Relatório Técnico
Segurança de Sistemas - Hash

Bruno Bragança Mendes

Resumo. *Este relatório técnico tem por objetivo compreender a utilização das funções resumo - Hash - para garantir a autenticidade de um arquivos calculando o hash para cada bloco do arquivo. O código completo esta disponível em [Mendes 2019]*

1. Introdução

Partindo da ideia que um *website* armazene um arquivo de vídeo, de que forma é possível garantir que o vídeo que está sendo executado em um computador é autêntico? Como pode ser possível confirmar que o mesmo não foi modificado? Realizar o *download* do vídeo, e posteriormente verificar sua autenticidade pode não ser viável, pois determinados vídeos pode ser reproduzidos *online* sem a possibilidade de que baixá-los.

Para confirmar a autenticidade do vídeo, o processo de segurança baseia-se em utilizar funções *hash* criptográficas, SHA-256, para autenticar blocos do vídeo durante sua execução, sem ter que aguardar o *download* completo do mesmo.

2. Criptografia e autenticidade

O processo de criptografia funciona dividindo o vídeo em blocos **b#** de tamanho **n** bytes (nesta atividade, 1024 bytes), e calculando a função *hash* (hb# de cada bloco individualmente concatenada com a função *hash* do bloco anterior, iniciando do último bloco até o primeiro bloco.

Observa-se que o último bloco pode ter um tamanho inferior a **n** bytes. A verificação de autenticidade ocorre quando o browser recebe o primeiro bloco do vídeo, ele calcula o *hash h0* e compara se é o mesmo *hash H(b0||h1)* que o site possui. Se forem iguais, a autenticidade está garantida, e o vídeo é reproduzido.

3. O programa

Para realizar o processo de criptografia foi criado um programa com execução em console na linguagem C# e um arquivo de vídeo no formato mp4 para ser criptografado (video03.mp4). Antes de realizar a leitura do arquivo, obtém-se o tamanho total do vídeo, calcula-se em quantos blocos de tamanho **n** bytes o vídeo será dividido e o tamanho do último bloco com tamanho provavelmente inferior a **n** bytes, obtendo a quantidade **q** de blocos.

Depois inicializa-se um vetor de tamanho de tamanho **q** e carrega-se o vídeo para o vetor **v** utilizando a função *FileStream*. Para guardar o *hash* foi utilizada uma matriz de tamanho **q x 2**, onde a primeira coluna contém o número do bloco e a segunda o *hash* do bloco.

Para calcular o *hash* em C# utilizou-se a classe **SHA256** inclusa por padrão no *namespace System.Security.Cryptography*, e a função *ComputeHash* para completar os 32 bytes caso seja necessário. Para cada bloco do vetor **v** aplica-se a fórmula explicada na sessão anterior para calcular o *hash*, e armazena-se os valores na matriz **m**.

O trecho de código Listing 1 contém o código utilizado para calcular o *hash* de cada bloco.

Listing 1. Cálculo do hash para cada bloco

```
//Inicializa a matriz de hashes e o SHA-256
byte[][] hashes = new byte[numBlocosVideo][];
SHA256 sha256 = SHA256Managed.Create();

//Calcula a hash de cada bloco e
//Armazena a hash de cada bloco em um array de bytes
//Do final para o inicio do arquivo
for (int index = video.Length - 1; index > 0; index--)
{
    // Caso o trecho n o tenha o numero de bytes minimo do SHA-256
    // ele e completado automaticamente
    hashes[index] = sha256.ComputeHash(video[index]);

    //Concatena o hash do bloco atual com o bloco anterior
    byte[] aux = new byte[video[index - 1].Length + hashes[index].Length];
    Array.Copy(video[index - 1], 0, aux, 0, video[index - 1].Length);
    Array.Copy(hashes[index], 0, aux, video[index - 1].Length,
        hashes[index].Length);

    //Substitui bloco anterior pelo bloco concatenado
    video[index - 1] = aux;
}

//Calcula a hash do ultimo bloco (primeiro)
hashes[0] = sha256.ComputeHash(video[0]);
```

4. Conclusão

Após calcular o *hash* de todos os blocos do vídeo, os mesmos são convertidos para hexadecimal e gravados em um arquivo de texto que contém o número do bloco e o *hash* de cada bloco (Figura 1). O primeiro bloco contém o *hash* EE24473E4A369A305C9C3D54629EFF01F609B8E2F61CA9CF6F3084F13FE346D6.

Com este programa foi possível compreender como funciona o processo de criptografia com funções *hash* e uma das formas de confirmação de autenticidade de arquivos carregados para a internet.

Referências

Mendes, B. B. "hash". `https://github.com/brbmendes`. `https://github.com/brbmendes/Hash`. (Acessado em 12 de nov. 2019).

```
0       EE24473E4A369A305C9C3D54629EFF01F609B8E2F61CA9CF6F3084F13FE346D6
1       16C73E691E4467BBD334760600B2B0CF2478B634DA054463B25FBD0C1A5CD22C
2       B5E03690641DF96F5F4D930DFF08092DD59866A85314D4C02AB75CA94296E5B2
3       1E4ADD36839C6B02B8459FB54B6B9CA8FBA408D725E26F3A90C24868F3FF39BE
4       C99B08CC913408C9CCA6B53F1F1559192FCA493EA26C8B5057AB94E73CFC7021
5       319B3A285F129E4C97FB6C186118FCE19DB99509B7B48885F5859972D08D108D
6       F050BFE43E49B0253B6EF63497D1E79C4CDB3B4830577C239A6FF7330050EE57
7       D8885A10677AAB6B01BD6A3C49623F1970C3B4FF61487F0C4F5A517FC038C6FC
8       0530A90D9D69E4478ECC51F9BF84C4CC1BDE74094AEE1C07CDD169F0737E3E3B
9       E34C4DAC18B6AE561B8082041A5384023AC9CE8B1999DFD4AB875AADB41AF68D
10      E8981DC31DEF14B9D36BAE668374E3DEA1E203BD470921590D275F624192648B
11      67D5735DE2716339FC460BE913192A73AB1E49021344EB6358EAC4E6D2D94E1F
12      D6F86179A66B584FA2CECF1AD42036B27CDDA4551C289FD2B327E4C559A5775C
13      6FFCA6EAD8B540A888A6C59B215DA82961876F2CF03D13BCC5FDA313265D271A
14      48585C70BCB2B282D4E661A252609AC5DAF618ECB57D1B14B14217AFE8DA185D
15      F953B4A484B94AC924ED3339507AF8497DE19CBD5D9F943C00B2454166880064
16      4153F6901261B912B8D8526E2358F788DF6EFC9FAF6DE5052EFA47BE135BDDF0
17      75C3D7151E09D6C10FF776DCA02A52B43EB466D74A2FEC173FAFA0C2948A4A37
18      33645FF47AEBDC6376DD352B53D320BDD3DA339076408EE132A4C5FC0B77F99C
19      7313F9A714C0951A35AE2714868A70D2411772D4C52DF97894886FEA086B92D7
20      55082B745CC7AC01ACD210D82354051B946D0CAA3D05BFBF5C0BEFAF7782C3C6
```

Figura 1. Início do arquivo com o *hash* de cada bloco