INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA RIO GRANDE DO NORTE

Professor: Macêdo Firmino
Disciplina: Segurança de Rede
Prática 17: Função Hash e HMAC com SHA.

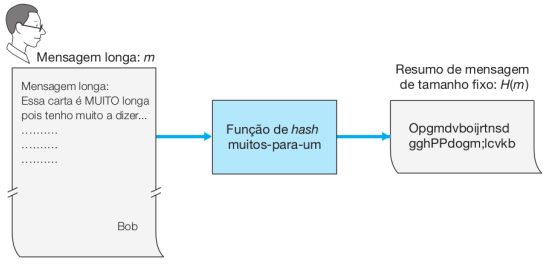
Olá, meus alunos!! Como é que vocês vão? Na aula de hoje iremos compreender o funcionamento das funções hash e a função HMAC, suas aplicações em segurança da informação, e experimentar na prática a geração e comparação de hashes e HMAC utilizando ferramentas do Linux. Vamos lá!!! Preparados???

Configurando o Ambiente

Para estudarmos estes conceitos e ferramenta iremos utilizar duas máquinas virtuais. Uma para transmitir os segredos (Iria) e outra para receber os segredos (Macedo). Podemos utilizar qualquer distribuição Linux. Nos nossos testes utilizei Kali Linux (máquina de Iria) e uma máquina Ubuntu (máquina Macedo), ambas configuradas em Rede Nat.

Função Hash

A função Hash (Resumo) é uma classe de algoritmo que transforma qualquer tipo de dado (como um texto, uma imagem ou um arquivo inteiro) em uma sequência curta de caracteres de tamanho fixo. Essa sequência é chamada de hash, resumo, ou impressão digital dos dados. Por exemplo, o algoritmo SHA-256 sempre gera uma saída de 256 bits, independentemente do tamanho do conteúdo original.



Estas funções são unidirecionais, ou seja, não é possível recuperar o dado original a partir do resumo gerado. Dessa forma, as funções Hash são largamente utilizadas para:

- Verificar a integridade de dados (checar se um arquivo ou mensagem foi alterado);
- Proteger senhas (armazenando apenas o hash, e não a senha em si);
- Assinar digitalmente documentos (a assinatura é feita sobre o hash do conteúdo);
- Construir blocos em um blockchain, garantindo que os dados não sejam modificados;
- Organizar dados em tabelas de busca, como em bancos de dados e sistemas de cache.

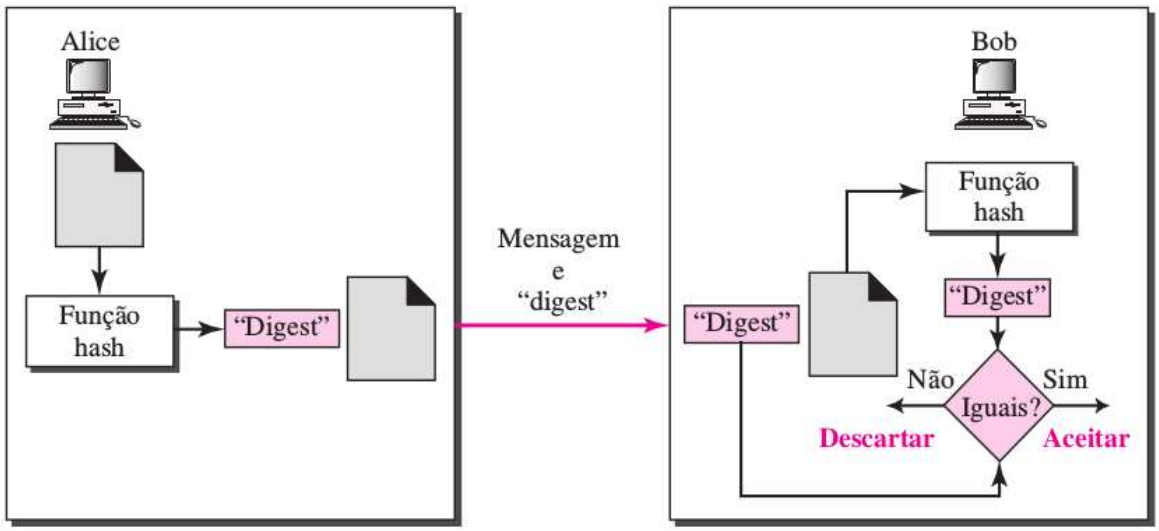
O *hash* da mensagem é criado no emissor e enviado juntamente com a mensagem para o receptor. Para verificar a integridade da mensagem, o receptor calcula a função *hash* novamente e compara o resultado com aquele recebido. Se ambos forem o mesmo, o receptor tem certeza de que a mensagem original não foi alterada.

