ALGORITMOS DE *HASH*

Devido à grande quantidade de informações que circulam nas redes, é muito importante garantir que os usuários tenham seus dados a salvo de agentes não autorizados. A transmissão segura e secreta de mensagens é uma preocupação constante e deu surgimento à criptografia, que, em essência, é a prática de tornar ilegível um texto previamente legível, e depois inverter tal operação, protegendo uma mensagem de entidades indesejadas. Atualmente, a criptografia é fundamentada em quatro características da segurança da informação (ISO/IEC 17799:2005):

- **Confidencialidade**: limitar o acesso à informação somente às entidades legítimas, ou seja, aos autorizados a vê-la.
- **Integridade**: garantir que a informação não foi alterada por meios desconhecidos ou não autorizados, ou seja, ela deve ter todas as características originais estabelecidas por seu proprietário.
- **Autenticidade**: confirmação da identidade da entidade que realizou a assinatura da mensagem.
- **Disponibilidade**: garantir que a informação esteja sempre disponível para aqueles usuários autorizados por seu proprietário.

Nesse sentido, os algoritmos de *hash* são importantes, pois são ferramentas que asseguram a integridade das informações transmitidas.

Uma função *hash* (conhecida também como função resumo) é qualquer algoritmo que possibilita a transformação de uma grande quantidade de dados com tamanho variável em pequenos dados de tamanho fixo. Por esse motivo, as funções *hash* são conhecidas por resumirem os dados. Os valores retornados por uma função

hash são chamados de valores *hash*, códigos *hash*, somas *hash* ou, simplesmente, *hashes*.

Uma das principais aplicações dessas funções é a comparação de dados grandes ou secretos. Outras aplicações das funções *hash* incluem a busca de elementos em bases de dados, a verificação da integridade de arquivos baixados ou o armazenamento e transmissão de senhas de usuários.

No caso das buscas em bancos de dados, ela se dá acessando diretamente o elemento desejado, a partir da criação de índices (como os sumários de livros). Esses índices são obtidos após a utilização de uma função *hash* nos dados. Portanto, não é a partir do dado que um elemento é encontrado, mas pelo seu resumo.

Para verificar a integridade de arquivos baixados ou transferidos, são utilizadas funções *hash* sobre o dado, armazenando-se o resumo gerado. Depois que o arquivo é transferido para o receptor, calcula-se o resumo dos dados recebidos. Se os resumos forem iguais, o arquivo é igual ao enviado.

Já no caso de armazenamento e transmissão de senhas de usuários, apenas são armazenados no servidor os resumos destas. Quando um usuário entra no sistema com suas credenciais, uma função *hash* (a mesma utilizada no armazenamento) calcula o resumo da senha inserida por ele. O servidor compara o resumo gerado com o armazenado e, caso sejam iguais, o usuário será autenticado.

Existem muitos tipos de funções *hash* e a complexidade do algoritmo depende das características que se pretendem garantir com a função. A propriedade fundamental de toda função *hash* é ser unidirecional, o que matematicamente significa que não existe uma função inversa, ou seja, não é possível recuperar a informação original a partir do *hash* gerado. Um exemplo clássico de função *hash* é a função resto da divisão. Por exemplo, os números da sequência 1, 11, 21, 31, 41, 51, 61… terão valor *hash* igual a 1 usando o resto por 10. Nesse caso, é impossível saber o número original, dado que se tem o resumo 1.

Uma característica desejável é a resistência à colisão. Uma colisão acontece quando dois dados originais geram o mesmo valor *hash*. No exemplo anterior, todos os números geraram o mesmo valor *hash* igual a 1. Portanto, o objetivo principal dos

projetistas de função *hash* é reduzir ao máximo a probabilidade de ocorrência das colisões. Para isso, é comum ajustar a distribuição dos *hashes*: quanto mais uniforme e dispersa é a função *hash*, menor é sua probabilidade de colisão. A Figura 1 mostra um exemplo de colisão, em que os nomes "João da Silva" e "Pedro Henrique" geram o mesmo resumo 02 após o uso de uma função *hash*.

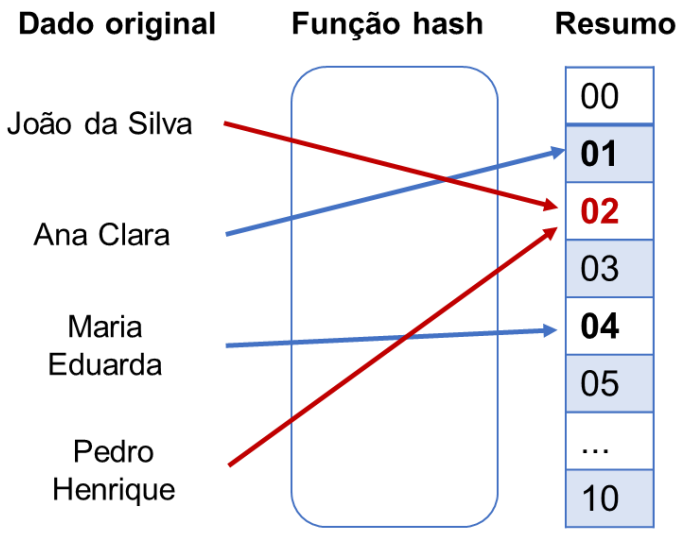

Figura 1 – Exemplo de colisão. Fonte: Elaborada pelo autor (2021).

