Trabalho de Implementação 2

Segurança Computacional - Turma B

Leonardo Rodrigues de Souza - 17/0060543
Lucas Dalle Rocha - 17/0016641

Universidade de Brasília - UnB {170060543,170016641}@aluno.unb.br

1 Descrição da cifra AES e modos ECB/CTR

O segundo trabalho da disciplina de Segurança Computacional, da Universidade de Brasília, consistiu na implementação da cifra de bloco AES, seus modos de operação ECB e CTR e testes realizados em imagens de extensão PPM. O algoritmo utilizado pelo AES é de chave simétrica e se baseia em manipulação de matrizes de blocos 4x4, que representam parte do texto limpo a ser cifrado (16 bytes) e a chave escolhida pelo usuário (16 bytes). A partir da chave, pode-se escolher a quantidade de rodadas que se deseja realizar sobre o bloco, de modo que com o aumento de rodadas, há um aumento da difusão da cifra.

1.1 Cifração AES

Para cifrar um texto usando a cifra de bloco AES, o algoritmo executa uma sequência de etapas: 1. *KeyExpansion*, em que o algoritmo gera chaves distintas para serem utilizadas em cada rodada, baseado em constantes de arredondamento e cálculos realizados no campo de *Galois* [2]; 2. *AddRoundKey*, em que é realizado a operação XOR entre os bytes do estado e a primeira instância da chave; 3. *SubBytes*, que utiliza uma matriz de substituição para fornecer a confusão da cifra, [3] *ShiftRows* e *MixColumns*, que desloca certas linhas da matriz e realiza a multiplicação da matriz pelo polinômio irredutível $x^8+x^4+x^3+x+1$, respectivamente, isto é, operam sobre os bytes da cifra para fornecer a difusão, e *AddRoundKey* da rodada atual, até a penúltima rodada; 4. *SubBytes*, *ShiftRows* e *AddRoundKey* para a última rodada. Assim, tem-se o bloco inicial cifrado a partir do número de rodadas escolhido.

1.2 Decifração AES

Para decifrar, utiliza-se a sequência inversa da cifragem, isto é: 1. *KeyExpansion* (o mesmo utilizado na cifragem); 2. *AddRoundKey* da última rodada, acompanhado das operações inversas de *ShiftRows* (retorna as linhas deslocadas) e *SubBytes* (utiliza-se a matriz de substituição inversa); 3. repete-se até a penúltima rodada as etapas de *AddRoundKey* (das chaves de maior rodada para as de menor rodada), *MixColumns* (a matriz utilizada é a constante inversa), *ShiftRows* e *SubBytes*; 4. *AddRoundKey* da chave inicial. Dessa forma, obtém-se o texto limpo, dado número de rodadas que foi cifrado.

1.3 Cifração ECB/CTR

Para cifração no modo ECB, o texto limpo é dividido em blocos de 128 bits e, dada cifragem pela utilização do AES, o criptograma obtido é concatenado em blocos na mesma posição do texto limpo. Já para a cifração no modo CTR, utiliza-se um contador, em bytes, que é incrementado a cada bloco a ser cifrado. Então, o contador é utilizado no processo de cifração do AES e sua saída é operada XOR com o bloco a ser cifrado, de modo a se obter o criptograma, e o contador ser incrementado para a cifragem do bloco posterior.

1.4 Decifração ECB/CTR

Para decifração no modo ECB, o criptograma também e dividido em blocos de 128 bits e a decifração é realizada pela utilização do AES, com a concatenação do texto limpo obtido, em ordem. Por fim, a decifração no modo CTR é extremamente simples: aplica-se o mesmo processo da cifração, sem que haja necessidade de modificar as etapas para a operação inversa, e o contador é incrementado da mesma maneira.

