Criptografia

Funções hash

Roadmap

- Funções one-way
- Principais características
- Algoritmos de hash comuns
- Funções hash criptográficas
- Segurança das funções hash

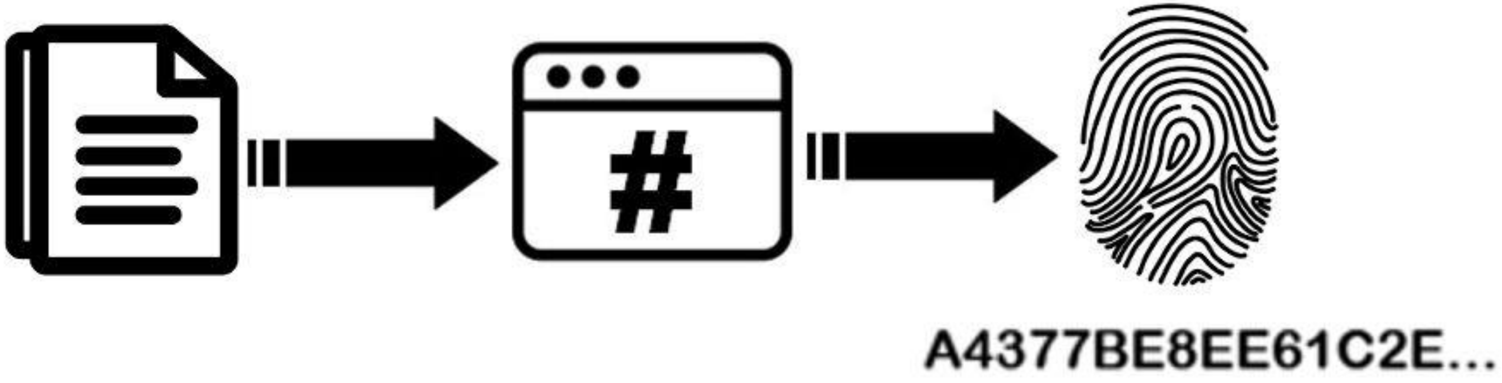
Universidade Federal do Ceará
Campus de Crateús

Ciência da Computação
Sistemas de Informação

emerson@crateus.ufc.br

Funções hash

Definição

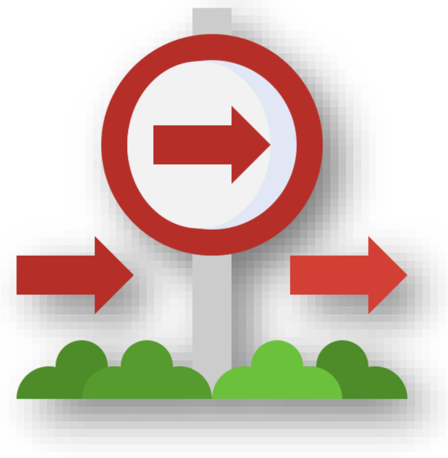


- As **funções hash** estão entre as primitivas criptográficas mais importantes. São usadas em muitos protocolos criptográficos, e são fundamentais para o funcionamento da blockchain.

- As funções de hash transformam qualquer tipo de dado de entrada, independentemente de seu tamanho, em uma cadeia de bits de comprimento fixo.

Definição

- As funções pertencem a um grupo conhecido como funções *one-way* (unidirecional).



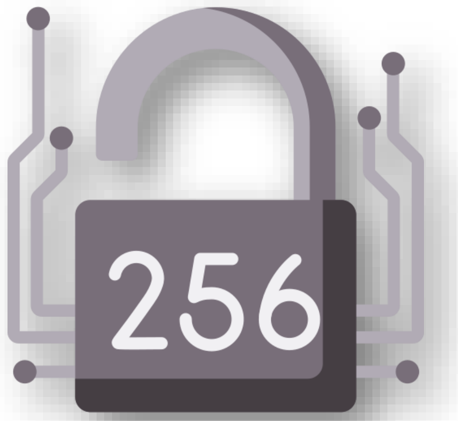
Funções unidirecionais

- Sejam $\boldsymbol{X}$ e $\boldsymbol{Y}$ conjuntos arbitrários. Uma função $f : \boldsymbol{X} \longrightarrow \boldsymbol{Y}$ é chamada de função *one-way* (unidirecional), se $\boldsymbol{f}(\boldsymbol{x})$ puder ser calculado eficientemente para cada $\boldsymbol{x} \in \boldsymbol{X}$, e for computacionalmente inviável calcular $\boldsymbol{f}^{-1}$ para todo $\boldsymbol{f} \in \boldsymbol{Y}$.