Uma característica necessária para as funções *hash* é a recorrência. Isso significa que, se uma função *hash* for aplicada sobre os mesmos dados originais, sempre serão gerados os mesmos resumos para os dados. Outra propriedade importante é a resistência à pré-imagem, que envolve a tentativa de encontrar um determinado valor de *hash* que seja igual ao do dado original.

Para garantir a segurança das funções *hash*, outras características são necessárias, como, por exemplo, uma grande quantidade de resumos possíveis, pois uma forma de ataque é a busca por um outro dado que gere o mesmo resumo. Atualmente, as funções *hash* modernas são capazes de gerar resumos de 160 a 512 *bits*, o que significa que existem até 1.048 possibilidades de resumo.

É possível aumentar a segurança das funções *hash* utilizando uma chave no momento da geração do resumo, e somente aquele que a conheça será capaz de gerar

o resumo a partir do dado original. Essa técnica é utilizada na verificação de dados transmitidos. Como o valor *hash* do dado original também é transmitido, um atacante poderia interceptar o dado e o valor *hash*, trocar o dado e inserir o novo valor nele. Tal prática faria com que o receptor achasse erradamente que a mensagem não foi alterada. Se o receptor e o emissor combinarem uma senha para a geração do resumo, o atacante não consegue gerar um valor *hash* válido.

Na criptografia, uma função *hash* criptográfico mapeia uma palavra de tamanho qualquer para uma palavra com um tamanho fixo $t \in N$. Para ser considerada segura, uma função *hash* criptográfico $X \rightarrow \{0, 1\}t$ deve respeitar as seguintes propriedades:

- **Resistência à pré-imagem**: dado um *hash* $h \in \{0, 1\}^t$, deve ser computacionalmente inviável encontrar uma mensagem $m \in X$, tal que $h = hash(m)$.
- **Resistência à segunda pré-imagem**: dada uma mensagem $m_1 \in X$, deve ser computacionalmente inviável encontrar outra mensagem $m_2 \in X$ diferente de m_1, tal que $hash(m_1) = hash(m_2)$.
- **Resistência à colisão**: é computacionalmente inviável encontrar duas palavras $m_1, m_2 \in X$ com $m_1 \neq m_2$, tal que $hash(m_1) = hash(m_2)$.

Atualmente, as funções *hash* mais utilizadas pertencem à família *secure hash algorithm* (SHA), publicada pelo National Institute of Standards and Technology (NIST). O primeiro algoritmo dessa família, SHA-1, já não é mais considerado seguro, devido ao avanço da criptoanálise e do poder computacional. O SHA-2 é uma família de duas funções *hash* similares, com diferentes tamanhos de bloco, conhecidas como SHA-256 e SHA-512. A função *hash* MD5 (*message digest* – resumo da mensagem) é ainda amplamente usada em aplicações legadas. Esse algoritmo produz um valor *hash* de 128 *bits*, para uma mensagem de entrada de tamanho arbitrário. Entretanto, é considerado inseguro, pois foi demonstrado, por meio de vários ataques, que tem vulnerabilidades.

Existem muitas ferramentas *on-line* que permitem gerar códigos *hash* para os algoritmos citados e outros. Podemos acessar, por exemplo, a seguinte página no GitHub: https://emn178.github.io/online-tools/sha256.html.

Se inserimos a palavra “Brasil”, o resultado será conforme apresentado na tabela:

Algoritmo	**Valor *hash***
SHA512	07d8b53b89c4f687e82bac6b944697500d2db6c356a01c85 ad53684fbf79d53536393aa554af68336df224eaf8447da0f0 d5f4fee62f08497fbce49ce5b22f1f
SHA256	6641c5a7ff9f56ef2baceefb41d9906583b9816a7b3e9c4c23 773646be584caf
MD5	aa43becf0d21463be7540bf3b40bf243

Tabela 1 – Valores *hash* para a palavra "Brasil". Fonte: Elaborada pelo autor (2021).

REFERÊNCIAS BIBLIOGRÁFICAS

BARRETO, J.; ZANIN, A.; MORAIS, I.; VETTORAZZO, A. **Fundamentos de segurança da informação**. Porto Alegre: Sagah, 2018.

CABRAL, C.; CAPRINO, W. **Trilhas em segurança da informação**: caminhos e ideias para a proteção de dados. Rio de Janeiro: Brasport, 2015.

FIGUEIREDO, L. M. Introdução à criptografia. **Fundação CECIERJ**, 2010. Disponível em: https://canal.cecierj.edu.br/recurso/6505. Acesso em: 29 maio 2021.

STALLINGS, W.; BROWN, L. **Segurança de computadores**: princípios e práticas. 2. ed. Rio de Janeiro: Elsevier, 2014.