Algumas propriedades importantes das funções Hash são:

- Determinística: a mesma entrada sempre gera o mesmo hash.
- Unidirecional: não é possível descobrir a entrada original a partir do hash.
- Rápida: pode ser aplicada rapidamente, mesmo com grandes volumes de dados.
- Sensível a alterações: uma pequena mudança na entrada altera todo o hash.
- Resistente a colisões: deve ser muito difícil encontrar duas entradas diferentes com o mesmo hash (embora isso possa ocorrer em algoritmos antigos como o MD5).

Os principal algoritmo de SHA é o *Secure Hash Function* (SHA). Ele foi projetada pela Agência de Segurança Nacional dos Estados Unidos e é a mais utilizado. As versão mais comuns são: SHA, SHA-1 e SHA-2. O SHA-1 é um hash de 160 bits. SHA-2 é na verdade uma "família" de hashes (SHA224, SHA256, SHA384 e SHA512) e vem em uma variedade de comprimentos, sendo o mais popular de 256 bits.

Funcionamento do Hash:



SHASum

O SHASum é uma família de programa que calcula e verifica as funções hashes SHA-1 e SHA-2. Ele é instalado por padrão na maioria das distribuições Linux. Os comandos são: shasum, sha224sum, sha256sum, sha384sum e sha512sum.

O comando shaSum calcula o resumo da mensagem de um arquivo. Isso permite que ele seja comparado ao resumo original da mensagem para verificarmos se o arquivo não foi modificado.

Iremos criar um arquivo de texto, calcular o seu hash, realizar alterações no arquivo e calcular novamente. Depois iremos enviar o arquivo e o hash para um amigo da turma, por exemplo por e-mail. O seu amigo deverá receber o arquivo, calcular o hash e verificar se os dados não foram alterados na transmissão.

Inicialmente iremos criar o arquivo. Você pode utilizar o editor de sua preferência, abaixo será mostrado um exemplo.

```
echo "Olá, mundo!" > arquivo1.txt
echo "Olá, Mundo!" > arquivo2.txt
```

O sha1sum produz um valor de hash de 160 bits (20 bytes) do arquivo, através do comando:

```
sha1sum arquivo1.txt

b92378171ddf31e6a1dfe5b099e15bda97eb01ab

sha1sum arquivo2.txt

12b5c2abcadf59bff45e768fd2c182825c4c3ea9
```

Como é possível observar nos resultados, pequenas mudanças no conteúdo geram grandes mudanças no hash. Além disso, não é possível a partir do hash obter o texto original.

Agora iremos renomear o arquivo e calcularmos novamente a função Hash.

```
mv arquivo1.txt arquivo3.txt

sha1sum arquivo3.txt

b92378171ddf31e6a1dfe5b099e15bda97eb01ab
```

Como resultado obtivemos o mesmo hash. Observe que alterando os atributos do arquivo, por exemplo, nome, permissão e localização o resultado do hash não se altera.

