Validação de integridade usando SHA-25

Bernardo C. Felipetto Oliveira[1]

[1]Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brazil

bernardo.oliveira.001@acad.pucrs.br

Abstract. *The main goal of this article is to validate the integrity of a video file by computing its Hash using the SHA- 256 function. The proposed problem and the design of the solution will be presented; to prove the correct implementation of data processing, we used a video file for testing, knowing the correct expected final Hash.*

Resumo. *Este artigo tem por objetivo validar a integridade de um arquivo de vídeo através da computação de seu Hash utilizando a função SHA-256. O problema proposto e o design da solução serão apresentados; para provar a implementação correta do processamento de dados, utilizamos um arquivo de vídeo para teste, conhecendo o seu Hash final correto.*

1. Introdução

As funções *hash* são realizadas através de operações matemáticas computadas sobre um certo dado. Funcionando como uma espécie de assinatura digital, ao comparar o resultado destas operações com o resultado esperado temos a garantia que os dados não foram alterados após seu envio, assim validando a integridade do arquivo. O aspecto mais importante destas operações é a resistência a colisões, isto é, dado duas entradas diferentes é impossível obter o mesmo *hash*. Para realizar o estudo sobre funções *hash* o seguinte cenário foi proposto:

- Realizar o download de um arquivo de vídeo, dividi-lo em blocos, calcular o *hash* de cada bloco de forma encadeada e descobrir o *hash* final referente ao bloco inicial

2. Implementação do Algoritmo

A linguagem escolhida foi *Python* devido ao seu suporte para as funções necessárias. O código inicia recebendo dois argumentos pela linha de comando, o caminho para o arquivo de vídeo e o *hash* esperado, respectivamente. O tamanho de bloco é definido como 1024KB e então cria-se um vetor contendo estes blocos de *bytes*. Caso o arquivo não seja múltiplo de 1024 o python automaticamente lê apenas os *bytes* restantes.

Listing 1. Divisão de blocos de 1024KB

```
VIDEO_FILE_PATH = sys.argv[1]
EXPECTED_H0 = sys.argv[2]

block_size = 1024
blocks = []

with open(VIDEO_FILE_PATH, "rb") as handle:
   block = handle.read(block_size)
   while block:
      blocks.append(block)
      block = handle.read(block_size)
}
```

O hash de cada bloco deve então ser calculado a partir dos seus *bytes* deste vetor, concatenados com os 32 *bytes* referentes ao *hash* anterior. Com o auxílio do *Python* apenas percorremos o vetor criado em sua ordem reversa calculando o SHA-256 do bloco com seu *hash*, sendo que o *hash* da ultima posição é iniciado como vazio. A função *digest* retorna os 32 *bytes* não encriptados que serão utilizados na proxima iteração. Ao final da execução o valor de h0 é convertido para um valor hexadecimal para representação textual, assim o representando em 64 *bytes*.

Listing 2. Cálculo do hash zero

```
h0 = b''
for block in reversed(blocks):
   h0 = hashlib.sha256(block + h0).digest()

hex_h0 = h0.hex()
```

Por fim é realizada apenas uma comparação simples entre o valor hexadecimal de h0 e o valor esperado, que foi informado por parâmetro anteriormente via linha de comando. Uma mensagem é exibida informando se o arquivo sofreu alteração ou não. O *hash* calculado para o video03 foi: **ee24473e4a369a305c9c3d54629eff01f609b8e2f61ca9cf6f3084f13fe346d6**

O vídeo de teste e código fonte completo podem ser acessados neste repositório: https://github.com/BernardoFelipetto/sha256

3. References

Hashlib. Python Documentation. 2019. Disponível em: https://docs.python.org/3/library/hashlib.html
Acesso em: 21 maio 2019.

Funções resumo (Hash functions). Avelino Francisco Zorzo. Material fornecido via Moodle.