Universidade Federal do Ceará – UFC

Aluno: Natã Santana de Morais

Número: 383808

Curso: Ciência da Computação

Disciplina: Criptografia

Implementação do algoritmo de encriptação CBC, CTR e GCM

Crateús – CE

10/10/2017

Introdução

Na criptografia, a encriptação é o processo de transformar uma mensagem original em uma mensagem ilegível, para terceiros. Também existe o conceito de decriptação, que nada mais faz o inverso da encriptação. Ou seja, transforma uma mensagem ilegível em uma mensagem legível e original. Ambas utilizam o conceito de chave. Nessa mesma área de estudo, existem os algoritmos criptográficos simétricos e assimétricos.

O AES (Advanced Encryption Standard) já é um dos algoritmos mais populares usados na criptografia de chave simétrica. O mesmo é uma cifra que tem tamanho fixo de 128 bits e uma chave de tamanho 128, 192 ou 256 bits.

Mas, podemos generalizar para qualquer tamanho. Para isso, existem modos de operação que permitem encriptar mensagens de qualquer tamanho de forma segura. Alguns modos de operação são: CBC, CTR e GCM.

Objetivos

O objetivo da implementação do CBC, CTR e GCM é conhecer e compreender os modos de operação que utilizam o AES como cifra. Aprender sobre o funcionamento desses modos de operação é de extrema importância quando o assunto é criptografia simétrica. Além disso, será importante para o aprendizado na manipulação dos conceitos teóricos de encriptação e decriptação utilizando o AES com a parte prática.

Desenvolvimento

A proposta dada foi a implementação dos modos de operação CBC, CTR e GCM. Primeiramente, precisaríamos de uma implementação do algoritmo da cifra do AES, já que a intersecção entre os três modos de operação é o próprio AES. Ou seja, os três modos utilizam a cifra AES como cifrador de bloco padrão. Foi proposto pelo professor, uma cifra do AES em linguagem C. Portanto, o restante da implementação seria em linguagem C.

A implementação foi dividida em três fases.

Na fase 1, foi implementado o CBC. Criou-se uma função que recebia como parâmetros: a mensagem de entrada, a chave, um local para armazenamento da mensagem de saída, o tamanho da mensagem a sofrer encriptação e um vetor de inicialização.

Dentro da função, realizamos algumas operações. Primeiramente, atribuímos a chave do parâmetro a variável de chave da cifra do AES, e logo após, expandimos a chave, se necessário. Após isso, realizamos uma interação que era incrementada em 16, ou seja, a mensagem iria ser lida em blocos de 16 bytes, no caso 128 bits.

Para cada bloco de 16 bytes, realizamos um XOR do bloco corrente da mensagem de entrada com o vetor de inicialização. Após isso, o resultado do XOR é associado a um estado do AES e é encriptado com a corrente cifra. O vetor de inicialização é atualizado com o resultado da encriptação.

Temos um caso particular, quando a mensagem tem um tamanho que não é múltiplo de 16. Quando isso ocorre, calculamos o resto da divisão do tamanho da mensagem por 16. Logo após, inserimos zeros (0x0) para completar o bloco que tem de ser de 128 bits. Acontecido isso, realizamos as mesmas operações já descritas.

No processo de decriptação, guardamos em uma variável, a entrada encriptada, e após isso realizamos o inverso da encriptação do AES. Após isso, é realizado um XOR entre o resultado do inverso da encriptação com o vetor de inicialização, gerando assim uma parte do texto claro. Depois de todas as interações, incluindo o caso particular já descrito, é dado como resultado o texto claro completo.

Na fase 2, foi implementado o CTR. As características de bloco são as mesmas do CBC, ou seja, a mensagem será quebrada em bloco de 16 bytes.

Criou-se uma função que recebia como parâmetros: a mensagem de entrada, a chave, um local para armazenamento da mensagem de saída, o tamanho da mensagem a sofrer encriptação e um vetor de inicialização.

Dentro da função, realizamos algumas operações. Primeiramente, atribuímos a chave do parâmetro a variável de chave da cifra do AES, e logo após, expandimos a chave, se necessário. As interações serão incrementadas em 16, ou seja, encriptando blocos de 16 bytes.

