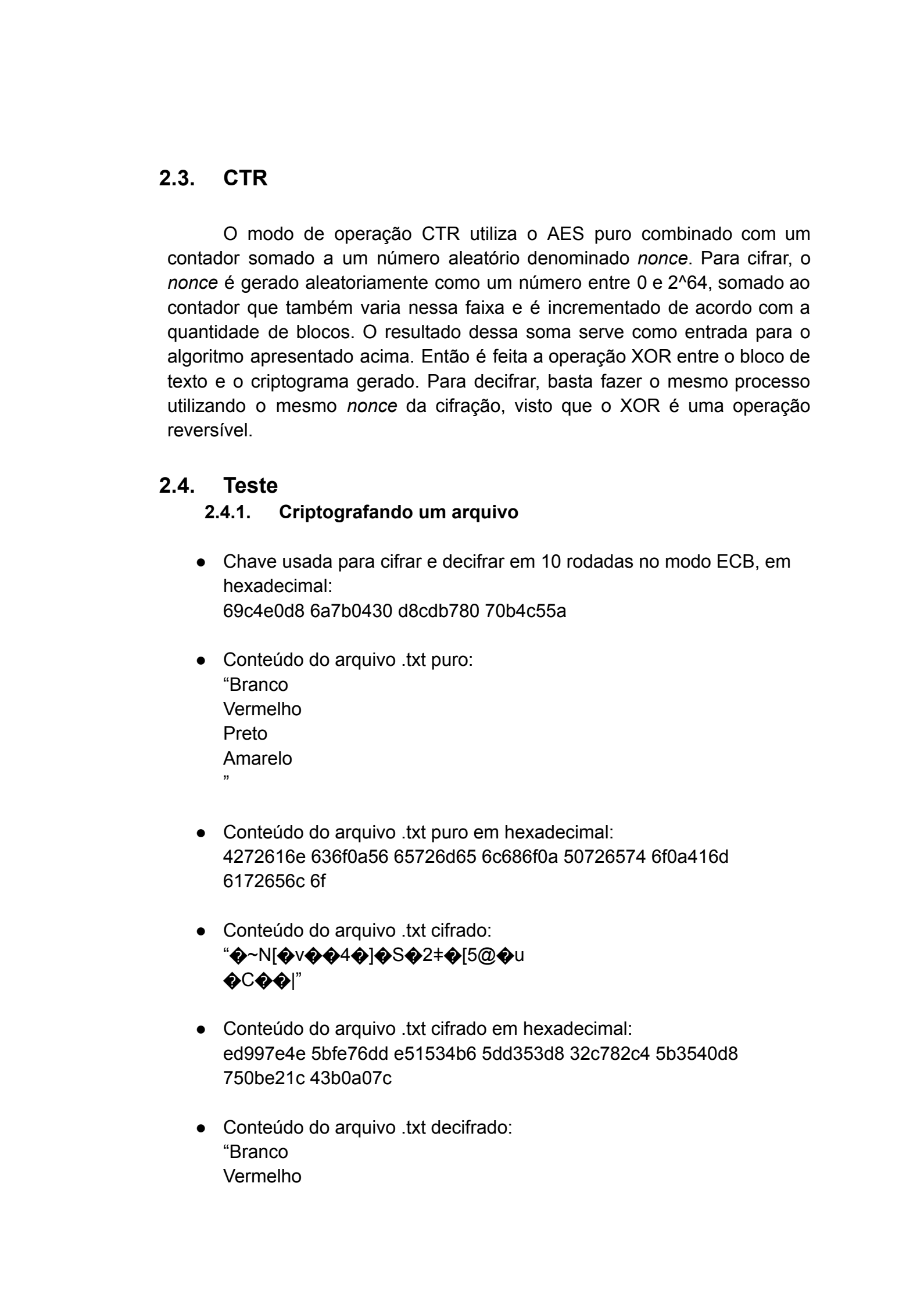
2.4. Teste

2.4.1. Criptografando um arquivo

- Chave usada para cifrar e decifrar em 10 rodadas no modo ECB, em hexadecimal:
  69c4e0d8 6a7b0430 d8cdb780 70b4c55a

- Conteúdo do arquivo .txt puro:
  “Branco
  Vermelho
  Preto
  Amarelo
  ”

- Conteúdo do arquivo .txt puro em hexadecimal:
  4272616e 636f0a56 65726d65 6c686f0a 50726574 6f0a416d 6172656c 6f

- Conteúdo do arquivo .txt cifrado:
  “ 

Preto
Amarelo
”

- Conteúdo do arquivo .txt decifrado em hexadecimal: 4272616e 636f0a56 65726d65 6c686f0a 50726574 6f0a416d 6172656c 6f

3. Referências

https://www.kavaliro.com/wp-content/uploads/2014/03/AES.pdf
https://www.ime.usp.br/~rt/cranalysis/AESSimplified
https://cs.ru.nl/~joan/papers/JDA_VRI_Rijndael_2002.pdf