Cifra de Bloco AES e Modos de Operação

Rodrigo L. da Silveira

Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
90.619-900 – Porto Alegre – RS – Brasil

rodrigo.silveira@edu.pucrs.br

Abstract. *This report aims to study and understand the AES block cipher and two of its modes of operation - CBC and CTR - resulting in the implementation of a program to encrypt and decipher the sample data provided in the statement of work 2.*

Resumo. *Este relatório tem por objetivo o estudo e a compreensão da cifra de bloco AES e dois de seus modos de operação – CBC e CTR – resultando na implementação de um programa capaz de cifrar e decifrar os dados de exemplos fornecidos no enunciado do trabalho 2.*

1. Introdução

O problema proposto no segundo trabalho da disciplina de Segurança de Sistemas consiste na implementação de um programa capaz de cifrar e decifrar dados utilizando o método de criptografia AES e os modos de operação CBC e CTR. Foram disponibilizados quatro textos cifrados e dois textos claros, com suas respectivas chaves, para a validação do programa. Os textos cifrados devem ser decifrados e os textos claros cifrados. O vetor de inicialização (IV) utilizado para gerar os textos cifrados encontra-se nos primeiros 16 bytes do texto.

2. AES (*Advanced Encryption Standard*)

O método de criptografia AES é uma cifra de bloco, com chave simétrica, estabelecida pelo Instituto Nacional de Padrões e Tecnologia do Estados Unidas (NIST) em 2001 para substituir o DES, seu antecessor. O tamanho da chave pode ser de 128 bits, 192 bit ou 256 bits, e o tamanho do bloco é sempre de 128 bits.

3. Modos de Operação

O AES trabalha com blocos de dados de tamanho fixo, geralmente 128 bits, porém, as mensagens a serem cifradas não necessariamente tem este mesmo tamanho ou um tamanho múltiplo dele. Para resolver a primeira situação, a mensagem será dividida em blocos de tamanho fixo de 128 bits e cada bloco será criptografado individualmente. Para o caso em que o tamanho da mensagem não seja um múltiplo de 128 bits, o último bloco ficará incompleto, quando isto ocorrer, o modo de operação selecionado irá preencher o bloco com os bits necessários para que possa ser cifrado pelo AES (*padding*). Os diferentes modos de operação possuem ainda algumas características, onde pode ser adicionado uma camada a mais de segurança com o modo de encadeamento dos blocos cifrados ou uma melhor performance na decifração com a paralelização desta etapa. A maioria dos modos de operação necessita de um vetor de inicialização (IV) que não precisa ser secreto, mas que nunca deve ser utilizado com a

mesma chave para não fornecer informações a respeito dela. Os modos de operações abordados neste trabalho serão CBC e CTR.

3.1. CBC (*Cipher Block Chaining*)

O CBC utiliza o bloco cifrado anterior para gerar o novo bloco cifrado, através da operação ou-exclusivo (XOR). Por este motivo faz-se necessário a utilização de um vetor de inicialização para cifrar o primeiro bloco. Neste modo de operação, a técnica de *padding* utilizado para completar o último bloco consiste na inserção de N bytes de valor N, onde N é o número de bytes necessários para completar o bloco, conforme exemplo a seguir.

Tabela 1. Exemplo de *padding* do CBC

Número de bytes faltando	Preenchimento
1	1
2	2 2
3	3 3 3
10	A A A A A A A A A A

3.2. CTR (*Counter*)

O CTR utiliza um contador somado ao IV, o resultado é cifrado e então é feita uma operação de ou-exclusivo (XOR) com o bloco a ser cifrado. O contador pode ser uma função, desde que ela leve muito tempo para repetir números. Este modo de operação possui a característica de não necessitar do bloco anterior para decifrar um bloco, apenas saber qual o contador foi utilizado ao cifrar, isto permite que esta operação seja feita de forma paralela, melhorando a performance.

4. Desenvolvimento

A implementação do programa proposto pelo presente trabalho foi realizada em linguagem de programação Java em um ambiente Windows. Para a cifrar e decifrar os dados foi utilizada a classe *Cipher* do pacote javax.crypto disponibilizado pela própria plataforma. Os casos de teste foram importados de um arquivo *Json* para a classe Tarefa, que possui as propriedades necessárias para a criptografia dos dados.

A classe AES possui como configuração inicial os parâmetros de modo de operação e a chave a ser utilizada e possui os métodos *encrypt* e *decrypt* para cifrar e decifrar respectivamente. Ambos os métodos recebem como parâmetro um texto. O método *encrypt* espera receber um texto claro com o vetor de inicialização anexado antes a frente da mensagem. Ele faz a identificação deste vetor e inicializa a classe *Cipher* com o modo de operação, a chave informada e o vetor de inicialização. A classe *Cipher* cifra o texto claro e retorna o texto cifrado. O método *decrypt* funciona de forma semelhante ao *encrypt*, exceto por esperar um texto cifrado como parâmetro e retornar um texto claro. O vetor de inicialização utilizado para cifrar o texto também é esperado anexado no início da mensagem.

O programa gerado é executado via linha de comando, podendo ser interativo ou automático, caso seja passado algum argumento via linha de comando, este argumento será tratado como o número do caso de teste a ser executado. Se nenhum argumento for informado, o programa irá mostrar as opções de testes e esperar que o usuário informe uma opção. Após o programa irá retornar o resultado a operação selecionada.

5. Resultados

Após a execução dos 6 casos de testes, os resultados obtidos, conforme esperado, foram os seguintes:

1. Basic CBC mode encryption needs padding.
2. Our implementation uses rand. IV
3. CTR mode lets you build a stream cipher from a block cipher.
4. Always avoid the two time pad!
5. 7A5A59C3C41CE0E086345B9220C1F029DFB9C2D8927D650232F6E4250DFC0CF1BB290946C6F3D33B5B62ACF93409BC45AB70AA1B962C00A8B0BB
6. 60D73DBEF0DAD0461D175BDA54EF8C9C5A41351E46AEA0BD348E61B8328DB8C351D342237E185ABEB0FD5A3CE13AE562BDD2F7A82F101388DCFD28730EFBF240219733B7D485E6ACB579AB0F520F008A295D6744A0DB0364925CE86B371186DDD339E08167DDE06592BC8125CF7B245B5A0671AFC15960F0E7ED1104D8131E34

References

Silveira, R. L. AES_T2. https://github.com/RLSilveira/AES_T2. (acesso em 02 de maio de 2019)