Funções hash

Definição

- As funções de hash estão diretamente relacionadas à verificação da integridade das mensagens.



Definição

- Uma função hash é um mapeamento unidirecional $\mathcal{H}: \{\mathbf{0}, \mathbf{1}\}^* \rightarrow \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{i}}$, para algum $\boldsymbol{i} \in \mathbb{N}$.
    - Neste mapeamento, $\mathcal{D} = \{\mathbf{0}, \mathbf{1}\}^*$ é o domínio da função, possivelmente infinito, que representa as possíveis mensagens de tamanho arbitrário.
    - $\mathcal{I} = \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{i}}$ é a imagem da função que representa os possíveis valores hash de tamanho $\boldsymbol{i}$.

Funções hash

Principais caracteríricas

- O **objetivo** central de uma função hash é gerar um valor (uma *tag*) que forneça uma identificação unívoca para cada mensagem.

- A forma como uma função hash é **construída** garante que, se houver qualquer alteração na mensagem original, ainda que modificado um único bit, um outro **valor hash** será gerado.
    - Dessa forma é possível identificar se uma mensagem foi **violada**.

- ***As funções de hash não tem o objetivo de garantir confidencialidade de uma mensagem e sim a integridade.***

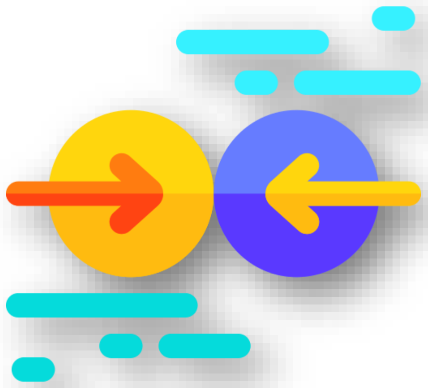
Funções hash

Principais caracteríricas

- Em geral, o tamanho da mensagem $\boldsymbol{x} \in \boldsymbol{X}$ é muito maior que o tamanho do valor hash $\boldsymbol{y} \in \boldsymbol{Y}$, isto significa que uma função hash pode reduzir a mensagem original $\boldsymbol{x}$ em uma *string* muito menor de **tamanho fixo**, independentemente do tamanho da entrada.
  - Por causa dessa redução de tamanho, as funções hash também são conhecidas como ***função resumo***.

- Assim, a quantidade de valores hash possíveis para cada função é $\boldsymbol{2^{i}}$, em que $\boldsymbol{i}$ é o tamanho da *tag* de saída.

Colisão



- Como o conjunto $\mathcal{D}$ é muito maior que o conjunto $\boldsymbol{I}$, duas mensagens $\boldsymbol{m_1}$ e $\boldsymbol{m_2}$ podem ser mapeadas para o mesmo valor hash $\boldsymbol{h}$ — isso é conhecido como uma **colisão**.

Definição

Uma colisão da função hash $\mathcal{H}$ ocorre quando duas mensagens $\boldsymbol{m_1} \neq \boldsymbol{m_2}$, produzem $\mathcal{H}(\boldsymbol{m_1}) = \mathcal{H}(\boldsymbol{m_2})$.

Funções hash

Cenário de uso

Alice quer proteger um arquivo m, armazenado em seu computador, de mudanças indevidas.

Alice pode gerar um valor hash de m com a função $\mathcal{H}(m) = h$, após sua última alteração.

