Criptografia - SHA-256

Larissa Fiorini Martins

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

larissa.martins@edu.pucrs.br

Abstract. *This article describes an implementation of a program that receives a video file and ensures the integrity of the video data using Secure Hash Algorithm SHA-256. The program behavior will be detailed, with a description of the main functions implemented and then the results obtained will be shown.*

Resumo. *Este artigo descreve uma implementação de um programa que dado um arquivo de vídeo, seja garantida a integridade dos dados do vídeo utilizando o algoritmo de hash seguro SHA-256. Será detalhado o funcionamento do programa, com uma descrição das principais funções implementadas e, por fim, serão mostrados os resultados obtidos.*

1. Introdução

Uma função *hash* é uma função matemática que converte um valor numérico de entrada em outro valor numérico compactado. A entrada da função *hash* pode ser de qualquer tamanho, mas a saída será sempre de tamanho fixo. São resistentes a colisão, ou seja, não há duas entradas que podem ser mapeadas para o mesmo hash de saída, dessa forma, um atacante não consegue modificar nenhum dos blocos de vídeo sem ser detectado.

As funções *hash* funcionam como uma assinatura digital que possibilita garantir a integridade do arquivo, pois é possível validar que os dados não foram alterados após seu envio visto que uma pequena alteração na mensagem geralmente produz uma grande alteração no *hash* resultante (IBM, 2019). Comparando o hash computado (a saída de execução do algoritmo) a um valor de hash conhecido e esperado, é possível determinar a integridade dos dados.

Este trabalho descreve uma implementação em Java, utilizando o algoritmo *SHA-256* (*Secure Hash Algorithm*) disponibilizado por uma biblioteca da linguagem. Serão descritos os detalhes de implementação, e, ao final, serão mostrados os resultados obtidos.

2. Implementação

A primeira etapa corresponde à leitura do arquivo de vídeo, que foi realizada no método *"divideBlocos"*, utilizando a biblioteca *BufferedInputStream* do *Java*. O arquivo de vídeo é então dividido em blocos de 1KB (1024 *bytes*), sendo que se o tamanho do arquivo não for um múltiplo de 1KB, então o último bloco será de tamanho menor, mas todos os demais blocos terão tamanho exato de 1KB. Os blocos são então armazenados em em um *ArrayList* de *bytes* para serem processados posteriormente, sendo que o primeiro elemento do *ArrayList* corresponde ao último bloco de dados do vídeo.

A segunda etapa é realizada no método *"calculaHash",* o qual é responsável por calcular o *hash* para cada bloco. Para o cálculo do *hash SHA-256* foi utilizada a biblioteca *Message Digest* do *Java,* conforme exemplo abaixo:

```
"MessageDigest md = MessageDigest.getInstance("SHA-256");"
```

Figura 1. Utilizando *SHA-256* em Java

Este método irá receber os blocos de *bytes* do vídeo e iniciar calculando o *hash SHA-256* do último bloco. Após, irá anexar o valor do hash calculado para o bloco atual no bloco anterior a ele. Então, é calculado o hash desse bloco anterior a ele e anexado o resultado no antepenúltimo bloco. E assim sucessivamente, percorrendo o *ArrayList* de blocos de trás para frente, do último bloco até chegar no primeiro bloco "*h0*". Se for o primeiro bloco, então irá encerrar a execução e imprimir o hash final "*h0*" calculado.

3. Resultados

Foram disponibilizados dois vídeos para a implementação deste trabalho, sendo que o *hash "h0"* do*"video_05"* foi disponibilizado no enunciado do trabalho como forma de validar a implementação. Os resultados obtidos para os dois vídeos disponibilizados estão descritos abaixo:

Arquivo de entrada: *"video_05.mp4"*
Resultado h0: *8e423302209494d266a7ab7e1a58ca8502c9bfdaa31dfba70aa8805d20c087bd*

Arquivo de entrada: *"video_03.mp4"*
Resultado h0: *ee24473e4a369a305c9c3d54629eff01f609b8e2f61ca9cf6f3084f13fe346d6*

4. Conclusão

O desenvolvimento deste trabalho me permitiu entender o funcionamento do algoritmo de *hash* seguro *SHA-256,* bem como a forma que pode ser implementado utilizando bibliotecas da linguagem *Java,* as quais auxiliam a codificação. Além disso, foi possível compreender a importância da integridade de mensagens em segurança de sistemas. Conclui-se que a implementação atingiu seu objetivo pois conseguiu encontrar o *hash* esperado, conforme descrito no enunciado do trabalho proposto.

5. Referências

IBM (2019) "Cryptographic Hash Functions". Disponível em: https://www.ibm.com/support/knowledgecenter/en/SSYKE2_8.0.0/com.ibm.java.security.component.80.doc/security-component/jsse2Docs/cryptographichashetc.html.

Oracle (2019) "Message Digest". Disponível em: https://docs.oracle.com/javase/7/docs/api/java/security/MessageDigest.html.