2 Implementação da cifra AES e modos ECB/CTR

2.1 Etapas de cifração AES

Para a implementação do processo de cifração em código, separou-se a matriz de estado (16 bytes) e a matriz da chave inicial (16 bytes), e, em seguida, houve inicialização das estruturas utilizadas na expansão da chave (tabela de substituição de bytes e constantes de arredondamento). Inicialmente, a chave é expandida, dado número de rodadas passado pelo usuário e utilização das estruturas supracitadas, através da rotação da última coluna da matriz, seguido da substituição pela S-BOX correspondente. Faz-se então a operação XOR com a matriz da constante de arredondamento e um XOR final com a primeira coluna da rodada passada (as demais colunas são apenas operadas XOR com as colunas respectivas da rodada passada). Aplica-se, então, a etapa de *AddRoundKey* pela utilização da chave inicial, seguida da sequência circular $SubBytes \rightarrow ShiftRows \rightarrow MixColumns \rightarrow AddKeyRound$ pelo número de rodadas escolhido pelo usuário, menos a última. É importante ressaltar que na etapa de difusão das colunas, caso fosse detectado *overflow* na multiplicação, o byte respectivo era operado XOR com $0x11B$, a fim de se manter no campo de *Galois*. Por fim, remove-se a etapa de difusão das colunas e aplica-se as supracitadas para a última rodada da cifragem, repetindo o processo para os blocos posteriores até que se obtenha o criptograma por completo. Vale informar que bytes de *padding* são adicionados na cifração do último bloco, a fim de que se complete a matriz 4x4, mas retirados durante a escrita do criptograma por completo.

2.2 Etapas de decifração AES

Já para a implementação do processo de decifração em código, há o recebimento do criptograma em matrizes de 16 bytes, assim como a chave inicial, e também é realizado o processo de expansão da chave. Desse modo, o processo aplicado é o inverso da cifração, bem como as etapas devidamente modificadas: *ShiftRows* retorna as linhas deslocadas no processo de cifração, *SubBytes* utiliza a S-BOX inversa (retorna os bytes substituídos por ela para processo de confusão) e *MixColumns* utiliza o polinômio inverso do processo de cifração, no campo de *Galois*, e assim como na cifragem, quando se identifica um *overflow* de algum byte, executa-se a operação XOR com $0x11B$. E, de modo semelhante a cifração, utiliza-se os mesmos bytes de *padding*.

2.3 Etapas de cifração ECB/CTR

Para a implementação do modo ECB, não foi necessário implementar alteração alguma ao AES, uma vez que os blocos já eram divididos em 16 bytes e esquematizados em uma matriz, ou seja, o modo apenas utiliza o AES puro por um número de rodadas escolhido pelo usuário, e armazena o criptograma em sequência. Já para o modo CTR, foi necessário implementar uma matriz que armazena o contador (inicialmente foi escolhido todos os bytes como $0x00$) e o AES é aplicado sobre esse contador. Em seguida, é realizado o processo final da operação XOR entre o texto puro a ser cifrado e o contador, para se obter o criptograma.

2.4 Etapas de decifração ECB/CTR

Para a implementação do modo ECB, também não foi necessário implementar alteração ao AES, pelo mesmo motivo da cifração. E o modo CTR, por sua vez, apresenta processo de decifração igual ao processo de cifração, inclusive pela utilização das mesmas etapas de cifração (não se aplica etapas inversas, como utilização da S-BOX inversa), e se obtém o texto puro pela concatenação final dos bytes restaurados.

3 Instruções para compilação e execução

Para a compilação e execução, foi utilizado o sistema operacional Ubuntu 20.04.2 LTS, além do *g++* versão 9.4.0 e GNU Make 4.2.1. O programa pode ser compilado pela utilização do *Makefile*, com o seguinte comando em seu terminal:

```
$ make compile
```

Ademais, pode-se compilar e executar o AES no arquivo de teste disponibilizado para testes de rápida execução (*tests/penguin.ppm*), em conjunto com a utilização do *Makefile*, para cifração da imagem e decifração do criptograma gerado logo em seguida, para averiguação da corretude do algoritmo, através da chave de teste de 16 bytes, em hexadecimal, $a5b90c1a2b5e3a014253698745214523$:

```
$ make run_ecb1
$ make run_ecb5
$ make run_ecb9
$ make run_ecb13
$ make run_ctr1
$ make run_ctr5
$ make run_ctr9
$ make run_ctr13
```

Caso seja de seu interesse compilá-lo manualmente, execute a instrução no diretório raiz:

```
$ g++ -o aes src/main.cpp
```

Por fim, para execução do AES nos arquivos teste:

```
$ ./aes [-c, -d] <key> -f <source_file>
-o <destination_file> [-ecb, -ctr] <num_rounds>
```

Exemplo:

```
$ ./aes -c a5b90c1a2b5e3a014253698745214523 -f tests/penguin.ppm
-o tests/penguin_encrypted.ppm -ecb 1
```

4 Resultado dos testes

Inicialmente, foi-se escolhido uma imagem de extensão PPM que consta os dois autores do projeto:

Figura 1. Imagem original (texto limpo).