Ao desconfiar que m sofreu alteração e tornou-se m', Alice faz uso da função hash $\mathcal{H}$, tal como $\mathcal{H}(m') = h'$

Se $h = h'$, então o arquivo **não** foi violado. Se $h \neq h'$ então Alice constata que o arquivo foi alterado indevidamente. t

Algoritmos de hash populares

MD5

- Etiqueta de saída = **128 bits** (quantidade de valores possíveis = 2^{128})

SHA1

- Etiqueta de saída = **160 bits** (quantidade de valores possíveis = 2^{160})

SHA2 (família de funções hash)

- SHA-224, SHA-256, SHA-384 e SHA-512

Funções hash

Propriedades básicas das funções hash

Propriedades básicas		
1. Tamanho arbitrário da mensagem	→	$\mathcal{H}(\boldsymbol{m})$ pode ser calculado para mensagens $\boldsymbol{m}$ de qualquer tamanho.
2. Comprimento de saída fixo (compressão)	→	$\mathcal{H}(\boldsymbol{m})$ transforma mensagens $\boldsymbol{m}$ de qualquer tamanho em um valor hash $\boldsymbol{h}$ de comprimento fixo
3. Eficiência	→	$\mathcal{H}(\boldsymbol{m})$ é rápido e fácil de calcular — em tempo polinomial
4. Determinística	→	$\mathcal{H}(\boldsymbol{m})$ produz valores de hash idênticos para dados de entrada idênticos
5. Pseudoaleatória	→	$\mathcal{H}(\boldsymbol{m})$ muda imprevisivelmente quando $\boldsymbol{m}$ muda.
6. Unidirecional	→	A inversão de $\mathcal{H}(\boldsymbol{m})$ é computacionalmente inviável — por exemplo, leva tempo exponencial.

Funções hash criptográficas

Para serem usadas em sistemas criptográficos, as funções hash devem satisfazer **três propriedades essenciais**.

Se tais propriedades forem satisfeitas, elas são chamadas de **funções de hash criptográficas.**

- A **segurança** das funções hash está diretamente relacionada à **dificuldade de gerar colisões.**
- Quanto **maior a dificuldade de encontrar colisões**, maior o nível de segurança da função.

Funções hash

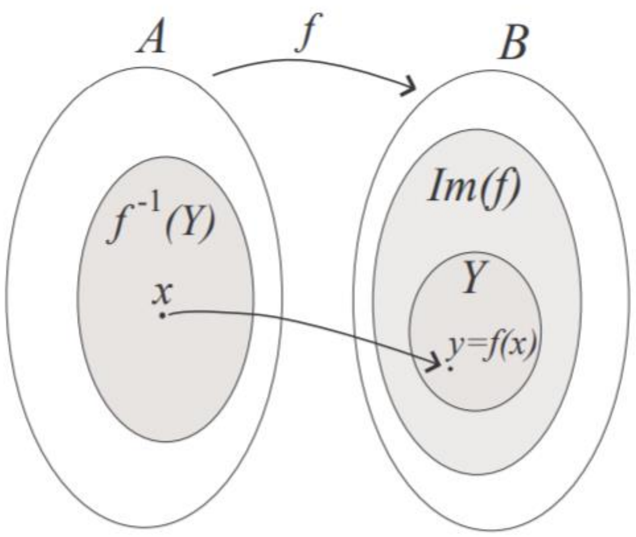
Funções hash criptográficas

Propriedades Essenciais

Resistência à pré-imagem	Resistência à segunda pré-imagem	Resistência à colisão
Dada uma mensagem m, é fácil calcular um valor hash h. Mas dado um valor hash h é computacionalmente difícil encontrar encontrar m;	Dado uma mensagem m, é computacionalmente difícil encontrar mensagem $m' \neq m$, tal que produza o mesmo hash $\mathcal{H}(m') = \mathcal{H}(m)$.	É computacionalmente difícil encontrar um par de mensagens qualquer (x, y) que produza o mesmo hash, tal que $\mathcal{H}(x) = \mathcal{H}(y)$.

Funções hash criptográficas

Resistente à pré-imagem (propriedade unidirecional)



