Integridade de Mensagens com SHA-256

Israel Deorce Vieira Júnior[1]

[1] Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul
Porto Alegre, RS - Brazil

{israel.deorce@acad.pucrs.br}

Resumo. *Este relatório descreve uma alternativa de implementação para o exercício proposto no segundo trabalho da disciplina de Segurança de Sistemas. O exercício envolve o desenvolvimento de um programa que dado um arquivo de video, que seja este separado em blocos de bytes e que se garanta a integridade destes utilizando o "Algoritmo de Hash Seguro"* SHA-256*. Desenvolveu-se a solução proposta em linguagem Java utilizando-se de bibliotecas como a* $FileInputStream$ *para leitura de dados, e a* $MessageDigest$ *para geração da* $hash$*. A solução apresentada é de forma textual explicativa, acompanhada de pseudocódigo.*

1. Introdução

Uma função de $hash$ aceita uma mensagem de tamanho variável M como entrada e produz um valor de $hash$ de tamanho fixo $h = H(M)$. Em termos gerais, o objetivo principal de uma função de $hash$ é a integridade de dados. Uma mudança em qualquer bit ou bits em M resulta, com alta probabilidade, em uma mudança no código de $hash$ [Stallings 2013].

O programa descrito neste relatório, aplica a função $hash$ SHA-256 em blocos de dados que compõem um vídeo, sendo cada bloco composto por 1024 bytes de dados, menos o último que comparta a quantidade exata restante se este for menor que 1024 bytes. Os valores $hash$ gerados possuem tamanho fixo de 32 bytes.

2. Leitura do Arquivo de Vídeo

De forma simples, a leitura do vídeo foi feita utilizando a biblioteca $FileInputStream$. A principio, buscou-se ler o arquivo diretamente de traz para frente separando em blocos de 1024 bytes de dados com exceção do último e realizando os cálculos de $hash$. Porém, após muitas tentativas sem sucesso, tomou-se a decisão de realizar a leitura de dados partindo do ponto inicial até o fim do vídeo, armazenando os dados para processamento.

Criou-se uma classe Java denominada "Bloco", que representa um bloco de tamanho menor ou igual a 1024 bytes, que armazena também os dados de uma função $hash$, e que é capaz de calcular a sua própria função $hash$ concatenando os valores dos dados e da $hash$ recebida (se houver). Armazenou-se esses blocos em uma lista, para só então percorrer esta lista fazendo os cálculos de $hash$ de trás para frente.

Esta solução não é a mais ideal, pois faz uso de muita memória, mas devido a nossa pouca experiência em trabalhar com a biblioteca $FileInputStream$, esta foi a melhor solução encontrada. O seguinte algoritmo descreve a leitura:

```
1  procedimento lerArquivo
2    enquanto fileInputStream.disponivel() ≠ 0 faca
3      se fileInputStream.disponivel < 1024 faca
4            ultimoBuffer = new ultimoBuffer[fileInputStream.disponivel]
5            fileInputStream.ler(ultimoBuffer)
6            bloco ← novo bloco(ultimoBuffer)
7            senao faca
8                 fileInputStream.ler(buffer)
9            bloco ← novo bloco(buffer)
10           fim se
11           blocos.adiciona(bloco)
12   fim enquanto
13  fim
```

3. Calculando as Hashes com SHA-256 e Gerando h0

Com a lista de blocos preenchida, realizou-se a leitura da mesma de traz para frente, com intuito de, calcular o valor $hash$ do último bloco utilizando a biblioteca $MessageDigest$, e passar este valor para o penúltimo, concatenando seus dados com a $hash$ do bloco posterior, e assim por diante até a saída do valor da $hash$ do primeiro bloco de vídeo, o h0. Ao final, encontro-se o h0 referente ao vídeo $video05.mp4$, e este é descrito abaixo junto do pseudocódigo do algoritmo utilizado:

8e423302209494d266a7ab7e1a58ca8502c9bfdaa31dfba70aa8805d20c087bd

```
1  procedimento hashVideo
2    hash ← nulo
3      para i ← blocos.tamanho ate ⩾ 0
4      enquanto fileInputStream.disponivel() ≠ 0 faca
5        se hash == nulo faca
6          hash ← blocos[i].calculaHash
7        senao faca
8          blocos[i].adicionaHash(hash)
9          hash ← blocos [i].calculaHash
10       fim se
11     fim para
12    retorna hash
13 fim
```

4. Conclusão

Este relatório apresentou os resultados obtidos na realização da atividade da disciplina de Segurança de Sistemas. Conclui-se que, os algoritmos utilizados estão corretos e o valor h0 foi calculado e encontrado com sucesso, como requerido no enunciado do trabalho. Pode-se concluir também, que o trabalho contribuiu para o avanço do nosso conhecimento a respeito de integridade de mensagens em segurança de sistemas.

Referências

Stallings, W. (2013). *Cryptography and Network Security: Principles and Practice*. Prentice Hall Press, Upper Saddle River, NJ, USA, 6th edition.