Para a cifração do modo ECB, foi feito o teste com rodadas 1, 5, 9 e 13, apresentadas abaixo:

Para a cifração do modo CTR, foi feito o teste com rodadas 1, 5, 9 e 13, apresentadas abaixo (foi utilizado o contador iniciado em $0x00$):

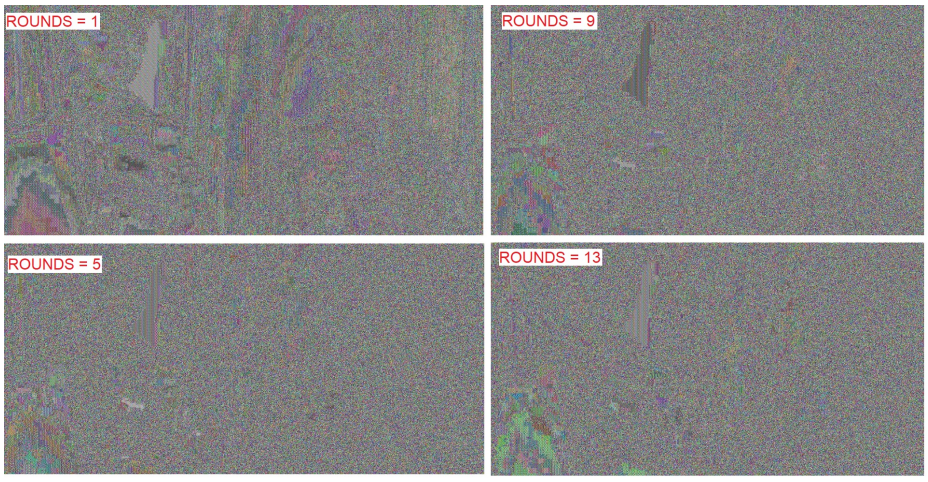

Figura 2. Imagens cifradas em AES e modo ECB, para rodadas 1, 5, 9 e 13.

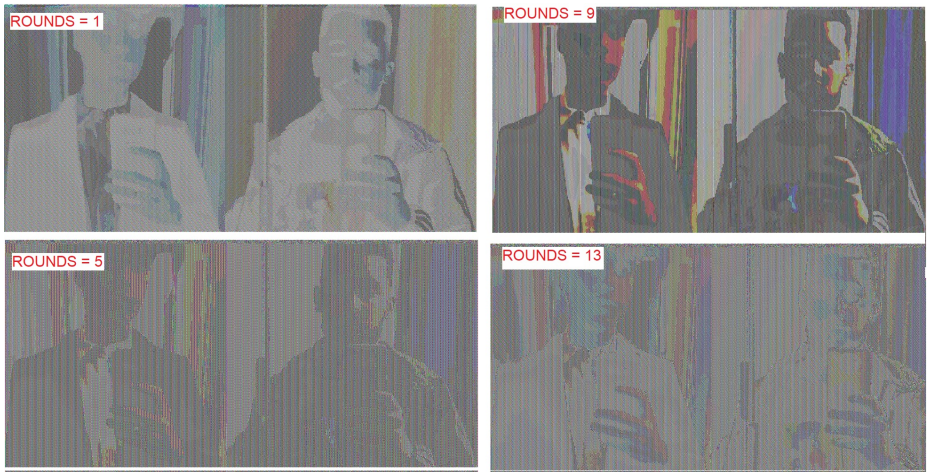

Figura 3. Imagens cifradas em AES e modo CTR, para rodadas 1, 5, 9 e 13.

Referências

1. Wikipedia contributors, Wikipedia, The Free Encyclopedia., Advanced Encryption Standard, https://en.wikipedia.org/wiki/Advanced_Encryption_Standard, Last accessed 14 Sep 2021
2. Wikipedia contributors, Wikipedia, The Free Encyclopedia., AES key schedule, https://en.wikipedia.org/wiki/AES_key_schedule, Last accessed 14 Sep 2021
3. Wikipedia contributors, Wikipedia, The Free Encyclopedia., Rijndael S-box, https://en.wikipedia.org/wiki/Rijndael_S-box, Last accessed 14 Sep 2021