HMAC

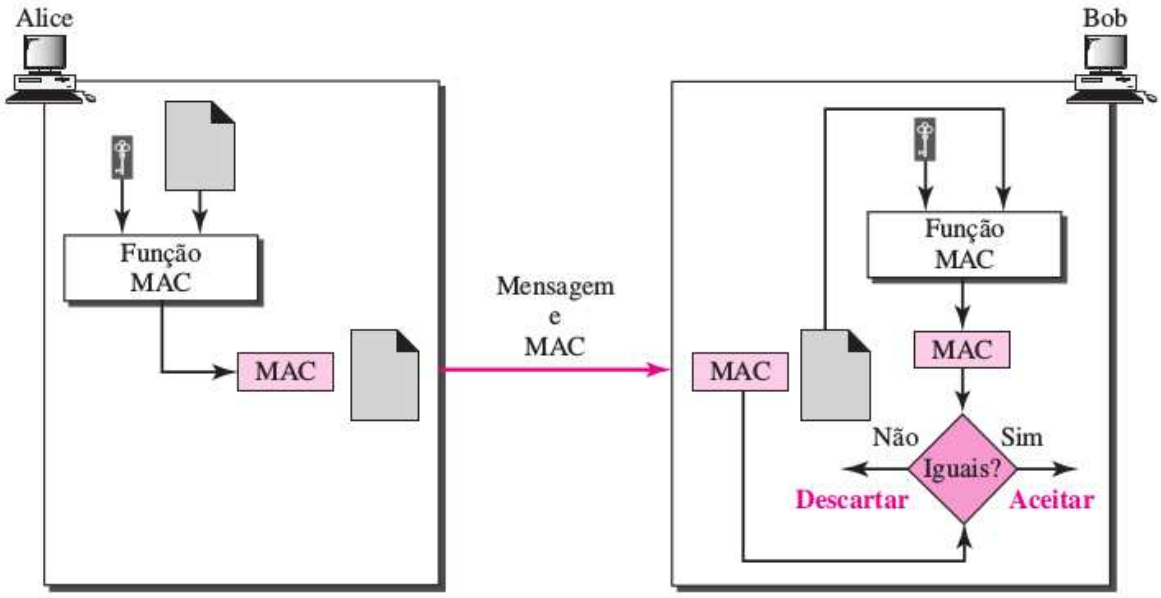
O HMAC (*Hash-based Message Authentication Code*) é um mecanismo criptográfico usado para verificar a integridade e a autenticidade de uma mensagem. Ele combina uma função hash com uma chave secreta compartilhada entre as partes envolvidas na comunicação.

Uma função hash garante apenas a integridade de uma mensagem. Para garantir a integridade e autenticidade, ou seja, não ocorreu alteração dos dados e que quem enviou foi realmente a pessoa que se diz ser é utilizado a função hash com chaves simétricas com o HMAC. O HMAC garante que somente quem conhece a chave secreta pode gerar ou verificar o valor autenticador, tornando-o útil para autenticação segura de mensagens.

O HMAC detectar se o conteúdo de uma mensagem foi modificado durante o trajeto. Desta forma, evita ataques do tipo *man-in-the-middle* (alguém alterando os dados no meio do caminho). O funcionamento do HMAC segue os seguintes passos:

1. A mensagem original é combinada com uma chave secreta.
2. Essa combinação é processada com uma função hash;
3. A saída final é o HMAC, que serve como uma "assinatura" curta e única daquela mensagem com aquela chave.

Funcionamento do HMAC:



Por exemplo, imagine que um servidor e um cliente compartilham uma chave secreta chamada "chave123". Sempre que o cliente envia uma mensagem ao servidor, ele também envia o HMAC da mensagem, gerado com essa chave.

O servidor, ao receber a mensagem, gera o HMAC por conta própria usando a mesma chave. Se o valor calculado bater com o valor recebido, a mensagem é considerada autêntica, íntegra e quem criou foi quem possuia a chave secreta. Caso contrário, ela pode ter sido alterada ou forjada.

```
echo "mensagem secreta" > msg.txt


openssl dgst -sha1 -hmac
 "chave123" msg.txt


39dc26c60a609e0a3f47f936c45eb1e5a84fdbab
```

Agora iremos mudar o conteúdo do arquivo e verificar novamente, por exemplo:

```
echo "mensagem alterada" > msg.txt

openssl dgst -sha1 -hmac
 "chave123" msg.txt

7957fe88a911f96020b245c3f0698b9820bfcb3e
```

Agora iremos mudar a chave secreta e verificar novamente, por exemplo:

```
openssl dgst -sha1 -hmac
 "chave456" msg.txt

210d54d97c26cc96deaedbcc0fa0fc39d2eb4615
```

Observamos um HMAC completamente diferente. Isso mostra que qualquer alteração na mensagem ou na senha muda o valor do HMAC.

Atividades

01. No google sala de aula baixe o arquivo "alunos_segurança.pdf". Calcule o hash SHA1Sum desse arquivo e compare o valor obtido pelo professor e apresentado no arquivo hash_sha1sum.txt.

a) Se os valores derem iguais, o que isto significa? E se derem diferentes o que significa? Coloquem a resposta no Google Sala de Aula.

02. Crie um arquivo de texto, chamado mensagem.txt com o seu nome e sua matrícula. Gere o HMAC SHA1 desse arquivo usando uma chave secreta "Senha123". Envie para o google sala de aula, o arquivo de texto e outro arquivo de texto com o valor do HMAC obtido.

03. O que você entendeu de diferente entre Hash e HMAC? Qual a finalidade de cada um?

04. Pesquise como o Hash é utilizado nas criptomoedas.