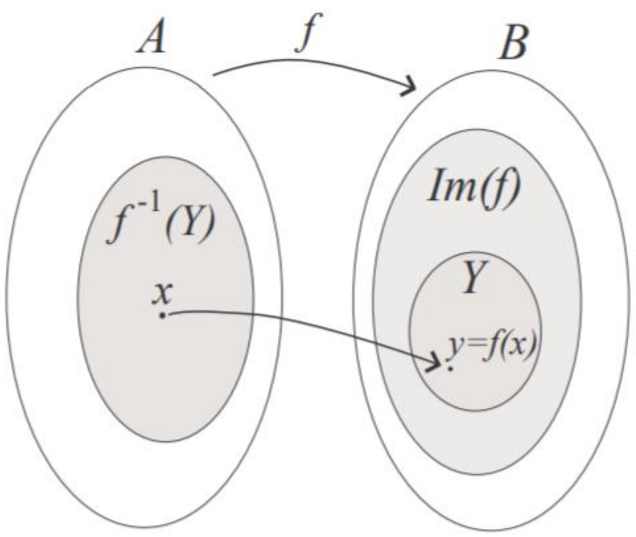
- Sejam $f : A \rightarrow B$ uma função e $Y \subseteq B$. Chama-se **pré-imagem** (ou imagem inversa) de Y sob f o conjunto $f^{-1}(Y)$ formado por todos os $x \in A$ tais que $f(x) \in Y$. Simbolicamente,

$$f^{-1}(Y) = \{x \in A : f(x) \in Y\}$$

- Em outras palavras, a pré-imagem de uma função hash é um subconjunto de todos os valores que produzem um hash específico quando passado como entrada para uma função hash.

Funções hash criptográficas

Resistente à segunda pré-imagem



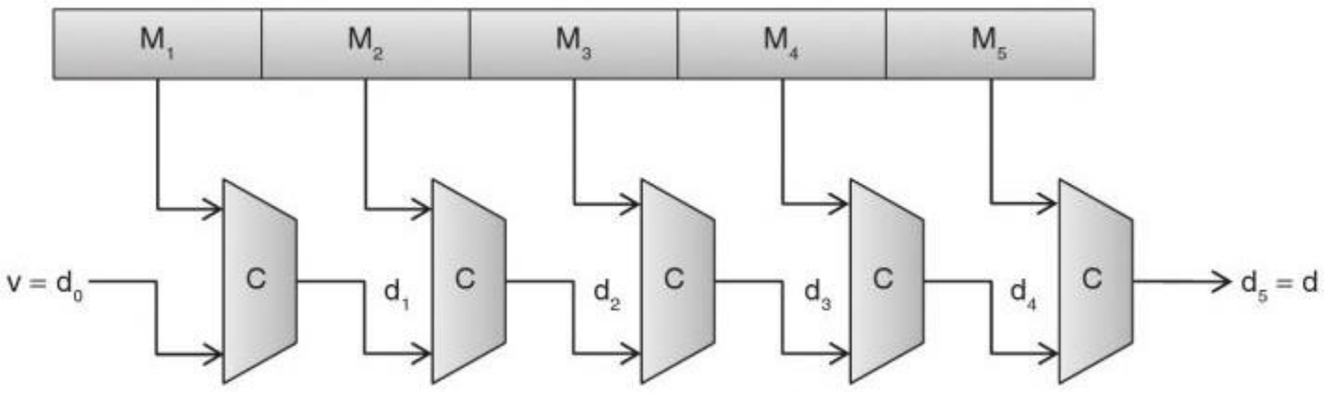
- A propriedade de resistência à **segunda pré-imagem** também envolve a pré-imagem de uma função de hash.
- Essa propriedade indica que deve ser **impraticável** encontrar uma segunda entrada da pré-imagem, $\boldsymbol{x'} \in \boldsymbol{f^{-1}(Y)} \neq \boldsymbol{x}$, que também produzirá o mesmo hash de mensagem conhecido, tal que $\mathcal{H}(\boldsymbol{x'}) = \mathcal{H}(\boldsymbol{x})$.
- Ou seja, se você conhece a entrada $\boldsymbol{x}$ que produziu um hash específico $\boldsymbol{h}$, **não** poderá obter uma segunda entrada $\boldsymbol{x'}$ que também gerará o mesmo hash $\boldsymbol{h}$. Isto é, deve $\mathcal{H}(\boldsymbol{x'}) \neq \mathcal{H}(\boldsymbol{x})$.

Funções hash criptográficas

Resistente à colisão

- Uma função hash $\mathcal{H}$ resistente a colisões significa que $\mathcal{H}$ foi implementada usando uma **função de compressão**, de tal forma que é impraticável encontrar colisões de hash.

