Rodrigo Leão da Silveira

Segurança de Sistemas T3 *Hash*

1. Introdução

O terceiro trabalho da disciplina de Segurança de Sistemas consiste na implementação de um sistema capaz de garantir a autenticidade da transmissão de um video de um servidor à um cliente sem que seja necessário enviar o video inteiro para verificação. Supondo que um website armazene um grande arquivo de video em que qualquer um possa fazer o download. Os *browsers* precisam garantir que o video que estão baixando é autentico antes de mostrar o conteúdo ao usuário. Para isto, uma solução possível é a verificação do arquivo através de uma função *hash*, onde o browser teria acesso ao *hash* do arquivo através de um canal seguro e, após o download do arquivo, ele verificaria a autenticidade calculando o *hash* do arquivo e comparando-o com o *hash* obtido. Porém, esta solução obrigaria o browser a baixar todo o video antes de mostra-lo ao usuário, o que tornaria a o processo todo muito lento.

2. Proposta

Como uma solução viável a este problema, o enunciado do trabalho propõe a criação de um algoritmo que seja capaz de dividir o video em blocos de 1024 bytes e calcular o *hash* de cada bloco concatenando o bloco atual com o *hash* do próximo bloco, onde o browser, com o *hash* do primeiro bloco recebido por um canal seguro, seria capaz de verificar o bloco 0 recebido, cada bloco contém 1024 bytes de conteúdo e mais o *hash* do próximo bloco, que será utilizado para verificar a autenticidade do bloco subsequente até todo o arquivo seja recebido. A figura 1 demostra como o algoritmo proposto deverá concatenar os *hashes*.

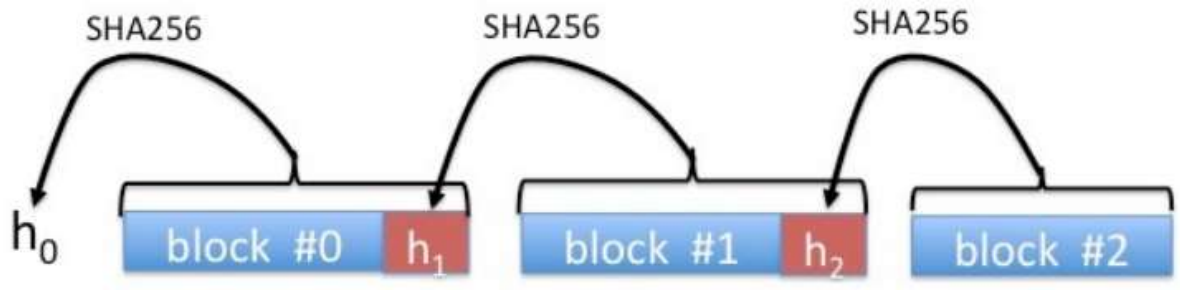

Figura 1. Diagrama da solução proposta.

3. Desenvolvimento

A implementação do algoritmo será dividida em 3 etapas, a divisão do arquivo em blocos, o cálculo do *hash* para os blocos e, como saída do algoritmo, os resultados gravados em um arquivo de texto. Fui utilizado ao ambiente *dotnet core* 3.1 em uma máquina *Windows* para o desenvolvimento.

Na primeira etapa é feito o cálculo de quantos blocos será necessário para dividir o arquivo e a efetiva divisão em blocos, é utilizada uma pilha para armazenar os blocos, visto que, conforme a figura 1 demostra, os *hashes* deverão ser calculados do último bloco para o primeiro. Na segunda etapa os blocos são removidos do topo da pilha e, exceto para o primeiro, é concatenado o *hash* do bloco processado anteriormente com os dados do bloco atual do arquivo e gerado um novo *hash*, que ficará armazena em uma variável temporária para a próxima iteração, cada *hash* gerado é armazena em uma nova pilha de *hashes*, que ordenará os *hashs* dos blocos para a saída do algoritmo. A função *hash* a ser utilizada é a SHA-256. Por fim, a etapa 3 irá converter os bytes dos *hashes* para hexadecimal e gravar eles em um arquivo texto na pasta output.

4. Resultados

Conforme sugerido no enunciado, foi utilizado o arquivo video5.mp4 para validação do algoritmo, onde foi verificado que o *hash* do bloco 0 gerado é o mesmo *hash* esperado. Como resultado para o arquivo video_03.mp4 obtivemos como *hash* do bloco 0 o seguinte *hash*:

```
0: EE24473E4A369A305C9C3D54629EFF01F609B8E2F61CA9CF6F3084F13FE346D6
```

O código fonte e os resultados completos podem ser conferidos no anexo submetido junto a este relatório no *Moodle* ou no seguinte link do *Github*: https://github.com/RLSilveira/SHA256_T3.git