Depois disso, guardamos o vetor de inicialização em uma variável auxiliar, e a mesma sofre a encriptação do AES. Com o resultado da encriptação é feito um XOR com a mensagem clara (16 bytes). Depois, incrementa-se em uma unidade o vetor de inicialização.

O processo é feito bloco a bloco. Se o tamanho não for múltiplo de 16, tiramos o resto da divisão e alocamos o restante da mensagem a ser encriptada em uma variável. Inserimos zeros, até que se complete 16 bytes. As operações após isso são as mesmas já descritas.

A decriptação utiliza os mesmos procedimentos, a diferença é que a entrada será o texto encriptado.

Na fase 3 e última, implementamos o GCM. As características de bloco, assim como o CTR e CBC, são de 16 bytes. O GCM usa operações semelhantes aos outros dois para sua encriptação, mas utiliza o conceito de autenticador.

Criou-se uma função que recebia como parâmetros: a mensagem de entrada, a chave, um local para armazenamento da mensagem de saída, o tamanho da mensagem a sofrer encriptação, um vetor de inicialização, um vetor AAD e um vetor T vazio, que posteriormente receberá um bloco de 16 bytes com a autenticação.

Dentro da função, realizamos algumas operações. Primeiramente, atribuímos a chave do parâmetro a variável de chave da cifra do AES, e logo após, expandimos a chave, se necessário. Depois disso, guardados o vetor de inicialização em uma variável CTR0 e logo após incrementamos o vetor de inicialização. As interações serão incrementadas em 16, ou seja, encriptando blocos de 16 bytes.

Aqui começaremos o processo que vai gerar nossa autenticação. Inicializamos um vetor H de 16 bytes com 0 e depois encriptamos o H. Feito isso, multiplicamos, em Galois field, o vetor H encriptado com o vetor AAD. Essa multiplicação é guardada na variável G, que irá ser o nosso autenticador.

O processo de encriptação da mensagem ocorre encriptando o vetor de inicialização e logo após é realizado um XOR com a mensagem de entrada.

Voltando a nossa autenticação, é realizado um XOR de G com o texto encriptado. O resultado disso é multiplicado em Galois fiel com H e atualiza o G.

Depois disso, o vetor de inicialização é incrementado.

O processo de encriptação quando a mensagem não é múltipla de 16 é semelhante ao CBC e CTR, mas, claro, usando as operações do GCM.

Ao final, encriptamos o CTR0 e realizamos o XOR do CTR0 encriptado com o G. Daí, atribuímos G ao T.

O processo de decriptação é semelhante. A diferença é que o XOR do G é com a entrada da mensagem encriptada. Para saber se a mensagem não foi alterada é só verificar se o T gerado na encriptação é o mesmo T gerado na decriptação.

Dificuldades encontradas

A implementação de modos de operações criptográficos não foi uma tarefa fácil. Foram encontradas algumas dificuldades.

Uma delas foi a compreensão do código da cifra do AES. Os problemas que provocaram isso foram: código muito extenso, muitas operações envolvidas e principalmente as operações de deslocamento de bits. A versão pode parecer simplificada, mas não o suficiente para ser autoexplicativa ou bem comunicativa.

Outra dificuldade foi o controle na manipulação de ponteiros. Quando lidamos com vetores, bytes, e criptografia é comum, na linguagem C, usarmos ponteiros. Ponteiros são uma arma poderosa, mas se não são bem usadas pode acarretar dores de cabeça.

Uma outra dificuldade foi a multiplicação em Galois Field (2^128). Essa operação necessitaria de uma complexidade um pouco maior. Uma solução que resolvemos utilizar foi a multiplicação em Galois Field (2^8), que é utilizada pela cifra do AES, para cada byte dos 16 do bloco.

Conclusão

A implementação dos modos de operação CBC, CTR e GCM foram de grande impacto e foi de crescente aprendizado. Apesar das dificuldades encontradas ou erros que podem eventualmente acontecer, mesmo que nos testes tenham dado certo, podemos afirmar que os resultados foram satisfatórios.