Função de compressão. É uma função que a partir de uma cadeia de bits de tamanho $\boldsymbol{m}$ gera uma cadeia de bits de tamanho $\boldsymbol{n}$, em que $\boldsymbol{m} > \boldsymbol{n}$. É um mapeamento $\boldsymbol{C}$: $\{\mathbf{0},\mathbf{1}\}^{\boldsymbol{m}} \rightarrow \{\mathbf{0},\mathbf{1}\}^{\boldsymbol{n}}$.



- **M**$_i$ representa cada bloco da mensagem **M**;
- **C** é a função de compressão;
- **d**$_i$ representa o valor hash gerada em cada iteração;
- **v** é o valor de inicialização.

MD5 – um algoritmo inseguro

- No MD5 já é possível encontra colisões com. O exemplo a seguir mostra duas entradas distintas que produzem o mesmo hash MD5:

string1 =“d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e488832571415a085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70”

string2 =“d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080280d1ec69821bcb6a8839396f965ab6ff72a70”

string1 MD5 hash = “79054025255fb1a26e4bc422aef54eb4”
string2 MD5 hash = “79054025255fb1a26e4bc422aef54eb4”

Funções hash

Propriedade avalanche

- A propriedade avalanche de uma função hash garante que qualquer alteração minúscula na entrada original gerará um hash significativamente alterado (aproximadamente 50% ou mais da saída resultante deve ser diferente).

EXAMPLE

O hash MD5 da palavra **password** é **5f4dcc3b5aa765d61d8327deb882cf99**
O hash MD5 da palavra **p@ssword** é **90f2c9c53f66540e67349e0ab83d8cd0**.

- A comparação dos dois hashes MD5 demonstra que uma única alteração de caractere na entrada original resultou em um hash drasticamente diferente.

Segurança das funções hash

- A segurança das funções hash está diretamente relacionada com a dificuldade de gerar colisões, quanto maior a dificuldade de encontrar colisão, maior o nível de segurança da função.

- Há basicamente duas formas de atacar a resistência à colisão de uma função hash.

- A primeira é por criptoanálise e a segunda é por meio de um método conhecido como ataque do aniversário.

Segurança das funções hash

Ataque por criptoanálise

- Na **criptoanálise** das funções hash o criptoanalista foca na estrutura interna da função de compressão.
- A construção da maioria das funções hash atuais são projetadas com base em uma estrutura iterada que faz uso repetidamente de uma função de compressão.
- O criptoanalista estuda a mudança no padrão dos bits em cada iteração para tentar identificar a estrutura interna da função de compressão.

Ataque do aniversário

- O ataque do aniversário é uma técnica que busca diminuir o tempo necessário para encontrar uma colisão.
- O ataque é feito com base no **paradoxo do aniversário**.

Paradoxo do aniversário

- O paradoxo do aniversário é baseado na seguinte pergunta:

Quantas pessoas precisam estar reunidas para que a probabilidade de pelo menos duas delas compartilharem a mesma data de aniversário seja maior que 50%?

Segurança das funções hash

Paradoxo do aniversário

- Considere um grupo de n pessoas e um ano de 365 dias.
- Como existem muitas possibilidades de satisfazer esse problema (encontrar mais de duas pessoas aniversariando na mesma data), então é mais fácil calcular a probabilidade de todas as n pessoas fazerem aniversários em datas diferentes.
- A **primeira pessoa** do grupo tem 365 chances em 365 de não compartilhar a mesma data de aniversário com outra pessoa, assim a probabilidade da primeira pessoa não fazer aniversário na mesma data de alguém é $365/365 = 1$.

Paradoxo do aniversário

- A **segunda pessoa** pode fazer aniversário em qualquer um dos $\mathbf{364}$ dias restantes, menos no dia do aniversário da primeira pessoa, assim $\mathbf{364/365 \ = \ 0,9973}$.
- Observe que a quantidade de dias possíveis sem coincidir datas de aniversário **vai diminuindo**, portanto a probabilidade de haver colisões vai aumentando.

