Relatório Segurança de Sistemas
Criptografia SHA-256

Hugo Neto[1]

[1]Escola Politécnica - Faculdade de Informática (PUCRS)
Porto Alegre – RS – Brazil
Github: https://github.com/hugonn/sha256.git

hugoneto@outlook.com

1. Introdução

O intuito deste relatório é demonstrar,resumidamente, o funcionamento e a implementação de um sistema de criptografia baseado em algoritmos de hash seguros (SHA). O algoritmo utilizado é o SHA256. Este, faz parte da série SHA-2 que é um grupo de hashes criptográficas compostas por funções matemáticas aplicada sobre dados.

O SHA256 funciona da seguinte forma: é calculado o hash a partir de um arquivo e comparado a um hash já conhecido e esperado. Caso os valores sejam iguais, é possível garantir a integridade dos dados que estão naquele arquivo. No caso deste trabalho, o hash disponibilizado para teste foi: ”8e423302209494d266a7ab7e1a58ca8502c9bfdaa31dfba70aa8805d20c087bd”.

A tecnologia escolhida foi a linguagem Python que faz usufruto da biblioteca PyCrypto.

O enunciado que motivou esse relatório encontra-se em: https://moodle.pucrs.br/pluginfile.php/2675568/mod_resource/content/1/SHA-exercise.pdf

2. Implementação

Seguindo o que fora pedido no enunciado do trabalho, a primeira coisa feita foi ajustar o tamanho padrão do bloco, 1KB, e carregar o tamanho dos videos que seriam feitos os cálculos do hash. Era necessário que o arquivo fosse percorrido de trás para frente já que o cálculo de hash deve ser feito dessa forma.

Em seguida, era necessário descobrir o tamanho do bloco variável. Esse, foi descoberto a partir do cálculo do resto do tamanho total do vídeo pelo tamanho do bloco.

```
tam_bloco = 1024 #blocos de 1KB

#Video teste do hash
tam_video1 = os.path.getsize("/home/hugo/Área de Trabalho/SHA256-Video/video05.mp4")

#video que precisa calcular o hash
tam_video2 = os.path.getsize("/home/hugo/Área de Trabalho/SHA256-Video/video_03.mp4")

# Descobre o tamanho variável do último bloco dos dois videos.
tUlt_bloco_vid1 = tam_video1 % tam_bloco
tUlt_bloco_vid2 = tam_video2 % tam_bloco

#inicialização das variaveis que vao conter o hash h0
bloco_hash_vid1 = ''
bloco_hash_vid2 = ''

#variavel que vai conter o tamanho do bloco
bloco_controle = 0

#abertura do arquivo
arq = open("/home/hugo/Área de Trabalho/SHA256-Video/video05.mp4",'rb')
```

Figura 1. Implementação - parte 1

Uma flag, ”flag_termino”, foi setada para controle. Caso ela fosse igual a zero, significava que o arquivo de vídeo inteiro tinha sido percorrido. Feito isso, era necessário ler o valor que estava nos blocos do arquivo. Assim, era cálculado o hash e adicionado ao bloco subsequente para que o mesmo fosse calculado até chegar em h0.

Figura 2. Implementação - parte 2

O objeto SHA256 foi criado (seguindo as orientações da biblioteca PyCrypto) e, em um loop, para cada bloco lido do arquivo, era calculado o hash do bloco em questão e em seguida era feito o cálculo dos hashes subsequentes. Uma variável guardava sempre o último hash. Assim, quando chegasse ao fim da execução, o h0 poderia ser descoberto.

```
    #Inicializa o objeto relacionado a biblioteca do pycrypto que tem os métodos necessários para calculo do
    obj_sha = SHA256.new()

    #Faz o update dos hashes das mensagens continuamente pedaço por pedaço dos dados
    obj_sha.update(bloco_data)

    #FAz o update do hash conforme percorre o arquivo
    if(bloco_hash_vid1):
        obj_sha.update(bloco_hash_vid1)

    bloco_hash_vid1 = obj_sha.digest()

    #Percorre o arquivo diminuindo o tamanho da flag_termino. QUando chegar em zero, significa que leu tudo
    flag_termino -= bloco_controle

# Printa H0
print("H0 ---->",bytes.decode(hexlify(bloco_hash_vid1)))
```

Figura 3. Implementação - parte 3

O primeiro teste foi feito com o vídeo que o hash era conhecido. Isso ocorreu pois era necessário saber se o cálculo do SHA 256 estava sendo feito corretamente.

Após executar o algoritmo, tivemos o resultado esperado:

```
H0 ----> 8e423302209494d266a7ab7e1a58ca8502c9bfdaa31dfba70aa8805d20c087bd
```

Figura 4. Resultado - Video 05

Agora, executamos o algoritmo com o vídeo que queremos descobrir o hash h0. Foi necessário apenas trocar algumas variáveis. O resultado obtido foi:

```
H0 ----> ee24473e4a369a305c9c3d54629eff01f609b8e2f61ca9cf6f3084f13fe346d6
```

Figura 5. Resultado - Video 03

Desta forma, conseguimos calcular o hash h0 corretamente.

3. Conclusão

Assim, encerra-se o ciclo de trabalhos da cadeira de Segurança de Sistemas. A implementação deste trabalho foi mais complicada que as outras devido a lógica necessária para o desenvolvimento desse software. Mas, com a facilidade na utilização da biblioteca Pycrypto, as dificuldades foram amenizadas. Sendo assim, a implementação acabou sendo facilitada por conta de tal biblioteca.

Referências

Wikipedia (2018). Sha-2. Último Acesso: 3 de Dezembro de 2018.