Paradoxo do aniversário

- Generalizando esta observação, a probabilidade $\overline{p}\,(n)$ de n pessoas terem aniversários diferentes é:

$$\overline{p}(n) = \left(\frac{365}{365}\right) \cdot \left(\frac{364}{365}\right) \cdot \left(\frac{363}{365}\right) \cdots \left(\frac{365 - (n-1)}{365}\right).$$

Então, esta equação pode ser escrita na forma:

$$\overline{p}(n) = \left(\frac{365!}{365^n(365-n)!}\right),$$

em que $\overline{p}(n)$ é a probabilidade de **não haver colisão** de datas, portanto a probabilidade de haver colisão é:

$$p(n) = 1 - \overline{p}(n).$$

Assim, para $n = 23$, a probabilidade de pelo menos duas pessoas compartilharem a mesma data de aniversário é de 50.7%.

Segurança das funções hash

O Ataque do aniversário nas funções hash

- O raciocínio do **paradoxo do aniversário** pode ser estendido para encontrar **colisões** em uma determinada função hash, pois o *problema do aniversário é apenas um problema específico.*
- O número 365 (representando a quantidade de dias do ano) será substituído por $\boldsymbol{n} = \mathbf{2}^{\boldsymbol{b}}$ em que $\boldsymbol{b}$ é quantidade de bits do valor hash e $\mathbf{2}^{\boldsymbol{b}}$ é quantidade de etiquetas hash possíveis.
- Assim, tomando como base o paradoxo do aniversário:

$$\boldsymbol{p}(\boldsymbol{k}) = \mathbf{1} - \left(\frac{\boldsymbol{n}!}{\boldsymbol{n}^{\boldsymbol{k}}(\boldsymbol{n}-\boldsymbol{k})!}\right),$$

calcula aproximadamente quantas entradas k para a função hash $\mathcal{H}$ são necessárias para gerar colisão com probabilidade de $\mathbf{0,5}$.

O Ataque do aniversário nas funções hash

- A partir da equação anterior, pode-se dizer que a quantidade de entradas $\boldsymbol{k}$ para gerar uma colisão, com probabilidade de $\mathbf{0,5}$ é:

$$\boldsymbol{k} = \sqrt{\mathbf{2}(\boldsymbol{ln}\mathbf{2})\boldsymbol{n}} = \mathbf{1.1774}\sqrt{\boldsymbol{n}}$$

- ou simplesmente

$$\boldsymbol{k} \approx \sqrt{\boldsymbol{n}}$$

O Ataque do aniversário nas funções hash

- A partir do ataque do aniversário, G. Yuval projetou um algoritmo chamado **Ataque pela raiz quadrada** (YUVAL, 1979).

- Dada uma função hash $\mathcal{H}$ com $\boldsymbol{n}$ bits de saída, o total de valores hash possíveis é igual a $\mathbf{2}^{\boldsymbol{n}}$.

- Produzir duas mensagens $\boldsymbol{m}$ e $\boldsymbol{m}$' tal que $\mathcal{H}(\boldsymbol{m}) = \mathcal{H}(\boldsymbol{m}')$ énecessário, aproximadamente, $\mathbf{2}^{\boldsymbol{n}/\mathbf{2}}$ tentativas.

- **Exemplo:** Dado $\boldsymbol{n} = \mathbf{32}$, então

$$\sqrt{\mathbf{2}^{\mathbf{32}}} = \mathbf{2}^{\mathbf{32}/\mathbf{2}} = \mathbf{65.536}$$

O Ataque do aniversário nas funções hash

- O ataque explora eficientemente uma propriedade específica das funções hash, a propriedade de resistência à colisão (propriedade 3).
- Observe que o ataque a essa propriedade, ainda que bem sucedido, não implica necessariamente na quebra geral de resistência à colisão da função atacada.
- **O ataque do aniversário não é eficiente contra a resistência à segunda pré-imagem.**

O Ataque do aniversário nas funções hash

- O nível de esforço computacional exigido para encontrar colisão em uma função hash com saída de tamanho n.
  - ***Resistência à pré-imagem*** = 2^n
  - ***Resistência à segunda pré-imagem*** = 2^n
  - ***Resistência à colisão*** = $2^{n/2}$
- Como o tamanho do conjunto imagem de uma função hash é muito menor que o domínio da função, **sempre haverá colisões** nesta função.

O que precisa ser feito para evitar o ataque é projetar uma função cujo o tamanho n da valor hash torne o cálculo de $\mathbf{2^{n/2}}$ *computacionalmente difícil.*

Fim!

[Aula 14] Funções hash